Radiosity

Leo Talbot

6/4/1999

Contents

Chapter 1

Introduction

1.1 Realistic Image Synthesis

Over the last few centuries physicists have developed mathematical models of the processes by which light interacts with surfaces and propagates through an environment. With the advent of the computer it became feasible to evaluate these models on a large scale. It is now possible to calculate an image given a scene geometry and material properties. This ability is important to applications ranging from industrial or architectural design to advertising and entertainment. The creation of images by evaluating a model of light propagation is called *image synthesis*. One such image synthesis method is called the *radiosity method*. This method contains one basic assumption: that all the surfaces in the environment reflect light diffusely. Analytical or numerical methods are then used to solve this mathematical model.

1.2 History

The radiosity method is a relative newcomer in the world of image synthesis. In 1984, researchers at Fukuyama and Hiroshima Universities in Japan, and at Cornell University in the United States began to apply radiosity methods from the field of radiative heat transfer to image synthesis. These methods were first developed in the 1950's for computing radiant interchange between surfaces, for engineering applications ranging from radiator and boiler design to aeronautics.

In image synthesis, radiosity methods are applicable to solving for the interreflection of light between ideal (Lambertian) diffuse surfaces. Initial algorithms were restricted to environments in which all surfaces could see each other. In following years, radiosity algorithms allowing occlusion were developed, and efficiency was improved through the use of a hierarchical subdivision of the environment.

Radiosity is a departure for image synthesis for several reasons. As opposed to the earlier empirical techniques, radiosity begins with an energy balance equation, which is then approximated and solved by numerical means. In contrast to ray tracing, which evaluates the illumination equation for directions and locations determined by the view and the pixels of the image, radiosity solves the illumination equation at locations distributed over the surfaces of the environment. This specification of the unknowns is *independent of the viewer position*, and thus radiosity methods are often called *view-independent* techniques. A final image can then by rapidly rendered using the complete solution. In this way, efficient interactive *walkthroughs* of simulated environments can be performed following the radiosity preprocess.

1.3 Radiosity and Finite Element Methods

Radiosity can be understood as a particular approach to solving the rendering equation under the assumption of Lambertian diffuse reflection. Heckbert and Winget [6] have shown that radiosity is essentially a finite element method.

Like Monte Carlo techniques, the finite element method is a broadly applicable approach to solving difficult integral equations, such as the rendering equation. The basic approach is to approximate an unknown function by subdividing the domain of the function into smaller pieces or *elements*, across which the function can be approximated using relatively simple functions like polynomials. The unknown function is thus projected into a *finite function space*, in which the approximated function is fully characterised by a finite number of unknowns. The resulting system can then be solved numerically.

The ideas underlying finite element methods were first discussed as early as the 1940s, although the term *finite element* did not become popular until the 1960s. The development of the finite element method closely parallelled related work in approximating functions using piecewise polynomials or *splines*.

It was not until computers became more routinely available in the 1960s and 1970s that these methods became a common technique for engineering analysis. Since then, there has been considerable research resulting in many working finite element codes and in a better theoretical understanding of convergence and other mathematical properties of such methods.

Chapter 2

Rendering Concepts

2.1 Summary of Photometric and Rendering Terms

2.1.1 Radiometric Terms

In most computer graphic systems, optical quantities are simply colours denoted by red, green and blue triplets. These triplets are used to specify many quantities including light sources, material properties and intermediate calculations. There are however a small but finite number of radiometric quantities that characterise the distribution of light in the environment. They are:

Radiant Energy: Energy, measured in *joules*.

Radiant Power: Energy per unit time. Can be thought of as *flux*, or the number of photons crossing a differential surface element each second. Measured in *watts*.

Radiance(L): Radiant energy per unit volume, or more precisely, power per unit projected area perpendicular to the ray per unit solid angle in the direction of the ray. The radiance distribution completely characterises the distribution of light in a scene. Note that it is a function of five independent variables, three that specify position and two that specify direction. All other radiometric quantities can be computed from it.

Irradiance(E): Radiant Energy per unit area falling on a surface.

Radiosity(B): Energy per unit area that leaves a surface.

Radiant Intensity(I): The radiant intensity in a given direction is equal to the irradiance at a point on the unit sphere centred at the source. This quantity is normally used for describing the energy distribution of a point light source.

2.1.2 The Bidirectional Reflection Distribution Function

Consider the light incident on a surface from a small differential solid angle in the direction $\vec{\omega}_i$. The amount of reflected light in another direction $\vec{\omega}_r$ is proportional to the incident irradiance from $\vec{\omega}_i$. The constant of proportionality is termed the *bidirectional reflection distribution function*, or BRDF. The BRDF is bidirectional because it depends on two directions. It is a distribution function because it is strictly positive. Notice that adding light from another incident direction has no influence on the amount of light reflected from other incident directions. Thus, reflection behaves linearly, and hence the total amount of light reflected by a surface in a specific direction is given by a hemispherical integral over all possible incident directions. This leads to the *reflection equation:*

$$L_r(\vec{\omega}_r) = \int_{\Omega_i} f_r(\vec{\omega}_i \to \vec{\omega}_r) L_i(\vec{\omega}_i) \cos\theta_i d\omega_i \tag{2.1}$$

Put another way, the reflected radiance in a particular direction is due to the radiance arriving from all directions weighted by the BRDF relating the incoming and reflected directions and by the projected solid angle.

2.2 The Rendering Equation

The reflectance equation makes it possible to compute the reflected light distribution from the incident light distribution and the BRDF of the material. The important remaining task is to specify, or preferably to compute, the incident light distribution. This is typically referred to as the *illumination model.*

The first step in using a global illumination model is to relate the illumination on one surface to the reflected light distribution from another surface. This requires that the spatial dependence of radiance is made explicit and that occlusion is considered.

Using the fact that radiance is invariant along a ray, the incident radiance at **x**' due to the radiance from **x** is

$$L_i(\mathbf{x}', \vec{\omega}_i') = L_o(\mathbf{x}, \vec{\omega}_o) V(\mathbf{x}, \mathbf{x}') \tag{2.2}$$

where $\vec{\omega}_i$ is a direction vector from **x'** to **x**, and $\vec{\omega}_o$ is in the opposite direction.

$$\vec{\omega}_i = -\vec{\omega}_o = \frac{\mathbf{x} - \mathbf{x}'}{|\mathbf{x} - \mathbf{x}'|} \tag{2.3}$$

The function $V(\mathbf{x}, \mathbf{x}')$ is a visibility function. It is 1, if **x** and **x**' are mutually visible; otherwise it is 0.

The next step is to change the hemispherical integral over all incident directions to an area integral over all the other surfaces in the environment. This is easily done by relating the solid angle subtended by the source to its projected surface area.

$$d\omega_i' = \frac{\cos\theta_o dA}{|\mathbf{x} - \mathbf{x}'|^2} \tag{2.4}$$

Dotting this to form the projected solid angle results in

$$d\omega_i' \cos\theta_o dA = G(\mathbf{x}, \mathbf{x}') dA \tag{2.5}$$

where

$$G(\mathbf{x}, \mathbf{x}') = G(\mathbf{x}', \mathbf{x}) = \frac{\cos\theta_i' \cos\theta_o}{|\mathbf{x} - \mathbf{x}'|^2} \tag{2.6}$$

Substituting $G(\mathbf{x}, \mathbf{x}')$ into the reflectance equation leads to the following integral equation over the surfaces, S:

$$L(\mathbf{x}', \vec{\omega}') = \int_S f_r(\mathbf{x}) L(\mathbf{x}, \vec{\omega}) G(\mathbf{x}, \mathbf{x}') V(\mathbf{x}, \mathbf{x}') dA \tag{2.7}$$

Since this equation only involves outgoing radiances and directions, the subscripts denoting incoming and outgoing directions can safely be dropped (except from f_r). This equation was first introduced to the computer graphics literature by Kayija [7], who appropriately named it the *rendering equation.*

There is one final step required to arrive at the full rendering equation, and that is account for all modes of light transport at a surface. In an environment consisting only of opaque surfaces, the only other source of light is due to emission from the surface.

$$L(\mathbf{x}', \vec{\omega}') = L_e(\mathbf{x}', \vec{\omega}') + \int_S f_r(\mathbf{x}) L(\mathbf{x}, \vec{\omega}) G(\mathbf{x}, \mathbf{x}') V(\mathbf{x}, \mathbf{x}') dA \tag{2.8}$$

where L_e is the two point intensity of emitted light.

2.3 The Radiosity Equation

The rendering equation can be simplified given the *radiosity assumption.* In radiosity, it is assumed that all surfaces in the environment are Lambertian diffuse reflectors. Thus the BRDF is independent of the incoming and outgoing directions and can be taken out from under the integral.

$$\begin{aligned} L(x' \to x'') &= L_e(x' \to x'') + f_r(x') \int_S L(x \to x') G(x, x') V(x, x') dA \\ &= L_e(x' \to x'') + \frac{\rho(x')}{\pi} \int_S L(x \to x') G(x, x') V(x, x') dA \end{aligned} \tag{2.9}$$

More importantly, the outgoing radiance from a Lambertian surface is the same in all directions and in fact equals the radiosity B divided by π. this leads to even more dramatic simplifications[1].

$$B(x) = E(x) + \rho(x) \int_S B(x') \frac{G(x, x')}{\pi} dA' \tag{2.10}$$

The rendering equation expresses the conservation of light energy at all points in space. The key feature of such an integral equation is that the quantity to be computed - in this case, the radiance or radiosity - appears on the left-hand side as well as under the integral on the right-hand side. For this reason, integral equations are notoriously difficult to solve. They very rarely have analytic solutions, and numerical methods must be used.

[1] Note the switch in notation. E is the energy per unit area emitted by the surface, or $\frac{L_e}{\pi}$.

Chapter 3

Discretising the Radiosity Equation

3.1 The Radiosity Equation

Repeating the radiosity equation:

$$B(\mathbf{x}) = E(\mathbf{x}) + \rho(\mathbf{x}) \int_S B(\mathbf{x}')G(\mathbf{x}, \mathbf{x}')dA' \tag{3.1}$$

where the geometric term, $G(\mathbf{x}, \mathbf{x}')$, now includes the visibility term, $V(\mathbf{x}, \mathbf{x}')$, and division by π.

The dimension of the *function space* of the radiosity function, $B(\mathbf{x})$, is infinite. This means that solving the radiosity equation for a point $\mathbf{x}$ on a surface does not determine the radiosity at an immediately neighbouring location. As a result, a full and exact solution to the radiosity equation requires either finding the exact functional form of the radiosity across each surface or computing radiosity values for an infinite number of surface points. This is an impossible task, so instead, the problem is reformulated as one that can be solved for a finite set of unknown values, which can then be used to construct an *approximate* solution. In radiosity, the finite element method is used to derive a linear system of equations that can be solved for discrete unknowns. These unknowns then define an approximation to the radiosity function.

3.2 The Radiosity Approach

As stated earlier, the radiosity method can be viewed as a simple finite element formulation. Finite element methods represent a complicated function by subdividing the domain into elements over which the function is approximated by a linear sum of simple *basis functions,* also sometimes called *shape functions.* The contribution of each basis function to the approximation is determined by values computed at a finite number of locations, or *nodes.* These nodal values become the unknowns in a linear system of equations, which can be solved using a variety of techniques.

The basic steps to formulating and solving a generic radiosity problem are outlined here:

1. Subdivide the surfaces into *elements.*
2. Select locations or *nodes* on the elements at which to determine discrete radiosity values. These will ultimately become the finite set of unknowns in the linear system of equations.
3. Assume a simple functional form for the variation of radiosity across an element. This is accomplished by associating a *basis function* with each node. The basis functions are typically simple polynomials with *local support* (i.e., zero everywhere except in the neighbourhood of the associated node). When weighted by the nodal radiosity values, the basis functions for a particular element sum to define and approximate radiosity at each point on the element.
4. Select a finite error metric to minimise by *projecting* a *residual function* onto a set of basis functions. This approximates the error as a finite sum, thus casting the infinite dimension problem into a finite set of linear equations.
5. Compute the coefficients of the linear system. These coefficients are primarily based on the geometric relationships that determine the transport of light between elements. The geometric relationships are called *form factors.*
6. Solve the resulting system of equations for the unknown nodal radiosity values.
7. Reconstruct the approximate radiosity solution as the linear sum of the basis functions weighted by the resulting nodal values from step 6.
8. Render an image, deriving shading values from the radiosity approximation as needed for the particular view. The approximation used during the solution may be modified during this stage to meet the particular requirements of rendering.

3.3 Approximating Radiosity across a Surface

The heart of the finite element method is the approximation of a complicated function by a linear sum of a finite number, n, of simpler basis functions, $N_i(\mathbf{x})$, each of which is nonzero over a limited region of the function domain. The domain of the function is subdivided into a *mesh* of elements, each of which has one or more nodes located at points, $\mathbf{x}_i$, inside or on the element boundary. The mesh of elements and nodes organises the basis functions.

Each node is associated with a basis function. Each basis function, although defined over the full domain of the original unknown function, is chosen to have a zero value everywhere except over a small region of the domain. This *support* of the basis function is confined to the elements adjacent to the node. This limited support implies that the coefficient of single basis functions will only effect a small range of the overall function. The approximation within an element is thus determined by summing only the few basis functions whose support overlaps the element, with the contribution of each basis function weighted by a coefficient value, B_i, at the associated node. The radiosity function, $B(\mathbf{x})$, is thus approximated by $\hat{B}(\mathbf{x})$ where

$$B(\mathbf{x}) \approx \hat{B}(\mathbf{x}) = \sum_{i=1}^{n} B_i n_i(\mathbf{x}) \tag{3.2}$$

Evaluating $\hat{B}(\mathbf{x})$ at a particular point $\mathbf{x}_j$ involves summing only those basis functions with their support over the point.

There are many possible basis functions. Low-order polynomials are the most common, including constant, linear, quadratic, and cubic functions. The constant (or "box") basis, which is often used in radiosity, is defined by

$$N_i(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbf{x} \text{ is outside element} \\ 1 & \text{if } \mathbf{x} \text{ is inside element} \end{cases} \tag{3.3}$$

The number and placement of nodes within elements will depend on the order of the polynomial. In the case of constant elements, defined by constant basis functions, a single node located in the centre of the element is commonly used.

3.4 Error Metrics

A set of discrete elements, with a total of n nodal points, $\mathbf{x}_i$, and n basis functions, N_i, defines a *finite dimension function space.* This finite function space has n degrees of freedom, the n nodal values, B_i, thus $\hat{B}(\mathbf{x})$ is everywhere defined by the finite set of the n coefficients, B_i.

Ideally, one would like the approximate radiosity solution $\hat{B}(\mathbf{x})$ to agree with the actual solution $B(\mathbf{x})$ everywhere. This is not possible in general since $\hat{B}(\mathbf{x})$ is restricted to the finite subspace of functions representable as the linear sum of the selected basis functions. Given the choice of basis functions, the goal is then to find the nodal values that generate a solution *minimising* the error according to some measure or *metric.*

The actual error, $\epsilon(\mathbf{x})$, is the difference between the approximate and exact solutions, that is,

$$\epsilon(\mathbf{x}) = B(\mathbf{x}) - \hat{B}(\mathbf{x}) \tag{3.4}$$

Unfortunately, it is impossible to determine $\epsilon(\mathbf{x})$ directly since the exact solution is not known.

An alternative characterisation of the error is provided by the residual. If $\hat{B}$ is substituted for both occurrences of B in the original radiosity equation

$$\hat{B}(\mathbf{x}) = E(\mathbf{x}) + \rho(\mathbf{x}) \int_S \hat{B}(\mathbf{x}')G(\mathbf{x}, \mathbf{x}')dA' \tag{3.5}$$

then the *residual* function is given by the difference of the left and right hand sides

$$r(\mathbf{x}) = \hat{B}(\mathbf{x}) - E(\mathbf{x}) - \rho(\mathbf{x}) \int_S \hat{B}(\mathbf{x}')G(\mathbf{x}, \mathbf{x}')dA' \tag{3.6}$$

An exact solution will make $r(\mathbf{x})$ zero everywhere, but this is unobtainable since, as we have seen, $\hat{B}$ is restricted to the subspace of functions realisable using the basis functions. Although $\hat{B}$ is restricted to a finite function space, the residual

r is not. However, a similar approach to that taken in approximating B can be taken to project the residual into a finite function space. In this way, rather than seek a solution technique that makes $r(\mathbf{x})$ zero everywhere, the goal will instead be to minimise $r(\mathbf{x})$ according to a *finite dimensional* error metric.

The general approach is to choose n independent *weighting functions*, $W_i(\mathbf{x})$, each having local support, much like the earlier basis functions. The *norm* or size of the residual is then approximated as a finite sum:

$$|r(\mathbf{x})| = \sum_{i=1}^{n} | < r(\mathbf{x}), W_i(\mathbf{x}) > | \tag{3.7}$$

This residual norm can be driven to zero by finding radiosity values such that each of the n terms $< r(\mathbf{x}), W_i(\mathbf{x}) >$ is equal to zero. As will be derived below, setting each of these terms to zero results in a linear equation. The set of n such linear equations (one for each weighting function of the n nodal values, B_i), can be solved simultaneously.

This approach defines a general class of methods called *weighted residual methods*. Two different choices for the weighting functions lead to two contrasting formulations, *point collocation* and the *Galerkin method* which will be discussed in the following sections.

3.4.1 Point Collocation

The simplest set of weighting functions are *delta* functions:

$$W_i(\mathbf{x}) = \delta(\mathbf{x} - \mathbf{x}_i) \tag{3.8}$$

which are zero unless $\mathbf{x}$ is coincident with the node at $\mathbf{x}_i$.

Using these weighting functions, the norm of the residual is minimised when $r(\mathbf{x})$ is exactly zero at all the node points. This clearly differs from requiring $r(\mathbf{x})$ to be zero everywhere. However, as the number of nodes increases this difference diminishes. In general, the points at which the residual function is minimised are selected to be located at the nodes used in approximating the radiosity function. This technique is known as *point collocation*.

If there are n nodes, there are now a finite number of conditions that must be met. These are captured by the n simultaneous linear equations, one for each node i located at location $\mathbf{x}_i$:

$$\hat{B}(\mathbf{x}_i) - E(\mathbf{x}_i) - \rho(\mathbf{x}_i) \int_S \hat{B}(\mathbf{x}')G(\mathbf{x}_i, \mathbf{x}')dA' = 0, \forall i \tag{3.9}$$

Expanding $\hat{B}$ gives

$$\sum_{j=1}^{n} B_j N_j(\mathbf{x}_i) - E(\mathbf{x}_i) - \rho(\mathbf{x}_i) \int_S \sum_{j=1}^{n} B_j N_j(\mathbf{x}')G(\mathbf{x}_i, \mathbf{x}')dA' = 0 \tag{3.10}$$

Grouping terms and factoring B_j, which is independent of $\mathbf{x}'$, out of the integral leaves

$$\left[\sum_{j=1}^{n} B_j \left[N_j(\mathbf{x}_i) - \rho(\mathbf{x}_i) \int_S N_j(\mathbf{x}')G(\mathbf{x}_i, \mathbf{x}')dA'\right]\right] - E(\mathbf{x}_i) = 0 \tag{3.11}$$

This can be restated simply as the set of n linear equations

$$\sum_{j=1}^{n} B_j K_{ij} - E(\mathbf{x}_i) = 0 \tag{3.12}$$

or in matrix/vector form as

$$\mathbf{KB} = \mathbf{E} \tag{3.13}$$

The coefficients K_{ij} are given by

$$K_{ij} = N_j(\mathbf{x}_i) - \rho(\mathbf{x}_i) \int_S N_j(\mathbf{x}')G(\mathbf{x}_i, \mathbf{x}')dA' \tag{3.14}$$

These coefficients are independent of the nodal radiosities and are defined solely by the geometry and material properties of the environment.

3.4.2 Galerkin Form of Weighted Residuals

An alternative approach to deriving a linear system of radiosity equations defines the approximate residual to be zero when a set of *weighted integrals* of the residual are zero.

The weighting functions in this case are selected to have local support much like the basis functions used to approximate the radiosity function. In this case the *weighted residual method* seeks a solutions for which there is a weighted "average" zero residual over each small region of the domain. If n weighting functions, $W_i(\mathbf{x})$, are defined, a solution is found if

$$< W_i(\mathbf{x}), r(\mathbf{x}) >= \int_S W_i(\mathbf{x}) r(\mathbf{x}) dA = 0, \forall i \tag{3.15}$$

or expanding $r(\mathbf{x})$ above

$$\begin{aligned} &0 = \textstyle\int_S W_i(\mathbf{x})\hat{B}(\mathbf{x})dA - \\ &\textstyle\int_S W_i(\mathbf{x})E(\mathbf{x})dA - \int_S W_i(\mathbf{x})\rho(\mathbf{x}) \int_S \hat{B}(\mathbf{x}')G(\mathbf{x}, \mathbf{x}')dA'dA \end{aligned} \tag{3.16}$$

The *Galerkin* formulation selects the same basis functions used to approximate the radiosity function as the weighting functions. Thus

$$0 = \int_S N_i(\mathbf{x})\hat{B}(\mathbf{x})dA - \int_S N_i(\mathbf{x})E(\mathbf{x})dA - \int_S N_i(\mathbf{x})\rho(\mathbf{x}) \int_S \hat{B}(\mathbf{x}')G(\mathbf{x}, \mathbf{x}')dA'dA \tag{3.17}$$

Finally, expanding $\hat{B}$ and grouping terms as before results in:

$$\begin{aligned} &\left[\textstyle\sum_{j=1}^{n} B_j \left[\int_S N_i(\mathbf{x})N_j(\mathbf{x})dA - \int_S N_i(\mathbf{x})\rho(\mathbf{x}) \int_S N_j(\mathbf{x}')G(\mathbf{x}, \mathbf{x}')dA'dA\right]\right] \\ &\qquad - \textstyle\int_S E(\mathbf{x})N_i(\mathbf{x})dA = 0 \end{aligned} \tag{3.18}$$

The unknowns, B_j, have been isolated in this expression. There are n such expressions, one for each node i and, as with point collocation, the linear system of equations can be expressed as the matrix equation

$$\mathbf{KB} = \mathbf{E} \tag{3.19}$$

3.5 Constant Element Radiosities

The relationships of the above equations to the physical problem of global illumination may still be somewhat obscure. The complexity in the above formulations is reduced if constant basis functions are used, with the reflectivity and emission of each element assumed to be constant.

For constant basis functions, the Galerkin formulation can be rewritten by performing the integration for each term in equation 3.18. First, using the fact that the box basis functions have values of only 1 and 0, and do not overlap with one another,

$$\int_S N_i(\mathbf{x})E(\mathbf{x})dA = \left\{ \begin{array}{cc} A_i & \text{if } i=j \\ 0 & otherwise \end{array} \right\} = \delta_{ij}A_i \tag{3.20}$$

where δ_{ij} is the Kronecker delta, and A_i is the area of element i since the corresponding basis functions are zero elsewhere. Likewise,

$$\int_S E(\mathbf{x})N_i(\mathbf{x})dA = E_iA_i \tag{3.21}$$

where E_i is the area average emission value for element i.

Finally, since the basis functions have unit value within the element, the basis functions themselves can be dropped and it is possible to integrate explicitly over element areas. Thus, assuming the reflectivity $\rho(\mathbf{x})$ is a constant ρ_i over A_i,

$$\int_S N_i(\mathbf{x})\rho(\mathbf{x}) \int_s N_j(\mathbf{x}')G(\mathbf{x},\mathbf{x}')dA'dA = \rho_i \int_{A_i}\int_{A_j} G(\mathbf{x},\mathbf{x}')dA_jdA_i \tag{3.22}$$

Making these substitutions into equation 3.18, dividing by A_i and moving the emission term to the right side results in, for all i:

$$\sum_{j=1}^{n} B_j\left[\delta_{ij} - \rho_i F_{ij}\right] = E_i \tag{3.23}$$

where F_{ij}, called the *form factor*, is given by

$$F_{ij} = \frac{1}{A_i}\int_{A_i}\int_{A_j} G(\mathbf{x},\mathbf{x}')dA_jDA_i \tag{3.24}$$

The form factor represents the fraction of energy that leaves element i and arrives directly at element j.

Rearranging equation 3.23 gives the *classical radiosity equation*:

$$B_i = E_i + \rho_i \sum_{j=1}^{n} B_jF_{ij} \tag{3.25}$$

Chapter 4

The Form Factor

4.1 The Coefficients of K

The matrix **K** results from the projection of the radiosity function onto a selected basis set followed by the projection of the residual onto a basis. The matrix **K** consists of the difference between two matrices, $\mathbf{M} - \mathbf{PF}$, where **M** accounts for the overlap (inner product) of neighbouring basis functions, **P** is a diagonal matrix of surface reflectivities, and **F** is a matrix of form factors.

The coefficients of the matrix **F** represent the weighted direct influence of the radiosity of a region under the support of one basis function to the radiosity of another such region. In the case of constant basis functions, the **M** matrix is simply a diagonal matrix of element areas. Thus, each equation in the system can be divided by the area of the corresponding element. Then **M** is simply the identity and

$$F_{ij} = \frac{1}{A_i} \int_{A_i} \int_{A_j} G_{ij} dA_j dA_i \tag{4.1}$$

where the geometric kernel is now denoted by G_{ij}. In this case, the coefficients F_{ij} take on a particularly intuitive physical meaning: F_{ij} is the fraction of light leaving element i that arrives directly at element j. The F_{ij} in this case are commonly called *form factors*.

The form factor is purely a function of the geometric relationship between elements and thus does not depend on the reflectivity of the surfaces. This is a potential advantage of the radiosity method, since it allows surface attributes to be changed without repeating the expensive form factor computation.

4.2 Two Formulations of the Form Factor

The full form factor integral for constant bases involves performing an integration over the support of two basis functions. Two different formulations of this form factor integral between elements can be derived.

The first formulation results from inserting the differential form factor into the form factor integral, equation 4.1. Thus, the element-to-element form factor is the *double area* integral

$$F_{ij} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi r^2} V_{ij} dA_j dA_i \tag{4.2}$$

The second formulation results from replacing the inner integration over area with an equivalent integration over the hemisphere around dA_i. The differential solid angle $d\omega_j$ from dA_i to dA_j is

$$d\omega_j = \frac{\cos\theta_j}{r^2} dA_j \tag{4.3}$$

Thus, the inner integral of equation 4.2 can be rewritten over the hemisphere of directions, Ω, and over dA_i, resulting in the *area-hemisphere* integral,

$$F_{ij} = \frac{1}{A_i} \int_{A_i} \int_{\Omega} \frac{\cos\theta_i}{\pi} V_{ij} d\omega_j dA_i \tag{4.4}$$

This alternate form of the integral will be used in some algorithms to compute the form factors from dA_i to *all* elements at once.

4.3 Computing the Form Factor

Both closed form analytic and numerical methods have been applied to solving the form factor integral. Closed form analytic formulae do not lend themselves to the direct evaluation of form factors between complex shapes or where occlusion is a factor. For more general situations, numerical approaches are required to approximate the form factor.

Numerical methods of evaluating integrals (known as *quadrature* methods) generally involve sampling the kernel at various points and approximating the integral as a weighted sum of the kernel evaluated at the sample points. In general, the more sample points selected, the more accurate the approximation, and the greater the computational cost. Thus, the goal in developing a numerical algorithm to solve the form factor integral is to get the most accuracy with the fewest (and/or cheapest) kernel evaluations.

There is a matrix of form factors to be evaluated. Each entry of a row/column of the form factor matrix shares a common element, thus one may take advantage of this coherence by simultaneously solving for a complete row and/or column of form factors.

4.4 Evaluating the Inner Integral

The area-area and the area-hemisphere forms of the double integral both share the same outer integral over element i. Thus, one can separate the question of evaluating the full integral into two parts: first, the evaluation of the inner integral from some dA_i, and second, the selection of one or more sample points on element i at which to evaluate the inner integral.

In fact, many implementations use only one representative point on element i. This assumes that the inner integral is constant across element i, which may be reasonable when the distance (squared) is much greater that the area of element j. However, change in visibility between elements i and j also affect the validity of this approximation.

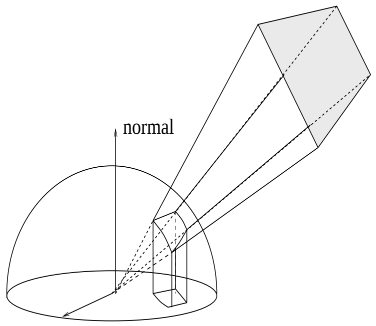

Figure 4.1: Nusselt Analogue

4.4.1 Hemisphere Sampling Algorithms

The use of the hemispherical integral provides a direct basis for evaluating the form factor from a differential area to *all* elements simultaneously. The geometric kernel, $G_{i\omega}$, is given by

$$G_{i\omega} = \frac{\cos\theta_i}{\pi} V_{ij} \tag{4.5}$$

The visibility term, V_{ij}, which is 1 if element j is visible in direction $d\vec{\omega}$, is the computationally expensive part of the kernel evaluation.

Hemisphere sampling approaches can be principally differentiated by the hidden surface algorithm they use. Certain algorithms provide efficiency by exploiting coherence in the geometry of visible surfaces, while others provide flexibility for stochastic, adaptive or nonuniform sampling.

4.4.2 Nusselt Analogue

The Nusselt analogue provides a geometric analogue to the differential area-to-area form factor equation. An imaginary unit hemisphere is centred on the differential area, as in Figure 4.1. The element is projected radially onto the hemisphere, and then orthogonally down from the hemisphere onto its base. The fraction of the base area covered by this projection is equal to the form factor.

4.4.3 The Hemicube

The Nusselt analogue illustrates the fact that if an element is projected radially onto any intermediate surface, the form factor for the projection will be the same as for the element itself. This observation forms the basis for the *hemicube* form factor algorithm, in which elements are projected onto the planar faces of a half cube.

A hemicube is placed around a differential area, with the hemicube faces subdivided into small grid cells. Each grid cell defines a direction and a solid angle. A *delta form factor*, ΔF, is computed for each cell, and stored in a lookup table. Only one eighth of the delta form factors need be stored, due to symmetry.

The form factor to an element is then approximated by projecting the element onto the hemicube and summing the delta form factors of the grid cells covered by the projection. The visibility problem of determining the closest surface for a regular grid of cells is a very familiar one. The hemicube uses the Z-buffer algorithm, which is simple and efficient, and has the advantage of wide availability in hardware graphics accelerators. The only difference between rendering onto the hemicube and normal image rendering is that in addition to a Z depth, and ID for the visible element is stored at each grid cell, instead of a colour. The form factor to element j is thus

$$F_{i,j} = \sum_{q \in j} \Delta F_q \tag{4.6}$$

where q represents delta grid cells covered by element j.

The efficiency of the hemicube derives from the incremental nature of the Z-buffer algorithm, which exploits coherency in the projected geometry. However, the inflexibility of the Z-buffer is also the source of many of the hemicube's disadvantages.

The hemicube divides the domain of integration into discrete, regularly spaced solid angles. When projected elements cover only a few of the grid cells, aliasing can occur. Very small elements may fall between grid cells and be missed entirely. This is a particular problem for small, high energy elements like light emitters. Increasing the hemicube resolution can help, but this is an inefficient solution. Since scan conversion requires a uniform grid, the hemicube spends as much time calculating form factors close to the horizon as it does near the normal direction.

The advantage of the hemicube is that it determines form factors to all elements from a single point. This can be a liability if only one form factor is required. In addition, computing the full area-hemisphere form factor requires repeating the hemicube at a sufficient number of sample points on element i to ensure the desired accuracy for elements that are relatively close together.

4.4.4 Single-Plane Method

The *single-plane* form factor algorithm developed by Sillion [12] partially addresses the inflexibility of the hemicube by replacing the Z-buffer with an adaptive hidden surface algorithm based on Warnock [14]. Sillion's algorithm projects elements onto a single plane above the differential area dA_i. Warnock's algorithm is used to subdivide the image plane adaptively into *windows* for which the element visibility can be determined trivially. Sillion's algorithm is able to subdivide coarsely for large elements and finely for small elements, and factors associated with regions on the projection plane can be precomputed.

Sillion's algorithm can also compute form factors that take into account specular surfaces. Rays are cast through the corners of the region if the visible element is specular and traced recursively until a diffuse element is encountered.

The single-plane method will miss elements near the horizon, but these elements will typically contribute very little to the overall radiosity, due to the cosine dependence of the form factor.

4.4.5 Monte Carlo Ray Tracing

Ray tracing provides an extremely flexible basis for evaluating the visibility term in the numerical integrand. Since rays are cast independently they can be shot to and from any distribution of points on the elements or directions in the hemisphere. Nonuniform and adaptive sampling can be used to distribute computational effort evenly. Rays can also be distributed stochastically, which can render the effects of inadequate sampling less noticeable by converting aliasing to noise.

Malley describes a straightforward method for generating a set of sample directions with the desired distribution [9]. Malley's method is essentially a Monte Carlo evaluation of Nusselt's Analogue run in reverse. He begins by generating a set of random points uniformly distributed in the circle under the hemisphere. To determine a direction to shoot a ray, one of these points is projected vertically onto the hemisphere. The ray is then directed radially out from the centre of the hemisphere through the projected point. Rays are shot for every point on the circle. The number of times each element in the scene is hit is recorded. The form factor is then given by the proportion of rays that hit the element.

4.4.6 Area Sampling Algorithms

The hemisphere sampling algorithms described above are most efficient when form factors to all elements from a single point must be computed at once. Certain solution techniques (e.g., progressive refinement) require form factors between only a single pair of elements at a time. In this case, the area-area formulation (equation 4.2) is a more efficient basis for algorithm development.

The integration can be performed by evaluating the kernel at randomly distributed points for a Monte Carlo solution. Alternatively, the integration can be performed by subdividing the area uniformly.

4.5 Full Area-to-Area Quadrature

Any of the differential form factor solutions discussed so far can be used to approximate the full area to area form factor. The differential area-to-area form factor is evaluated at one or more points on element A_i, and the result averaged. However, since many rays connecting the two surfaces originate at the same points on A_i, this approach samples A_i inefficiently.

The double area integral can be approximated more accurately by distributing the endpoints of the rays over both elements. In a Monte Carlo approach ray endpoints on both elements would be distributed randomly, or according to some quasi-random distribution like the Poisson disk.

Chapter 5

Radiosity Matrix Solutions

The linear equations describing the radiosity problem are

$$\mathbf{KB} = \mathbf{E} \tag{5.1}$$

where $\mathbf{K}$ is a matrix of interactions, $\mathbf{B}$ is a vector of nodal radiosity values, and $\mathbf{E}$ is a vector of emission terms.

5.1 Qualities of the K Matrix

- *Size:* In general, the matrix is square and of size n by n, where n is the number of basis functions. In *hierarchical* methods, rectangular matrices appear.
- *Sparsity:* The matrix will typically be more full that sparse. The ij^{th} position of the matrix will be zero only when the reflectivity of a surface is zero, or when element i cannot see element j.
- *Symmetry:* $\mathbf{K}$ is not symmetric in its current form. However, a simple transformation can produce an equivalent system in which the matrix is symmetric. If each row i is multiplied by the area of the i^{th} element, then the $\mathbf{F}$ matrix is made symmetric. $\mathbf{K}$ can also be premultiplied by the inverse of the reflectivities to complete the symmetry, but zero-valued reflectivities will be a problem. Hence, symmetry will not be relied on in the solution methods described below.
- *Diagonal Dominance:* A matrix is said to be diagonally dominant if the absolute value of the sum of the off-diagonal terms is less than or equal to the absolute value of the diagonal term itself. This is strictly true for the radiosity problem with constant basis functions. The diagonal dominance of $\mathbf{K}$ ensures that particular iterative solution methods such as Gauss-Seidel (see section 5.3.2)will converge.
- *Spectral Radius:* The spectral radius of a matrix is a particular type of norm that describes the size of its largest eigenvalue. It is an indicator of the speed at which iterative methods will converge.

- *Condition:* The condition number of a matrix describes how sensitive the solution is to small perturbations in the input. In general, the matrix arising in radiosity applications is *well-conditioned*, indicating that most solution methods are applicable.

5.2 Linear System Solution Methods

5.2.1 Direct Methods

Direct methods for solving linear systems are perhaps most familiar but are not well suited to large systems of equations. Such methods, like Gaussian elimination, can be applied to systems such as the radiosity problem, but they have a computational complexity of $O(n^3)$. Thus, these methods are unsuited for large systems. Iterative solution methods are therefore the focus of this section.

5.2.2 Iterative Methods

Iterative methods begin with a guess for the solution and proceed by performing inexpensive operations that move the guess to a better guess. Given the linear system

$$\mathbf{KB} = \mathbf{E} \tag{5.2}$$

containing the unknown vector $\mathbf{B}$, and an initial guess, $\mathbf{B}^{(0)}$, the error is defined by the residual, $r^{(0)}$, where

$$\mathbf{r}^{(0)} = \mathbf{KB}^{(0)} - \mathbf{E} \tag{5.3}$$

Clearly, if the solution guess is correct, then the residual is zero.

The initial guess, $\mathbf{B}^{(0)}$, can influence the rate of convergence. In general, an initial guess that is closer to the final solution will require fewer iterations. If there is no information available upon which to base an initial guess, $\mathbf{B}^{(0)}$ can simply be a vector of zeros. For radiosity, a good initial guess is $\mathbf{B}^{(0)} = \mathbf{E}$.

5.3 Relaxation Methods

The types of iterative algorithms explored below are called relaxation methods. The idea is that at each step of the algorithm, one element of the residual vector will be set to zero. Of course, other elements in the residual may have increased, but hopefully an improvement has been made overall.

5.3.1 Jacobi Iteration

Perhaps the simplest iterative scheme is to update each element $B_i^{(k)}$ of the solution vector by solving for that variable using the current guess $B^{(k)}$. Thus n steps (one iteration) can be performed simultaneously (one for each i). Solving for a single B_i, beginning with the i^{th} row from the matrix equation 5.1

$$\sum_j K_{ij} B_j = E_i \tag{5.4}$$

moving all but the i^{th} term to the right hand side

$$K_{ii}B_i = E_i - \sum_{j \neq i} K_{ij}B_j \tag{5.5}$$

and dividing by K_{ii} results in a new value for B_i for iteration $k + 1$

$$B_i^{(k+1)} = E_i - \sum_{j \neq i} K_{ij}\frac{B_j^{(k)}}{K_{ii}} \tag{5.6}$$

With the definition of the residual vector (equation 5.3), this can be simplified by adding the residual divided by the diagonal to the previous guess,

$$B_i^{(k+1)} = B_i^{(k)} + \frac{r_i^{(k)}}{K_{ii}} \tag{5.7}$$

Thus, *if the residual vector is known* the full iteration (all n steps) can be performed in linear time, $O(n)$. Although each step in isolation would relax one element of the residual if the old solution were still valid, none other that the first element of B dealt with would actually result in zeroing out the corresponding residual. Thus, this is not, strictly speaking a relaxation method.

5.3.2 Gauss-Seidel Iteration

Gauss-Seidel iteration is a slight variation of the Jacobi method. It provides a true relaxation method and will usually improve the speed at which the solution converges. At each step during an iteration, the Gauss-Seidel method uses the most up-to-date version of the solution vector, rather than the solution vector computed by the previous iteration. Thus, at each of the n steps of an iteration, the new entry of $\mathbf{B}$ is computed using the values $B^{(k+1)}$ computed by previous steps of the current iteration. Otherwise, values computed during the previous iteration are used. Thus, to relax r_i, set

$$B_i^{(k+1)} = E_i - \sum_{j=1}^{i-1} K_{ij}\frac{B_j^{(k+1)}}{K_{ii}} - \sum_{j=i+1}^{n} K_{ij}\frac{B_j^{(k)}}{K_{ii}} \tag{5.8}$$

Performing the step for a single entry of $\mathbf{B}$ requires taking the dot product of the current guess, $\mathbf{B}^{(k),(k+1)}$, with a row of the matrix, and thus requires $O(n)$ operations. Consequently, a full iteration consisting of n steps is $O(n^2)$. Alternatively, since

$$r_i^{(k)} = E_i - \sum_{j=0}^{n} K_{ij}B_j^{(k)} \tag{5.9}$$

if all the residuals are updated after *each step*, one can set

$$B_i^{(k+1)} = B_i^{(k)} + \frac{r_i^{(k)}}{K_{ii}} \tag{5.10}$$

In this case the cost of a step is reduced to $O(1)$, but the residual update will be $O(n)$ for each step and a full iteration is again $O(n^2)$.

5.3.3 Southwell Iteration

A variation of the Gauss-Seidel method provides an alternative algorithm with a slightly different physical interpretation. Rather than relax each residual in turn, the row i with the largest residual will always be selected.

Thus, a single step, p of the Southwell algorithm is

$$\text{For}\, i, \,\text{such that}\, r_i = Max(\mathbf{r}) : B_i = E_i - \sum_{j \neq i} \frac{B_j K_{ij}}{K_{ii}} \tag{5.11}$$

It would seem that Southwell would require $O(n^2)$ operations to compute all the r_i's before picking the greatest one. (The computation of each r_i above involves computing the dot product of $\mathbf{B}^{(p)}$ with the row $\mathbf{K}_i$.) Fortunately, if at some step p, $\mathbf{r}^{(p)}$ is known for a given $\mathbf{B}^{(p)}$, the next approximation can be made if the changes $\Delta\mathbf{B}^{(p)}$ are known. Thus

$$\mathbf{B}^{(p+1)} = \mathbf{B}^{(p)} + \Delta\mathbf{B}^{(p)} \tag{5.12}$$

and the updated residual can be computed as:

$$\mathbf{r}^{(p+1)} = \mathbf{E} - \mathbf{K}(\mathbf{B}^{(p)} + \Delta\mathbf{B}^{(p)}) = \mathbf{r}^{(p)} - \mathbf{K}\Delta\mathbf{B}^{(p)} \tag{5.13}$$

since

$$\mathbf{r}^{(p)} = \mathbf{E} = \mathbf{K}\mathbf{B}^{(p)} \tag{5.14}$$

However, in this case all the $\Delta\mathbf{B}^{(p)}$ are zero except ΔB_i. Therefore,

$$r_j^{(p+1)} = r_j^{(p)} - \frac{K_{ji}}{K_{ii}} * r_i^{(p)}, \forall j \tag{5.15}$$

and expanding $\mathbf{K}$ for the case of constant elements,

$$r_j^{(p+1)} = r_j^{(p)} + \rho_j F_{ji} * r_i^{(p)}, \forall j \tag{5.16}$$

Updating $\mathbf{r}$ thus takes only $O(n)$ steps involving multiplying a scalar by a *column* of the matrix, $\mathbf{K}$.

The final requirement is to compute $\mathbf{r}^{(0)}$ easily at the start of the algorithm. Simply choosing $\mathbf{B}^{(0)}$ to be $\mathbf{0}$ (the zero vector) gives

$$\mathbf{r}^{(0)} = \mathbf{E} - \mathbf{K}\mathbf{B}^{(0)} = \mathbf{E} \tag{5.17}$$

Shooting: A Physical Interpretation of Southwell Iteration

Each step of the Southwell algorithm does two things. It updates one element of the radiosity vector, B_i, and updates *all* of the residuals. If constant bases are used, the physical interpretation of $\mathbf{B}$ is the vector of nodal radiosity values and thus the radiosity of each element. What is the physical interpretation of the residuals $\mathbf{r}$? The residuals are the *unshot* radiosity of each element. In other words, the residual represents the radiosity that has been received as illumination by the elements but has not yet been "reflected" or *shot* back out to contribute further to the illumination of the environment.

It is clear from equation 5.16 that each step takes one residual and adds fractions of it to each of the others. The specific fractions are the i^{th} column of $\mathbf{K}$. Rewriting equation 5.16 using the reciprocity principle,

$$r_j^{(p+1)} = r_j^{(p)} + \rho_j F_{ji} r_i^{(p)} = r_j^{(p)} + \rho_j F_{ij} \frac{A_i}{A_j} r_i^{(p)} \tag{5.18}$$

or rearranging,

$$r_j^{(p+1)} A_j = r_j^{(p)} A_j + \rho_j F_{ij} r_i^{(p)} A_i \text{ for each residual } r_j \tag{5.19}$$

shows that the total *unshot energy* (radiosity $\times$ area), is reduced in each step since both the ρ and the sum of the F_{ij}'s are less than one.

Progressive Refinement

Cohen *et al.* [1] describe a slightly different form of Southwell's algorithm, called *progressive refinement* or *progressive radiosity.* The goal of progressive refinement is to display results after each *step* of an iterative process, in order to provide immediate feedback. The requirement is thus not only to provide an algorithm that converges quickly, but one that makes as much progress at the beginning of the algorithm as possible. In progressive refinement, the flow of the traditional radiosity approach becomes an iterative repetition of the steps of computing form factors, performing steps of the solution and displaying intermediate results.

In this algorithm one shooting step updates all the other elements. Energy is shot from the element that currently has the most unshot radiosity. One shooting step takes $O(n)$ operations and can be viewed as multiplying the scalar B_i by a column of the form factor matrix. Cohen *et al.* show that in many cases only a small fraction of shooting steps is required to approximate a solution closely.

At first glance, Southwell and progressive refinement seem to be quite different. Southwell updates only one entry in $\mathbf{B}$, while progressive refinement updates them all at each step. However, recognising $\mathbf{B}$ in the progressive refinement algorithm to be the sum of $\mathbf{B}$ and $\mathbf{r}$ of the Southwell algorithm, the two algorithms are in fact almost identical.

5.4 Hierarchical Methods

The hierarchical method is an alternative approach to reducing the computational cost of the radiosity algorithm, by attempting to reduce the number of individual form factors that have to be computed. For example, two groups of elements separated widely in space might reasonably have the total interaction between all pairs of individual elements represented by a single number computed once for the entire group. Attaining this goal involves developing a *hierarchical* subdivision of the surfaces and an associated hierarchy of interactions. The hierarchy will provide a framework for deriving interactions between groups of elements, which will result in computing many fewer than $O(n^2)$ interactions. In fact, it will turn out that only $O(n)$ form factors are required to represent the linear operator $\mathbf{K}$ to within a desired error tolerance.

5.4.1 A Physical Example

Imagine a room containing a table on which are placed several small objects. Light reflected from the table top contributes some illumination to the wall. Intuitively, however, the shading of the wall does not depend significantly on the small details of the illumination leaving the table top. If the objects on the table are rearranged so that the shadows on the table are slightly altered, the shading on the wall does not change significantly.

Representing the contribution of the table top to the wall by a single average value will give a similar result to computing individual contributions from many small elements on the table top. This is because ideal diffuse reflection effectively averages the light arriving over an incoming solid angle. If the solid angle is not too great, as is the case when the source is far away relative to its size from a receiver, it is reasonable to average the source radiosity before the integration step. For radiosity, this means replacing several form factors with a single form factor.

On the other hand, when the table top is eventually rendered in an image, its shading must capture all the details of the light and shadows falling on it. These details will be lost of the effect of illumination arriving at the table is averaged over a large area. Thus, the radiosity of a particular surface or group of surfaces will have to be represented at at least two levels of detail: coarsely when the surface is a source, and finely when it is a receiver.

5.4.2 Two-Level Hierarchy

The first hierarchical algorithm for radiosity was developed by Cohen *et al.* [2]. It provides two levels of hierarchy and is based directly on the above-noted distinction between sources and receivers. Surfaces are subdivided coarsely into *patches* to represent the surface when it acts as a source of illumination to other surfaces. The same surfaces, when receiving illumination, are represented by a finer subdivision of the patches into *elements*. Images are rendered using the element mesh. The element mesh can be subdivided adaptively as more information about the illumination of the surface becomes known to achieve an appropriate element size, but the patch subdivision is specified beforehand by the user. The algorithm assumes constant basis functions.

The steps in the algorithm as as follows:

1. Divide the surface into m patches and n smaller elements ($m \ll n$), where each patch is composed exactly of the union of some subset of the elements. The patches will act as the sources or "shooters" and the elements will act as the receivers or "gatherers". The patches are indexed by i or j and the elements by q.

2. Compute the $m \times n$ form factors from each element to each patch. A single entry from element q to patch j would be F_{qj}.

3. Compute the $m \times m$ form factors between pairs of patches directly from the element-to-patch form factors by summing the form factors for the elements q belonging to patch i, weighted by the areas of the element:

$$F_{ij} = \sum_{q \in i} F_{qj} \frac{A_q}{A_i} \tag{5.20}$$

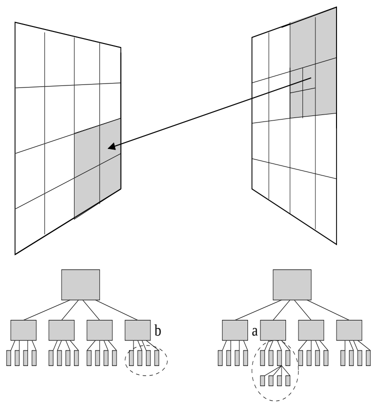

Figure 5.1: Hierarchical quadtrees and interactions.

4. Solve the $m \times m$ system of equations using Gauss-Seidel to get the patch radiosities:
$$B_i = E_i + \rho_i \sum_{j=1}^{m} B_j F_{ij} \tag{5.21}$$

5. Back solve for the element radiosities. This is accomplished by plugging the element-to-patch form factors computed in step 2 and the patch radiosities, b_j, from step 4, into the radiosity equation expressing the element radiosity as the sum of contributions from each patch:
$$B_q = E_q + \rho_q \sum_{j=1}^{m} B_j F_{qj} \tag{5.22}$$

Following the solution, the radiosity function is approximated by the n elements. However, the solution has required computing only $m \times n$ form factors, where m is much less that n.

5.4.3 Multilevel Hierarchy

Hanrahan *et al.* [4] have generalised the notion of subdivision hierarchy to multiple levels, allowing both receivers and sources to be treated at the appropriate level of detail. The recursive subdivision of each surface results in a *quadtree* of nodes, each of which represents some portion of the surface. Energy can be shot from *any node* to any other node at any level in the hierarchy. One such interaction is indicated by the arrow in Figure 5.1, where energy from a group of seven elements (a) is transported to a group of four elements (b). If energy is shot to a node at a level above the leaves, the nodes below will *inherit* the energy received above. This approach leads to fewer total shooting (gathering) operations. The key result will be that only $O(n)$ interactions (form factors) are required to meet a given error tolerance, where n is the number of leaves of the quadtrees (i.e., the number of elements).

Hierarchical Refinement

In contrast to previous algorithms, the multilevel hierarchical algorithm never explicitly builds a matrix. Instead, as the subdivision is performed recursively, *links* are built on-the-fly between nodes of the quadtrees associated with each surface.

A link represents the physical relationship between one set of elements and another set of elements, which determines the potential for energy transport between them. In other words, each link will represent a subset of form factors from the original n by n matrix. A link between a pair of nodes at the lowest level connects two individual elements and thus represents a single entry in the matrix. Links at higher levels represent successively larger groups of form factors.

The aim is to develop a method of predicting whether a subset of form factors will be *coherent* (i.e., similar in magnitude) before actually computing the subset of form factors. Such a prediction method is called an *oracle*.

Solution of the Hierarchical System

Although no explicit matrix is formed, the set of links created defines a linear system that can be solved for the element radiosities. At each quadtree node, the data structure contains two radiosity values, B_s (for shooting) and B_g (for gathering), as well as emission, area and reflectivity. The system is solved iteratively, where each iteration is performed on each top-level surface, and consists of the following two steps:

1. Gather energy over each incoming link.
2. *Push* the received energy down to the children of the node, and *pull* the results back up the quadtrees by area averaging, thus preparing for the next iteration. On the way down the quadtree, the gathered radiosity of each node is simply added to the nodes below. At the leaves, the sum of the gathered radiosity *pushed* down the tree is added to the leaf node's own gathered radiosity and emitted radiosity and the result transferred to the leaf 's shooting radiosity. This quantity is returned up the tree. Since radiosity is units of power/area, the radiosity *pulled* up the tree is the average over a node's children.

The Oracle Function

The Oracle function plays a key role in the efficiency of the overall hierarchical algorithm. The job of the oracle is to estimate the error that will be introduced by *linking* two nodes, rather than creating a series of links at lower levels. In the work reported by Hanrahan *et al.*, the value of the oracle is derived from an estimate of the unoccluded form factor:

$$F_{pq} \approx \frac{\cos\theta}{\pi}\omega_q \tag{5.23}$$

The estimate is computed at the centres of the areas represented by quadtree nodes p and q. The factor ω_q is the solid angle subtended by a disk surrounding the area of node q. In [3] both F_{pq} and F_{qp} are estimated. If either estimate is larger than a given F_ϵ, the element corresponding to the larger is subdivided.

Chapter 6

Meshing and Subdivision

As described earlier, the radiosity function is approximated by

$$B(\mathbf{x}) \approx \hat{B}(\mathbf{x}) = \sum_{i=1}^{n} B_i N_i(\mathbf{x}) \tag{6.1}$$

The accuracy of the approximation is influenced by the size, shape and orientation of the elements, as well as by the order of the basis functions. An optimal mesh uses as few elements as possible to achieve a desired accuracy, which generally means distributing errors in the approximation as evenly as possible among the elements.

To achieve a high quality mesh, it is first necessary to be able to measure the accuracy of the approximation produced by a particular subdivision and choice of basis functions.

6.1 Error Metrics

As stated earlier, the true error in the radiosity approximation can never be calculated, since the value of $B(\mathbf{x})$ is not known. Instead, the error $\epsilon(\mathbf{x})$ must be estimated.

Local Estimate

One approach to estimating the error is to compare the original approximation

$$\hat{B}(\mathbf{x}) = \sum_{i=1}^{n} B_i N_i(\mathbf{x}) \tag{6.2}$$

where the basis functions, $N_i(\mathbf{x})$, are of order k, to a higher order approximation

$$\tilde{B}(\mathbf{x}) = \sum_{i=1}^{n} B_i \tilde{N}_i(\mathbf{x}) \tag{6.3}$$

where the new basis functions, $\tilde{N}_i(\mathbf{x})$, are of order $k+1$ or greater. This estimate of the error is typically evaluated over a local region of the domain, for example, over an element when deciding whether it should be replaced by several smaller elements.

Residual of the Approximate Solution

The residual of the approximate solution (Equation 3.6) provides another approach to characterising $\epsilon(\mathbf{x})$. Clearly, if the residual is zero everywhere, then $\hat{B}(\mathbf{x})$ is an exact solution to the integral equation and the actual error is also zero. Strictly speaking, the residual provides a global, rather than a local error estimate; a high residual in a certain local area does not mean that $\hat{B}(\mathbf{x})$ is a poor approximation over that particular region. However, in practice the residual is often a useful indicator of local error. The residual is expensive to compute because of the integration and is normally evaluated only at selected points.

Error Based on the Behaviour of the Kernel

Error can be characterised by examining the behaviour of the kernel function itself rather than the radiosity approximation. For example, while computing the form factors one might recognise large local variations in the geometric kernel, $G(\mathbf{x}, \mathbf{x}')$, across certain pairs of elements. These gradients might result from surfaces that are in close proximity or from shadow edges or penumbrae.

6.2 Mesh Characteristics and Accuracy

An important goal when creating a mesh is to produce an approximation of the required accuracy as inexpensively as possible. This requires understanding how various characteristics of the mesh affect the accuracy of the approximation. These characteristics can be classified into four main categories: mesh density, element order, element shape and discontinuity representation.

6.2.1 Mesh Density

Mesh density is determined by the size of the elements into which the domain is subdivided. The density of the element mesh determines the number of degrees of freedom available to the approximation. More elements provide more degrees of freedom, which allows the approximation to follow the actual function more closely and increases the degree to which small features like shadows can be resolved. However, uniformly increasing the mesh density in a domain can be very inefficient, as subdivision may occur where the error was already negligible.

6.2.2 Element Order and Continuity

Element order also has a direct effect on accuracy. Higher-order elements interpolate the function using higher-order polynomials and use more information about the behaviour of the function, either by evaluating both values and derivatives at the nodes or by evaluating the function at additional nodes. Thus, higher-order elements can follow the local variations in a function more closely than the same number of lower-order elements. However, higher-order elements are generally more expensive to evaluate. Thus, one approach is to use higher-order elements only where the error is high.

The type of basis functions used also affects the degree of continuity in the approximation at element boundaries. Continuity of value and of lower-order derivatives $(C^0, C^1, C^2, \ldots)$[1] is important because the human visual system is highly sensitive to relative spatial variation in luminance and its derivatives.

Since interpolation within elements is a linear sum of polynomials, the approximation is smooth (C^∞) on element interiors. However, continuity at element boundaries is not guaranteed. For *Lagrange* elements interpolation inside or on the boundary of the element depends only on the function values at the nodes belonging to that element. because interpolation along boundaries uses only nodes on the boundary, linear and higher order Lagrange elements guarantee C^0 continuity at element boundaries. (Constant elements are discontinuous in value at element boundaries).

For radiosity, providing continuity greater than C^0 is motivated primarily by perceptual issues. Thus, elements with this property have been applied mainly to the rendering stage.

6.2.3 Element Shape

Elements should provide well-behaved, easily evaluated basis functions. For this reason, standard element shapes consist of simple geometries, typically triangles and rectangles. These standard shapes can be mixed to subdivide a complicated geometry more efficiently.

If Gouraud shading is to be used for rendering, a number of special problems relating to element shape must be avoided. For example, Gouraud interpolation over a concave element can generate discontinuities in value.

Aspect Ratios

Element shape affects the efficiency with which the mesh samples the radiosity function. To make the most efficient use of a given number of nodes, the nodes should be evenly distributed over the domain of the function assuming that the behaviour of the function is unknown). A distribution of nodes that is denser in one direction than another is inefficient, since if elements are subdivided far enough to make sampling adequate in the sparse direction, sampling in the denser direction will be greater than necessary. A reasonably uniform distribution of nodes can be obtained by requiring elements to have as high an *aspect ratio* as possible.

However, if the behaviour of the function is known, anisotropic sampling may be more efficient that a uniform distribution, since the function changes more slowly in the direction orthogonal to the gradient and thus fewer samples are required along that direction. Elements with a lower aspect ratio may be more efficient in this case, if oriented correctly.

Mesh Grading

A nonuniform, or *graded* mesh may be required to distribute the error evenly in the approximation. In a *well-graded* mesh, element size and shape vary smoothly in the transition between regions of higher and lower density.

[1] The notation C^k indicates that a function is continuous in all derivatives up to and including k. C^0 thus indicates a function that is continuous in *value* (i.e., there are no sudden jumps in value), C^1 that the function is continuous in *slope* (i.e., there are no kinks), and C^∞ that the function is *smooth* (i.e., continuous in all derivatives).

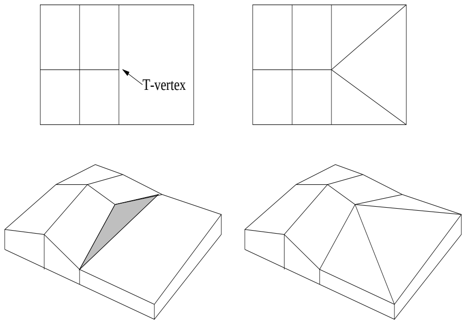

Figure 6.1: Conforming versus nonconforming elements

Mesh Conformance

It is important that adjacent elements *conform* across shared boundaries, as shown in Figure 6.1. Discontinuities in values created by nonconforming elements can cause distinctive image artifacts. Conforming elements share nodes along boundaries, thus ensuring C^0 continuity at the boundaries. In practice, this means that *T-vertices* must be avoided. Conformance is not an issue for constant elements, which are often used during the radiosity solution, but it is critical for rendering, where linear or higher-order elements are normally used.

6.2.4 Discontinuities

If light sources have a constant emission value across the surface, and if changes in visibility are ignored, then the radiosity function $B(\mathbf{x})$ across a receiving surface will be continuous in all derivatives. If changes in visibility (i.e. shadows) are included, the radiosity function can contain discontinuities of any order. Discontinuities in value and in the first and second derivatives are the most important, since these often provide vital visual cues, and hence, the failure to produce such discontinuities accurately can degrade image quality dramatically.

Value Discontinuity

Discontinuities in value of the radiosity function occur where one surface touches another. Incorrectly resolved value discontinuities can lead to light and shadow "leaks". If a boundary falls across the interior of elements, and one side of the boundary is "bright", and the other side is in shadow, interpolation creates a smooth transition and the shadow on one side appears to leak across the boundary.

Derivative Discontinuity

Discontinuities in the first or second derivative occur at penumbra and umbra boundaries, as well as within the penumbra. When mesh elements span these discontinuities, interpolation often produces an inaccurate and irregular shadow boundary. The correct resolution of discontinuities requires that they fall along element boundaries. Thus, discontinuity boundaries must either be determined before meshing, or the mesh must adapt dynamically to place element edges along the discontinuities.

6.3 Automatic Meshing Algorithms

Meshing algorithms can be broadly classified according to whether or not they use information about the behaviour of the function to be approximated. Although obtaining an optimal mesh normally requires such knowledge, in practice some degree of meshing without it is almost always necessary. Meshing in this case generally means subdividing as uniformly as possible.

Meshing techniques that use knowledge of the function can be characterised as either a priori or a posteriori. A priori methods specify all or part of the mesh before the solution is performed. Discontinuity meshing is an a priori method. A posteriori algorithms determine or refine the mesh after the solution has been at least partially completed. An initial approximation is obtained using a uniform or other mesh. The mesh is then refined in regions where the local error is high.

6.3.1 A Posteriori Meshing

A posteriori algorithms can be categorised as follows:

- *r-refinement:* reposition nodes
- *h-refinement:* subdivide existing elements
- *p-refinement:* increase polynomial order of existing elements
- *remeshing:* replace existing mesh with a new mesh

Each of these approaches addresses one or more of the basic mesh characteristics: mesh density, basis function order, and element shape. Radiosity algorithms have so far relied almost exclusively on h-refinement.

6.3.2 Adaptive Subdivision

Adaptive subdivision follows the basic outline of an a posteriori method: a solution is computed on a uniform initial mesh, and then the mesh is refined by subdividing elements that exceed some error tolerance.

Error Estimation for Adaptive Subdivision

Many algorithms subdivide according to a discrete approximation to one or more of the error norms described in section 6.1. For example, Cohen *et al.* [2] compare the radiosities of an element and its neighbours. If these differ in value by more than a user-specified tolerance, the element is subdivided. This heuristic generally produces acceptable results, although it tends to over-subdivide where the gradient is high but constant. It may also fail to identify elements that should be subdivided. If an element contains a local minimum, the nodal values alone do not provide enough information about the behaviour of the function on the element interior.

Vedel and Peuch [13] use the gradient at the nodes as well as the function value. Elements are subdivided if the gradients at the element nodes vary by more than a certain tolerance. This criterion avoids subdividing where the gradient is high but constant. It may also detect a local minimum within an element whose nodal values happen to be similar. A more stringent criterion can be constructed by combining the nodal values and gradient methods.

6.4 Basic Mesh Subdivision Techniques

6.4.1 Mesh Template Methods

Subdivision techniques are broadly classified according to whether or not the mesh topology is predetermined. Algorithms that use a predetermined topology subdivide by mapping a mesh *template* to the geometry. The template is a predefined subdivision into standard elements, such as a rectangular grid. The essential step of the meshing process consists of determining a mapping between the geometry and the template. The most direct way is simply to superimpose the template on the geometry. The geometry is then split along the grid lines. Unfortunately, this approach often results in undesirable elements near boundaries, but this can sometimes be improved by smoothing the mesh. The use of a template is simplified if the template can be mapped to fit the geometry exactly. For example, a rectangular template can be mapped exactly onto any convex quadrilateral. An alternative for complicated geometries is to subdivide the geometry into simple regions, and then map templates to the regions.

6.4.2 Decomposition Methods

The alternative, in which the topology is not predetermined, is to decompose the geometry into elements by locating and connecting nodes according to some procedure. As a trivial example, a polygon might be subdivided by connecting its vertices to form triangles. These methods are generally more flexible than template methods. There are two basic approaches: the nodes and elements are generated together, or the nodes are generated first, and then connected to form edges.

Nodes-Elements-Together Decomposition

The geometry is subdivided by creating nodes and joining them with edges to generate new elements one by one. Subdivision can be performed recursively, first subdividing the entire geometry into initial elements, then splitting those elements again until the desired element size is reached.

Decomposition by Advancing Front

Decomposition proceeds from the boundary inward, splitting off finished elements one at a time. This method allows control over element quality, since quality criteria can be explicitly incorporated into the rules according to which elements are split off from the front. However, difficulties can occur when meshing complicated geometries. Backtracking may be required to undo decisions.

Nodes-First Decomposition.

Decomposition is accomplished by positioning nodes first, and then connecting them to form edges. Because nodes are placed and connected in independent steps, this method is extremely flexible. Nodes can be placed anywhere, with nonuniform density if desired.

Nodes are usually connected by triangulation. Delaunay triangulation is commonly used since it guarantees triangles that are as well shaped as possible.

6.4.3 Discontinuity Meshing

A posteriori refinement is not very effective at reducing error in the neighbourhood of discontinuities. Reducing this error requires either a very high mesh density, which is expensive, or an exact alignment of element edges with the discontinuity, which is difficult to achieve using an a posteriori algorithm. However, it is possible to determine discontinuities a priori. Discontinuities in the radiosity function correspond to transitions in occlusion between source and receiving surfaces. These are purely geometric in nature and can be determined before radiosities are computed.

Value Discontinuities

A value discontinuity occurs when one surface touches another. The discontinuity is caused by the abrupt transition from complete visibility to complete occlusion that is encountered in crossing the line of contact between the surfaces. The locations of these discontinuities can be identified by determining the geometric intersections of all surfaces. Once boundaries have been identified, they can be inserted into the representation of the polygon as specially flagged edges.

First and Second Derivative Discontinuities

These discontinuities result from qualitative transitions in occlusion between surfaces, i.e. shadow boundaries. Imagine viewing a partially occluded source from a point moving along a receiving surface. As the viewpoint moves, more of the source is revealed. The visibility of the source varies according to the orientation of the source and the occluding polygon. Whenever an edge joins or leaves the set of overlapping edges, a new variation is determined and there is a discontinuity in the first or second derivative. This transition is called a *visual event*. There are 2 classes of visual events, *Vertex-Edge* (VE or EV) and *Edge-Edge-Edge* (EEE). A VE event occurs when a vertex of the source crosses an edge of the occluder or conversely when an edge of the source crosses a vertex of the occluder. The event defines a *critical surface*, that is, the *wedge* formed by the vertex and the edge that cause the event. The intersection of this wedge with the scene polygons determines the discontinuity boundaries. EEE events involve transitions caused by overlapping edges of multiple occluders.

Discontinuity Meshing Algorithms

Early algorithms dealt only with events that defined the boundaries of the umbra and penumbra. Nishita and Nakamae [10] determined these boundaries by

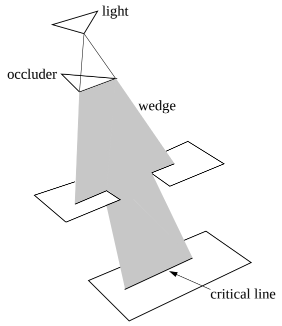

Figure 6.2: Steps in the method of Lichinski for the computation of discontinuity edges

computing shadow volumes formed by an object and each vertex of the light polygon. The umbra volume for an occluder is given by the intersection of the shadow volumes, and the penumbra by the union.

Heckbert [5] and Lichinski *et al.* [8] have independently developed similar algorithms that compute discontinuity boundaries due to VE (and EV) events involving the vertices or edges of source polygons. For a given source polygon, the wedge for each VE event is formed and intersected with the scene polygons. These intersections determine discontinuity edges, which are then inserted into the mesh data structure for the polygon.

Chapter 7

Rendering

Once the radiosity solution has been computed, the remaining step is to render an image. During rendering, the model is projected onto the image plane, the visible surfaces are determined, and a colour is computed for each pixel. The pixel colour is derived from the radiance of the surface location visible at the pixel. The radiance can be determined directly from the approximation of the radiosity function. However, this approximation is often not the best choice for rendering, due primarily to characteristics of the human visual system.

Given an eye location and viewing frustum, an image plane is calculated. For each pixel on the image plane, a ray is shot from the eye through the pixel, until it intersects a surface. The radiance at the pixel can be calculated from the value of the radiosity function at that point. Repeating this process for each pixel produces the image. Unfortunately, an approximation that provides sufficient accuracy during the solution may not be adequate for rendering. One consequence is that the constant elements often used during the radiosity solution cannot be used during rendering, since discontinuities in value are immediately perceived as unrealistic.

Elements providing the desired continuity could be used throughout the radiosity solution, but if the solution is otherwise accurate enough, this unnecessarily increases its cost. Instead, most radiosity implementations derive a new approximation with the desired characteristics just prior to rendering.

7.1 Interpolation Methods for Rendering

When the original approximation $\hat{B}$ is not adequate for rendering, it can be replaced by another approximation using a new set of basis functions and coefficients. The new basis functions are selected for

1. the desired degree of continuity to avoid perceptual artifacts,
2. fidelity to the original approximation,
3. and suitability for fast rendering using hardware Gouraud shading (if appropriate).

If linear elements, which provide a C^0 approximation, are used in the solution they can be applied directly for rendering. If constant elements are used, a linear interpolation must be derived from the constant element solution to produce C^0 shading in the image. A piecewise linear interpolation can be constructed by creating linear elements with nodes at the element vertices. The values at these vertices are then derived from the nodes of the original constant elements.

One approach is to create new nodes at the corners of the existing elements. The radiosity at these nodes can then be obtained by averaging the radiosities of the adjacent elements. Values at nodes on the boundary of the geometry can be extrapolated from interior nodes, although care must be taken not to create negative values. Finally, values in the interior of the elements are linearly (for triangular elements) or bilinearly (for quadrilateral elements) interpolated from the new nodes. This approach tends to smooth the radiosity function, which can help mask artifacts, but it can also remove subtle shading variations.

Another approach is to create a new element subdivision by connecting the existing nodes at the centres of each original element with edges, thus using the same set of nodes during the solution and rendering stages.

Interpolation

Once values have been obtained at element corners, linear or bilinear interpolation can be used to determine the radiosity at each pixel during rendering. This generally involves mapping the world coordinates of the point to be shaded to the parametric coordinates of the element. For triangular elements, barycentric coordinates are a good parameterisation for interpolation, while bilinear interpolation is normally used for quadrilateral elements.

Bilinear interpolation is normally performed in the parametric coordinates of the element, and hence it is independent of the projection of the element onto the image plane. Some algorithms prefer to perform this interpolation in image space as it is less expensive. In Gouraud shading, colours defined at polygon vertices are interpolated along the polygon edges in the screen projection. At each scanline, these values are then interpolated horizontally between two edges intersecting the scanline. This can be easily accelerated using specialised graphics hardware. Unfortunately, Gouraud shading has limitations that can severely degrade image quality if not taken into account. Since the perspective transformation from world space to image space is nonaffine, linear interpolation in image space is not equivalent to interpolation in the parametric space of the element. Furthermore, Gouraud shading does not uniquely determine the interpolated value on the polygon interior for a given location. It is dependent on the orientation of the polygon on the image plane, and the interpolated shading may change visibly as the view changes. Gouraud shading also handles concave polygons and T-vertices poorly. Most of these problems can be overcome by using triangular elements.

C^1 Interpolation

Linear interpolation is a huge improvement over constant elements, but it also has some problems. It is discontinuous in the first derivative at element boundaries, with the result that Mach bands often appear along mesh boundaries. A finer mesh density, or higher order interpolation can reduce or eliminate Mach bands. However, these do not necessarily provide higher than C^0 continuity at boundaries. Elements that provide a C^1 approximation will eliminate Mach bands in most cases without the need for a finer subdivision.

The problem of interpolating the radiosity function during rendering is analogous to the problem of constructing a surface in three-dimensional space for modelling. A surface that is everywhere C^1 can be constructed using quadratic or higher order Bezier patches. A Bezier patch describes a surface that passes through control points on the corners of the patch. Other control points on the boundary and interior determine the tangent to the surface at the patch corners and the shape of the surface on the interior. Surfaces with the desired continuity can also be constructed using more general B-splines. Salesin *et.al.* [11]have applied the Clough-Tocher element to rendering for radiosity. This element is constructed from three cubic triangular Bezier patches and provides a C^1 interpolant at boundaries.

7.2 Two-Pass Methods

Two-pass methods evaluate the rendering equation directly at image resolution, using the radiosity solution to provide the approximation of secondary illumination. They are particularly good in handling non-Lambertian reflection.

Monte Carlo Ray Tracing per Pixel

During the rendering pass the rendering equation is evaluated using Monte Carlo path tracing, with paths limited to one bounce. Multiple rays are traced from the eye through each pixel into the scene. From the intersection point of each ray, a single ray is traced to a random point on a light source and a second ray is traced in a random direction chosen with a probability proportional to the cosine of the angle measured from the normal.

The radiance at the point intersected by the secondary ray is obtained from the radiosity solution. If the secondary ray hits a light source, it must be counted as making a contribution of zero, since the sources are already accounted for by the ray to the light. Meshing algorithms can be greatly simplified when the radiosity solution is no longer required to capture shading features like shadow in great detail.

Form Factors per Pixel

The radiosity at each pixel is computed by evaluating a form factor to every element in the radiosity mesh. This is less efficient than the Monte Carlo approach, since it computes a contribution from every element, regardless of its contribution. On the other hand, it has the advantage of never missing an important source of illumination. Although expensive, it produces extremely high quality images.

Although meshing requirements for the radiosity solution are less rigorous when the final bounce is evaluated at image resolution, sampling in image space can exacerbate certain types of errors. For example, the illumination provided by constant elements can cause shading artifacts for nearby receivers. Monte Carlo integration of the form factor may be preferable in this case since it will produce noise that is less objectionable than coherent errors.

7.3 Incorporating Surface Detail

It has been assumed so far that surface properties such as reflectivity are constant over each surface. This has allowed the reflectivity term to be moved outside the integral during the computation of the linear integral operator. Surface detail is often simulated by using a texture map or a bump map.

Texture Mapping

When surface reflectivity is specified by a texture map, the resulting shading typically contains very high spatial frequencies. These are difficult to represent in a finite element approximation. As an alternative, during the radiosity solution, the reflectivity ρ of a textured surface can be approximated by the average texture reflectivity, $\bar{\rho}$, over the support of a basis function. This is a reasonable assumption in most cases, since the high frequency variations in radiance are averaged together during integration.[1]

When rendering with two-pass methods, textures can be handled easily. If using interpolation, the radiosity at any pixel $\mathbf{x}_p$ is:

$$\hat{B}(\mathbf{x}_p, \rho(\mathbf{x}_p)) = \hat{B}(\mathbf{x}_p, \bar{\rho}) \frac{\rho(\mathbf{x}_{p)}}{\bar{\rho}} \tag{7.1}$$

The radiosity is simply multiplied by the texture mapped reflectivity for the pixel over the average reflectivity.

Bump Mapping

Two-pass methods handle bump maps in a straightforward manner by using the perturbed normal at each pixel during the view pass. Bump mapping when interpolation is being used is also possible. Instead of computing a single radiosity value at each node, the variation of radiosity with surface orientation at the node is sampled by computing independent radiosity values for several surface normals distributed over the hemisphere. During rendering, the perturbed normal at a particular pixel is determined from the bump map. For each of the element nodes, the radiosity corresponding to that orientation is interpolated form the hemisphere sampling, and the radiosity at the pixel is bilinearly interpolated from these nodal values.

[1] This assumption can be violated when a texture map contains large scale variations, e.g, a black and white checkerboard pattern.

Chapter 8

Extensions and Future Research

Earlier, the radiosity equation was derived from the general rendering equation by restricting the problem in various ways. Diffraction, polarisation, scattering by the transport media, and directional reflection are all omitted. Although computationally convenient, some of these assumptions are too restrictive for general-purpose image synthesis.

8.1 Nondiffuse Light Sources

The simplest extension to the basic radiosity method is to allow light sources to emit with a non-Lambertian distribution. The simplicity of this derives form the fact that lights are normally predefined. They are typically treated as emitters only. Different light sources and their form factors are:

- Isotropic Point light: light emanates from a point with equal radiant intensity in all directions. The flux density falls off according to $1/r^2$.

$$F_{ij} = \int_{A_j} \frac{1}{4\pi} \frac{\cos\theta_j}{r^2} V(\mathbf{x}_j, \mathbf{x}_i) dA_j \tag{8.1}$$

- Parallel light: the light source is at infinity in a particular direction. Thus, the flux density is constant. Direct sunlight can be approximated using a parallel light.

$$F_{ij} = \int_{A_j} \cos\theta_j V(-\vec{\omega}, \mathbf{x}_j) dA_j \tag{8.2}$$

- Spot light: light emanates from a point with a variable intensity that falls off from a maximum as the direction deviates from a given axis.

$$F_{ij} = \int_{A_j} \frac{\cos^n\theta_i \cos\theta_j}{r^2} V(\mathbf{x}_i, \mathbf{x}_j) dA_j \tag{8.3}$$

- General luminaires: light emanates from a point or area according to a general distribution function defined either by a *goniometric diagram*, often available from lighting manufacturers, or by an analytic functional form.

- Sky light: light emanates from a hemisphere representing the sky, possibly accounting for weather conditions and solar position (but no the sun itself).

8.2 Directional Reflection

In radiosity, the BRDF is limited to Lambertian diffuse reflection. Since the non-Lambertian components of reflection play an important part in everyday visual experience, radiosity images are often not completely realistic. The absence of specular highlights not only reduces realism, but also removes important visual cues to shape and curvature. The restriction to matte finishes is also a serious limitation for design applications where the evaluation of surface appearance is important.

If specular reflection is to be included, a major overhaul of the form factor calculation engine is needed. This generally involves calculating extra form factors, which take indirect reflection into account. It is similar in many ways to bidirectional ray tracing, but at the moment, it is far too expensive to used in the real world.

8.3 Future Research

Current research in radiosity is being concentrated into three main areas:

- Meshing, particularly better discontinuity meshing algorithms, and alternative shadow boundary detection.
- Stochastic ray tracing.
- Parallelisation

Chapter 9

Radiosity In The Real World

In general, radiosity is most useful in applications where ideal diffuse interreflection is important and the geometry is static. The view-independent approach is particularly valuable when the exploration of a three-dimensional model is required, rather than the production of a single static image.

9.1 Architectural Design

Architectural design is in many ways the ideal application for the radiosity method. Architects must communicate a design to a client, who may have difficulty visualising the final building from blueprints and drawings. As a result, architects normally depend on artist's impressions of the building, or on complex scale models. These methods are extremely inflexible, as they only allow fixed views in the case of paintings, or they fail to convey the proper feeling of interior space, as in the case of scale models. They are time consuming to produce, and do not allow experimentation with interior colouring and lighting.

Radiosity is particularly well suited to architectural design. Many interior surfaces (e.g., upholstery, textiles, matte paints) are reasonably well approximated by ideal diffuse reflection, and diffuse interreflection makes an important contribution to the illumination of interiors. Architectural models are usually static. The radiosity solution, when rendered, allows interactive walkthroughs of the shaded model, giving the designer and the client the opportunity to explore the interior space interactively.

Radiosity simulations do have some drawbacks though. Many common surface materials and finishes cannot be modelled accurately (e.g., metals, polishes, glass). Two-pass methods and other approaches that incorporate specular reflection are important.

The cost and performance of radiosity is another issue. Architectural models tend to be very large; models in excess of 100,000 polygons are not uncommon. The $O(n^2)$ computation and storage cost makes current radiosity implementations impractical for problems of this size.

9.2 Commercial Products

- Bentley System's Incorporated *MicroStation TriForma* engineering and architectural visualisation tools.
- CAD Master's *Accurender 3.* A radiosity and raytracing plugin for AutoCAD 14
- Cagliari Corporation's *TrueSpace4.* A hybrid radiosity/raytracing 3D authoring tool.
- LightWorks Design *LightWorks.* A plug-in renderer for QuickDraw 3D on the Macintosh. Uses ray-tracing and radiosity.
- LightScape Technologies *LightScape.* A radiosity engine for architectural visualisation. Also uses ray-tracing for adding specular reflections and highlights.
- Sony Tektronix's *STmage.* A 3D modeller. Uses radiosity for lighting calculations, and ray-tracing for rendering.

Bibliography

[1] Cohen, M., Chen, S. E., Wallace, J. R., and Greenberg, D.P. A progressive refinement approach to fast radiosity image generation. *Computer graphics (SIGGRAPH '88 Proceedings)* **22**:4 (Aug. 1988), pp. 75-84.

[2] Cohen, M., Greenberg, D. P., Immel, D. S. and Brock, P. J. An efficient radiosity approach for realistic image synthesis. *IEEE Computer graphics and Applications* **6**:3 (Mar. 1986), pp. 26-35.

[3] Cohen, M. and Wallace, J. *Radiosity and Realistic Image Synthesis.* Academic Press, 1993.

[4] Hanrahan, P., Salzman, D., and Aupperle, L. A rapid hierarchical radiosity algorithm. *Computer Graphics (SIGGRAPH '91 Proceedings)* **25**:4 (July 1991), pp. 197-206.

[5] Heckbert, P. Discontinuity meshing for radiosity. In *Third Eurographics Workshop on Rendering* (Bristol, UK, May 1992), pp. 203-226.

[6] Heckbert, P.S., and Winget, J.M. Finite element methods for global illumination. Tech. Rep. UCP/CSD 91/643, computer Science Division (EECS), University of California, Berkeley, July 1991.

[7] Kayija, J. T. The rendering equation. *Computer Graphics (SIGGRAPH '96 Proceedings)* **20**:4 (Aug. 1986), pp 143-150.

[8] Lischinski, D., Tampieri, F., and Greenberg, D. P. Discontinuity meshing for accurate radiosity. *IEEE Computer Graphics and Applications* **12**:6 (Nov. 1992), pp. 25-39.

[9] Malley, T.J. A shading method for computer generated images. Master's thesis, Department of Computer Science, University of Utah, June 1988.

[10] Nishita, T. and Nakamae, E. Continuous tone representation of three-dimensional objects taking account of shadows and iterreflection. *Computer Graphics (SIGGRAPH '85 Proceedings)* **19**:3 (July 1985), pp 23-30.

[11] Salesin, D., Lischinski, D., and DeRose, T. Reconstructing illumination functions with selected discontinuities. In *Third Eurographics Workshoop on Rendering* (Bristol, UK, May 1992), pp 99-112.

[12] Sillion, F. and Puech, C. A general two-pass method integrating specular and diffuse reflection. *Computer Graphics (SIGGRAPH '89 Proceedings)* **23**:3 (July 1989), pp 335-344.

[13] Vedel, C., and Peuch, C. A testbed for adaptive subdivision in progressive radiosity. In *Second Eurographics Workshop on Rendering* (Barcelona, Spain, May 1991).

[14] Warnock, J. A hidden-surface algorithm for computer generated half-tone pictures. Technical Report TR 4-15, Dept. of computer Science, University of Utah, June 1969.