Chapter 10

RADIOSITY METHOD

The radiosity method is based on the numerical solution of the shading equation by the **finite element method**. It subdivides the surfaces into small elemental surface patches. Supposing these patches are small, their intensity distribution over the surface can be approximated by a constant value which depends on the surface and the direction of the emission. We can get rid of this directional dependency if only diffuse surfaces are allowed, since diffuse surfaces generate the same intensity in all directions. This is exactly the initial assumption of the simplest radiosity model, so we are also going to consider this limited case first. Let the energy leaving a unit area of surface i in a unit time in all directions be B_i, and assume that the light density is homogeneous over the surface. This light density plays a crucial role in this model and is also called the **radiosity** of surface i.

The dependence of the intensity on B_i can be expressed by the following argument:

1. Consider a differential dA element of surface A. The total energy leaving the surface dA in unit time is $B \cdot dA$, while the flux in the solid angle $d\omega$ is $d\Phi = I \cdot dA \cdot \cos\phi \cdot d\omega$ if ϕ is the angle between the surface normal and the direction concerned.

2. Expressing the total energy as the integration of the energy contributions over the surface in all directions and assuming diffuse reflection

only, we get:

$$B = \frac{1}{dA} \cdot \int^{2\pi} \frac{d\Phi}{d\omega}\, d\omega = \int^{2\pi} I \cdot \cos\phi \; d\omega = I \cdot \int_{\theta=0}^{2\pi} \int_{\phi=0}^{\pi/2} \cos\phi \sin\phi \; d\theta d\phi = I \cdot \pi \tag{10.1}$$

since $d\omega = \sin\phi \; d\phi d\theta$.

Consider the energy transfer of a single surface on a given wavelength. The total energy leaving the surface $(B_i \cdot dA_i)$ can be divided into its own emission and the diffuse reflection of the radiance coming from other surfaces (figure 10.1).

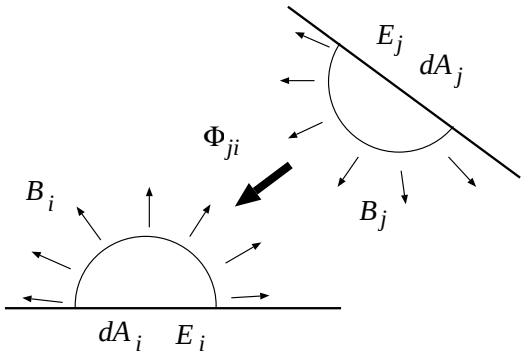

Figure 10.1: Calculation of the radiosity

The emission term is $E_i \cdot dA_i$ if E_i is the emission density which is also assumed to be constant on the surface.

The diffuse reflection is the multiplication of the diffuse coefficient ϱ_i and that part of the energy of other surfaces which actually reaches surface i. Let F_{ji} be a factor, called the **form factor**, which determines that fraction of the total energy leaving surface j which actually reaches surface i.

Considering all the surfaces, their contributions should be integrated, which leads to the following formula of the radiosity of surface i:

$$B_i \cdot dA_i = E_i \cdot dA_i + \varrho_i \cdot \int B_j \cdot F_{ji} \cdot dA_j. \tag{10.2}$$

Before analyzing this formula any further, some time will be devoted to the meaning and the properties of the form factors.

The fundamental law of photometry (equation 3.15) expresses the energy transfer between two differential surfaces if they are visible from one another. Replacing the intensity by the radiosity using equation 10.1, we get:

$$d\Phi = I \cdot \frac{dA_i \cdot \cos\phi_i \cdot dA_j \cdot \cos\phi_j}{r^2} = B_j \cdot \frac{dA_i \cdot \cos\phi_i \cdot dA_j \cdot \cos\phi_j}{\pi \cdot r^2}. \quad (10.3)$$

If dA_i is not visible from dA_j, that is, another surface is obscuring it from dA_j or it is visible only from the "inner side" of the surface, the energy flux is obviously zero. These two cases can be handled similarly if an indicator variable H_{ij} is introduced:

$$H_{ij} = \begin{cases} 1 \text{ if } dA_i \text{ is visible from } dA_j \\ \\ 0 \text{ otherwise} \end{cases} \quad (10.4)$$

Since our goal is to calculate the energy transferred from one finite surface (ΔA_j) to another (ΔA_i) in unit time, both surfaces are divided into infinitesimal elements and their energy transfer is summed or integrated, thus:

$$\Delta\Phi_{ji} = \int_{\Delta A_i} \int_{\Delta A_j} B_j \cdot H_{ij} \cdot \frac{dA_i \cdot \cos\phi_i \cdot dA_j \cdot \cos\phi_j}{\pi \cdot r^2}. \quad (10.5)$$

By definition, the form factor F_{ji} is a fraction of this energy and the total energy leaving surface j ($B_j \cdot \Delta A_j$):

$$F_{ji} = \frac{1}{\Delta A_j} \cdot \int_{\Delta A_i} \int_{\Delta A_j} H_{ij} \cdot \frac{dA_i \cdot \cos\phi_i \cdot dA_j \cdot \cos\phi_j}{\pi \cdot r^2}. \quad (10.6)$$

It is important to note that the expression of $F_{ji} \cdot \Delta A_j$ is symmetrical with the exchange of i and j, which is known as the **reciprocity relationship**:

$$F_{ji} \cdot \Delta A_j = F_{ij} \cdot \Delta A_i. \quad (10.7)$$

We can now return to the basic radiosity equation. Taking advantage of the homogeneous property of the surface patches, the integral can be replaced by a finite sum:

$$B_i \cdot \Delta A_i = E_i \cdot \Delta A_i + \varrho_i \cdot \sum_j B_j \cdot F_{ji} \cdot \Delta A_j. \quad (10.8)$$

Applying the reciprocity relationship, the term $F_{ji} \cdot \Delta A_j$ can be replaced by $F_{ij} \cdot \Delta A_i$:

$$B_i \cdot \Delta A_i = E_i \cdot \Delta A_i + \varrho_i \cdot \sum_j B_j \cdot F_{ij} \cdot \Delta A_i. \tag{10.9}$$

Dividing by the area of surface i, we get:

$$B_i = E_i + \varrho_i \cdot \sum_j B_j \cdot F_{ij}. \tag{10.10}$$

This equation can be written for all surfaces, yielding a linear equation where the unknown components are the surface radiosities (B_i):

$$\begin{bmatrix} 1-\varrho_1 F_{11} & -\varrho_1 F_{12} & \dots & -\varrho_1 F_{1N} \\ -\varrho_2 F_{21} & 1-\varrho_2 F_{22} & \dots & -\varrho_2 F_{2N} \\ \cdot & & & \\ \cdot & & & \\ \cdot & & & \\ -\varrho_N F_{N1} & -\varrho_N F_{N2} & \dots & 1-\varrho_N F_{NN} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \cdot \\ \cdot \\ \cdot \\ B_N \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \cdot \\ \cdot \\ \cdot \\ E_N \end{bmatrix} \tag{10.11}$$

or in matrix form, having introduced matrix $\mathbf{R}_{ij} = \varrho_i \cdot F_{ij}$:

$$(\mathbf{1} - \mathbf{R}) \cdot \mathbf{B} = \mathbf{E} \tag{10.12}$$

($\mathbf{1}$ stands for the unit matrix).

The meaning of F_{ii} is the fraction of the energy reaching the very same surface. Since in practical applications the elemental surface patches are planar polygons, F_{ii} is 0.

Both the number of unknown variables and the number of equations are equal to the number of surfaces (N). The solution of this linear equation is, at least theoretically, straightforward (we shall consider its numerical aspects and difficulties later). The calculated B_i radiosities represent the light density of the surface on a given wavelength. Recalling Grassman's laws, to generate color pictures at least three independent wavelengths should be selected (say red, green and blue), and the color information will come from the results of the three different calculations.

Thus, to sum up, the basic steps of the **radiosity method** are these:

1. F_{ij} form factor calculation.

2. Describe the light emission (E_i) on the representative wavelengths, or in the simplified case on the wavelength of red, green and blue colors. Solve the linear equation for each representative wavelength, yielding $B_i^{\lambda_1}$, $B_i^{\lambda_2}$ ... $B_i^{\lambda_n}$.

3. Generate the picture taking into account the camera parameters by any known hidden surface algorithm. If it turns out that surface i is visible in a pixel, the color of the pixel will be proportional to the calculated radiosity, since the intensity of a diffuse surface is proportional to its radiosity (equation 10.1) and is independent of the direction of the camera.

Constant color of surfaces results in the annoying effect of faceted objects, since the eye psychologically accentuates the discontinuities of the color distribution. To create the appearance of smooth surfaces, the tricks of Gouraud shading can be applied to replace the jumps of color by linear changes. In contrast to Gouraud shading as used in incremental methods, in this case vertex colors are not available to form a set of knot points for interpolation. These vertex colors, however, can be approximated by averaging the colors of adjacent polygons (see figure 10.2).

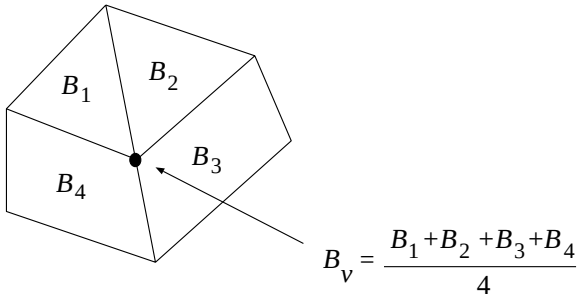

Figure 10.2: Color interpolation for images created by the radiosity method

Note that the first two steps of the radiosity method are independent of the actual view, and the form factor calculation depends only on the

geometry of the surface elements. In camera animation, or when the scene is viewed from different perspectives, only the third step has to be repeated; the computationally expensive form factor calculation and the solution of the linear equation should be carried out only once for a whole sequence. In addition to this, the same form factor matrix can be used for sequences, when the lightsources have time varying characteristics.

10.1 Form factor calculation

The most critical issue in the radiosity method is efficient form factor calculation, and thus it is not surprising that considerable research effort has gone into various algorithms to evaluate or approximate the formula which defines the form factors:

$$F_{ij} = \frac{1}{\Delta A_i} \cdot \int_{\Delta A_i} \int_{\Delta A_j} H_{ij} \cdot \frac{dA_i \cdot \cos\phi_i \cdot dA_j \cdot \cos\phi_j}{\pi \cdot r^2}. \tag{10.13}$$

As in the solution of the shading problem, the different solutions represent different compromises between the conflicting objectives of high calculation speed, accuracy and algorithmic simplicity.

In our survey the various approaches are considered in order of increasing algorithmic complexity, which, interestingly, does not follow the chronological sequence of their publication.

10.1.1 Randomized form factor calculation

The randomized approach is based on the recognition that the formula defining the form factors can be taken to represent the probability of a quite simple event if the underlying probability distributions are defined properly.

An appropriate such event would be a surface j being hit by a particle leaving surface i. Let us denote the event that a particle leaves surface dA_i by $PLS(dA_i)$. Expressing the probability of the "hit" of surface j by the total probability theorem we get:

$$\Pr\{\text{hit } \Delta A_j\} = \int_{\Delta A_i} \Pr\{\text{hit } \Delta A_j \mid PLS(dA_i)\} \cdot \Pr\{PLS(dA_i)\}. \tag{10.14}$$

The hitting of surface j can be broken down into the separate events of hitting the various differential elements dA_j composing ΔA_j. Since hitting of dA_k and hitting of dA_l are exclusive events if $dA_k \neq dA_l$:

$$\Pr\{\text{hit } \Delta A_j \mid PLS(dA_i)\} = \int_{\Delta A_j} \Pr\{\text{hit } dA_j \mid PLS(dA_i)\}. \tag{10.15}$$

Now the probability distributions involved in the equations are defined:

1. Assume the origin of the particle to be selected randomly by uniform distribution:
$$\Pr\{PLS(dA_i)\} = \frac{1}{\Delta A_i} \cdot dA_i. \tag{10.16}$$

2. Let the direction in which the particle leaves the surface be selected by a distribution proportional to the cosine of the angle between the direction and the surface normal:
$$\Pr\{\text{particle leaves in solid angle } d\omega\} = \frac{\cos\phi_i \cdot d\omega}{\pi}. \tag{10.17}$$

The denominator π guarantees that the integration of the probability over the whole hemisphere yields 1, hence it deserves the name of probability density function. Since the solid angle of dA_j from dA_i is $dA_j \cdot \cos\phi_j / r^2$ where r is the distance between dA_i and dA_j, and ϕ_j is the angle of the surface normal of dA_j and the direction of dA_i, the probability of equation 10.15 is:

$$\Pr\{\text{hit } dA_j \mid PLS(dA_i)\} =$$

$$\Pr\{dA_j \text{ is not hidden from } dA_i \wedge \text{particle leaves in the solid angle of } dA_j\}$$

$$= \frac{H_{ij} \cdot dA_j \cdot \cos\phi_j \cdot \cos\phi_i}{r^2 \cdot \pi} \tag{10.18}$$

where H_{ij} is the indicator function of the event "dA_j is visible from dA_i".

Substituting these into the original probability formula:

$$\Pr\{\text{hit}\} = \frac{1}{\Delta A_i} \cdot \int_{\Delta A_i} \int_{\Delta A_j} H_{ij} \cdot \frac{dA_i \cdot \cos\phi_i \cdot dA_j \cdot \cos\phi_j}{\pi \cdot r^2}. \tag{10.19}$$

This is exactly the same as the formula for form factor F_{ij}. This probability, however, can be estimated by random simulation. Let us generate n

particles randomly using uniform distribution on the surface i to select the origin, and a cosine density function to determine the direction. The origin and the direction define a ray which may intersect other surfaces. That surface will be hit whose intersection point is the closest to the surface from which the particle comes. If shooting n rays randomly surface j has been hit k_j times, then the probability or the form factor can be estimated by the relative frequency:

$$F_{ij} \approx \frac{k_j}{n}. \tag{10.20}$$

Two problems have been left unsolved:

- How can we select n to minimize the calculations but to sustain a given level of accuracy?
- How can we generate uniform distribution on a surface and cosine density function in the direction?

Addressing the problem of the determination of the necessary number of attempts, we can use the laws of large numbers.

The inequality of Bernstein and Chebyshev [Rén81] states that if the absolute value of the difference of the event frequency and the probability is expected not to exceed ϵ with probability δ, then the minimum number of attempts (n) is:

$$n \geq \frac{9 \log 2/\delta}{8\epsilon^2}. \tag{10.21}$$

The generation of random distributions can rely on random numbers of uniform distribution in [0..1] produced by the pseudo-random algorithm of programming language libraries. Let the probability distribution function of the desired distribution be $P(x)$. A random variable x which has $P(x)$ probability distribution can be generated by transforming the random variable r that is uniformly distributed in [0..1] applying the following transformation:

$$x = P^{-1}(r). \tag{10.22}$$

10.1.2 Analytic and geometric methods

The following algorithms focus first on the inner section of the double integration, then estimate the outer integration. The inner integration is given some geometric interpretation which is going to be the base of the calculation. This inner integration has the following form:

$$d_i F_{ij} = \int_{\Delta A_j} H_{ij} \cdot \frac{\cos \phi_i \cdot \cos \phi_j}{\pi \cdot r^2} \, dA_j. \tag{10.23}$$

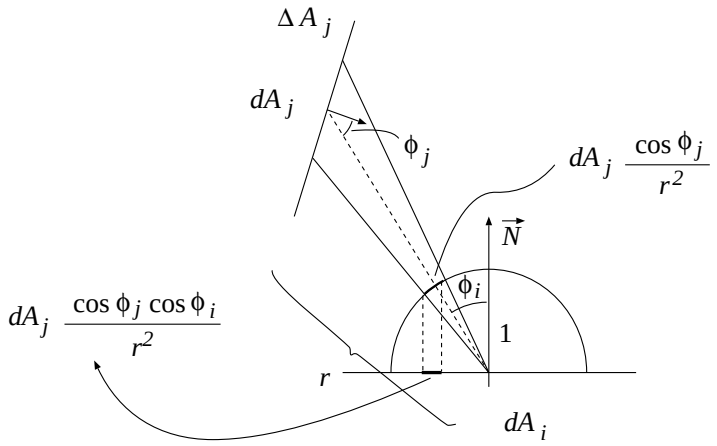

Figure 10.3: Geometric interpretation of hemisphere form factor algorithm

Nusselt [SH81] has realized that this formula can be interpreted as projecting the visible parts of ΔA_j onto the unit hemisphere centered above dA_i, then projecting the result orthographically onto the base circle of this hemisphere in the plane of dA_i (see figure 10.3), and finally calculating the ratio of the doubly projected area and the area of the unit circle (π). Due to the central role of the unit hemisphere, this method is called the **hemisphere algorithm**.

Later Cohen and Greenberg [CG85] have shown that the projection calculation can be simplified, and more importantly, supported by image synthesis hardware, if the hemisphere is replaced by a half cube. Their method is called the **hemicube algorithm**.

Beran-Koehn and Pavicic have demonstrated in their recent publication [BKP91] that the necessary calculations can be significantly decreased if a **cubic tetrahedron** is used instead of the hemicube.

Having calculated the inner section of the integral, the outer part must be evaluated. The simplest way is to suppose that it is nearly constant on ΔA_i, so the outer integral is estimated as the multiplication of the inner integral at the middle of ΔA_i and the area of this surface element:

$$F_{ij} = \frac{1}{\Delta A_i} \int\limits_{\Delta A_i} d_i F_{ij} \; dA_i \approx d_i F_{ij} = \int\limits_{\Delta A_j} H_{ij} \cdot \frac{\cos \phi_i \cdot \cos \phi_j}{\pi \cdot r^2} \; dA_j. \quad (10.24)$$

More accurate computations require the evaluation of the inner integral in several points on ΔA_i and some sort of numerical integration technique should be used for the integral calculation.

10.1.3 Analytic form factor computation

The inner section of the form factor integral, or as it is called the form factor between a finite and differential area, can be written as a surface integral in a vector space, denoting the vector between dA_i and dA_j by $\vec{r}$, the unit normal to dA_i by $\vec{n}_i$, and the surface element vector $\vec{n}_j \cdot dA_j$ by $d\vec{A}_j$:

$$d_i F_{ij} = \int\limits_{\Delta A_j} H_{ij} \cdot \frac{\cos \phi_i \cdot \cos \phi_j}{\pi \cdot r^2} \; dA_j = - \int\limits_{\Delta A_j} H_{ij} \cdot \frac{(\vec{n}_i \cdot \vec{r})}{\pi \cdot |\vec{r}|^4} \cdot \vec{r} \; d\vec{A}_j = \int\limits_{\Delta A_j} \vec{w} \; d\vec{A}_j. \quad (10.25)$$

If we could find a vector field $\vec{v}$, such that $rot \; \vec{v} = \vec{w}$, the area integral could be transformed into the contour integral $\int \vec{v} d\vec{l}$ by Stoke's theorem. This idea has been followed by Hottel and Sarofin [HS67], and they were successful in providing a formula for the case when there are no occlusions, or the visibility term H_{ij} is everywhere 1:

$$d_i F_{ij} = \frac{1}{2\pi} \sum_{l=0}^{L-1} \frac{\text{angle}(\vec{R}_l, \vec{R}_{l \oplus 1})}{|\vec{R}_l \times \vec{R}_{l \oplus 1}|} (\vec{R}_l \times \vec{R}_{l \oplus 1}) \cdot \vec{n}_i \quad (10.26)$$

where

1. $\text{angle}(\vec{a}, \vec{b})$ is the signed angle between two vectors. The sign is positive if $\vec{b}$ is rotated clockwise from $\vec{a}$ looking at them in the opposite direction to $\vec{n}_i$,

2. $\oplus$ represents addition modulo L. It is a circular next operator for vertices,

3. L is the number of vertices of surface element j,

4. $\vec{R}_l$ is the vector from the differential surface i to the lth vertex of the surface element j.

We do not aim to go into the details of the original derivation of this formula based on the theory of vector fields, because it can also be proven relying on geometric considerations of the hemispherical projection.

10.1.4 Hemisphere algorithm

First of all the result of Nusselt is proven using figure 10.3, which shows that the inner form factor integral can be calculated by a double projection of ΔA_j, first onto the unit hemisphere centered above dA_i, then to the base circle of this hemisphere in the plane of dA_i, and finally by calculating the ratio of the double projected area and the area of the unit circle (π). By geometric arguments, or by the definition of solid angles, the projected area of a differential area dA_j on the surface of the hemisphere is $dA_j \cdot \cos\phi_j / r^2$. This area is orthographically projected onto the plane of dA_i, multiplying the area by factor $\cos\phi_i$. The ratio of the double projected area and the area of the base circle is:

$$\frac{\cos\phi_i \cdot \cos\phi_j}{\pi \cdot r^2} \cdot dA_j. \tag{10.27}$$

Since the double projection is a one-to-one mapping, if surface ΔA_j is above the plane of A_i, the portion, taking the whole ΔA_j surface into account, is:

$$\int_{\Delta A_j} H_{ij} \cdot \frac{\cos\phi_i \cdot \cos\phi_j}{\pi \cdot r^2}\, dA_j = d_i F_{ij}. \tag{10.28}$$

This is exactly the formula of an inner form factor integral.

Now we turn to the problem of the **hemispherical projection** of a planar polygon. To simplify the problem, consider only one edge line of the polygon first, and two vertices, $\vec{R}_l$ and $\vec{R}_{l\oplus 1}$, on it (figure 10.4). The hemispherical projection of this line is a half great circle. Since the radius

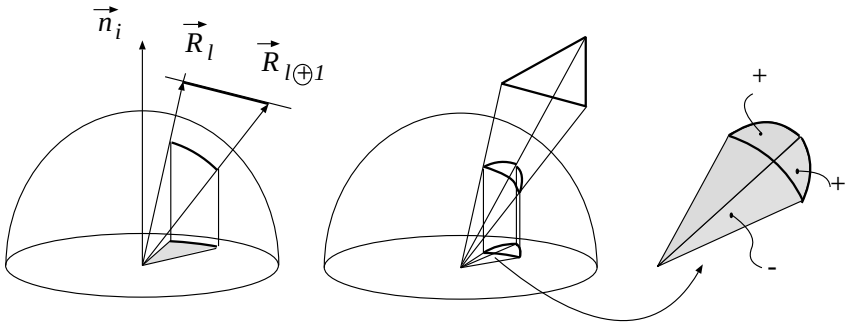

Figure 10.4: Hemispherical projection of a planar polygon

of this great circle is 1, the area of the sector formed by the projections of $\vec{R}_l$ and $\vec{R}_{l\oplus 1}$ and the center of the hemisphere is simply half the angle of $\vec{R}_l$ and $\vec{R}_{l\oplus 1}$. Projecting this sector orthographically onto the equatorial plane, an ellipse sector is generated, having the area of the great circle sector multiplied by the cosine of the angle of the surface normal $\vec{n}_i$ and the normal of the segment $(\vec{R}_l \times \vec{R}_{l\oplus 1})$.

The area of the doubly projected polygon can be obtained by adding and subtracting the areas of the ellipse sectors of the different edges, as is demonstrated in figure 10.4, depending on whether the projections of vectors $\vec{R}_l$ and $\vec{R}_{l\oplus 1}$ follow each other clockwise. This sign value can also be represented by a signed angle of the two vectors, expressing the area of the double projected polygon as a summation:

$$\sum_{l=0}^{L-1} \frac{1}{2} \cdot \text{angle}(\vec{R}_l, \vec{R}_{l\oplus 1}) \frac{(\vec{R}_l \times \vec{R}_{l\oplus 1})}{|\vec{R}_l \times \vec{R}_{l\oplus 1}|} \cdot \vec{n}_i. \tag{10.29}$$

Having divided this by π to calculate the ratio of the area of the double projected polygon and the area of the equatorial circle, equation 10.26 can be generated.

These methods have supposed that surface ΔA_j is above the plane of dA_i and is totally visible. Surfaces below the equatorial plane do not pose any problems, since we can get rid of them by the application of a clipping algorithm. Total visibility, that is when visibility term H_{ij} is everywhere 1,

however, is only an extreme case in the possible arrangements. The other extreme case is when the visibility term is everywhere 0, and thus the form factor will obviously be zero.

When partial occlusion occurs, the computation can make use of these two extreme cases according to the following approaches:

1. A continuous (object precision) visibility algorithm is used in the form factor computation to select the visible parts of the surfaces. Having executed this step, the parts are either totally visible or hidden from the given point on surface i.

2. The visibility term is estimated by firing several rays to surface element j and averaging their 0/1 associated visibilities. If the result is about 1, no occlusion is assumed; if it is about 0, the surface is assumed to be obscured; otherwise the surface i has to be subdivided, and the whole step repeated recursively [Tam92].

10.1.5 Hemicube algorithm

The hemicube algorithm is based on the fact that it is easier to project onto a planar rectangle than onto a spherical surface. Due to the change of the underlying geometry, the double projection cannot be expected to provide the same results as for a hemisphere, so in order to evaluate the inner form factor integral some corrections must be made during the calculation. These correction parameters are generated by comparing the needed terms and the terms resulting from the hemicube projections.

Let us examine the projection onto the top of the hemicube. Using geometric arguments and the notations of figure 10.5, the projected area of a differential patch dA_j is:

$$T(dA_j) = H_{ij} \cdot \left(\frac{R}{r}\right)^2 \cdot dA_j \cdot \frac{\cos\phi_j}{\cos\phi_i} = H_{ij} \cdot \frac{dA_j \cdot \cos\phi_j \cdot \cos\phi_i}{\pi \cdot r^2} \cdot \frac{\pi}{(\cos\phi_i)^4} \tag{10.30}$$

since $R = 1/\cos\phi_i$.

Looking at the form factor formula, we notice that a weighted area is to be calculated, where the weight function compensates for the unexpected $\pi/(\cos\phi_i)^4$ term. Introducing the compensating function w_z valid on the top of the hemicube, and expressing it by geometric considerations of figure 10.5

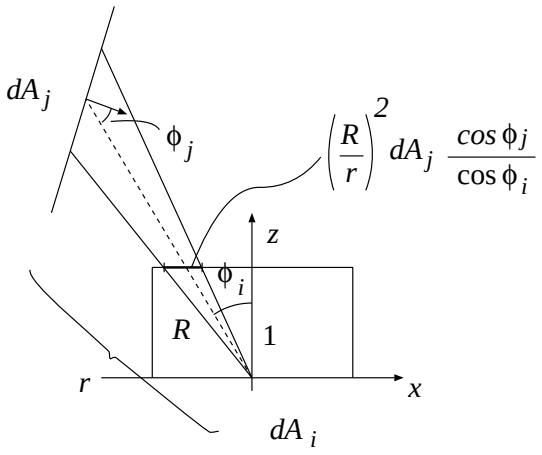

Figure 10.5: Form factor calculation by hemicube algorithm

which supposes an (x, y, z) coordinate system attached to the dA_i, with axes parallel with the sides of the hemicube, we get:

$$w_z(x, y) = \frac{(\cos \phi_i)^4}{\pi} = \frac{1}{\pi(x^2 + y^2 + 1)^2}. \tag{10.31}$$

Similar considerations can lead to the calculation of the correction terms of the projection on the side faces of the hemicube:

If the side face is perpendicular to the y axis, then:

$$w_y(x, z) = \frac{z}{\pi(x^2 + z^2 + 1)^2} \tag{10.32}$$

or if the side face is perpendicular to the x axis:

$$w_x(y, z) = \frac{z}{\pi(z^2 + y^2 + 1)^2}. \tag{10.33}$$

The weighted area defining the inner form factor is an area integral of a weight function. If ΔA_j has a projection onto the top of the hemicube only, then:

$$d_i F_{ij} = \int\limits_{\Delta A_j} T(dA_j) \cdot \frac{\cos^4 \phi_i}{\pi}. \tag{10.34}$$

Instead of integrating over ΔA_j, the same integral can also be calculated on the top of the hemicube in an x, y, z coordinate system:

$$d_i F_{ij}^{\text{top}} = \int\limits_{T(\Delta A_j)} H_{ij}(x,y) \cdot \frac{1}{\pi(x^2+y^2+1)^2}\, dxdy \tag{10.35}$$

since $\cos\phi_i = 1/(x^2+y^2+1)^{1/2}$. Indicator $H_{ij}(x,y)$ shows whether ΔA_j is really visible through hemicube point $(x,y,1)$ from ΔA_i or if it is obscured.

This integral is approximated by a finite sum having generated a $P \times P$ raster mesh on the top of the hemicube.

$$d_i F_{ij}^{\text{top}} = \int\limits_{T(\Delta A_j)} H_{ij}(x,y) \cdot w_z(x,y) dxdy \approx$$

$$\sum_{X=-P/2}^{P/2-1} \sum_{Y=-P/2}^{P/2-1} H_{ij}(X,Y) \cdot w_z(X,Y) \frac{1}{P^2}. \tag{10.36}$$

The only unknown term here is H_{ij}, which tells us whether or not surface j is visible through the raster cell called "pixel" (X,Y). Thanks to the research that has been carried out into hidden surface problems there are many effective algorithms available which can also be used here. An obvious solution is the application of simple ray tracing. The center of dA_i and the pixel defines a ray which may intersect several other surfaces. If the closest intersection is on the surface j, then $H_{ij}(X,Y)$ is 1, otherwise it is 0.

A faster solution is provided by the z-buffer method. Assume the color of the surface ΔA_j to be j, the center of the camera to be dA_i and the 3D window to be a given face of the hemicube. Having run the z-buffer algorithm, the pixels are set to the "color" of the surfaces visible in them. Taking advantage of the above definition of color (color is the index of the surface), each pixel will provide information as to which surface is visible in it. We just have to add up the weights of those pixels which contain "color" j in order to calculate the differential form factor $d_i F_{ij}^{\text{top}}$.

The projections on the side faces can be handled in exactly the same way, except that the weight function has to be selected differently (w_x or w_y depending on the actual side face). The form factors are calculated as a sum of contributions of the top and side faces.

The complete algorithm, to calculate the F_{ij} form factors using the z-buffer method, is:

```
for i = 1 to N do for j = 1 to N do F_ij = 0;
for i = 1 to N do
    camera = center of ΔA_i;
    for k = 1 to 5 do          // consider each face of the hemicube
        window = kth face of the hemicube;
        for x = 0 to P − 1 do
            for y = 0 to P − 1 do pixel[x, y] = 0;
        Z-BUFFER ALGORITHM (color of surface j is j)
        for x = 0 to P − 1 do for y = 0 to P − 1 do
            if (pixel[x, y] > 0) then
                F_{i,pixel[x,y]} += w_k(x − P/2, y − P/2)/P^2;
        endfor
    endfor
endfor
```

In the above algorithm the weight function $w_k(x-P/2, y-P/2)/P^2$ must be evaluated for those pixels through which other surfaces are visible and must be added to that form factor which corresponds to the visible surface. This is why values of weight functions at pixel centers are called **delta form factors**. Since the formula for weight functions contains many multiplications and a division, its calculation in the inner loop of the algorithm can slow down the form factor computation. However, these weight functions are common to all hemicubes, thus they must be calculated only once and then stored in a table which can be re-used whenever a value of the weight function is needed.

Since the z-buffer algorithm has $O(N \cdot P^2)$ worst case complexity, the computation of the form factors, embedding $5N$ z-buffer steps, is obviously $O(N^2 \cdot P^2)$, where N is the number of surface elements and P^2 is the number of pixels in the z-buffer. It is important to note that P can be much less than the resolution of the screen, since now the "pixels" are used only to approximate an integral finitely. Typical values of P are $50 \ldots 200$.

Since the z-buffer step can be supported by a hardware algorithm this approach is quite effective on workstations supported by graphics accelerators.

10.1.6 Cubic tetrahedral algorithm

The hemicube algorithm replaced the hemisphere by a half cube, allowing the projection to be carried out on five planar rectangles, or side faces of the cube, instead of on a spherical surface. The number of planar surfaces can be decreased by using a cubic tetrahedron as an intermediate surface [BKP91], [BKP92].

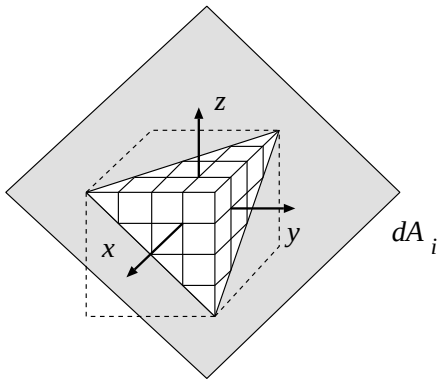

Figure 10.6: Cubic tetrahedral method

An appropriate cubic tetrahedron may be constructed by slicing a cube by a plane that passes through three of its vertices, and placing the generated pyramid on surface i (see figure 10.6). A convenient coordinate system is defined with axes perpendicular to the faces of the tetrahedron, and setting scales to place the apex in point $[1, 1, 1]$. The base of the tetrahedron will be a triangle having vertices at $[1, 1, -2]$, $[1, -2, 1]$ and $[-2, 1, 1]$.

Consider the projection of a differential surface dA_j on a side face perpendicular to x axis, using the notations of figure 10.7. The projected area is:

$$dA'_j = \frac{dA_j \cdot \cos\phi_j}{cos\theta} \cdot \frac{|\vec{R}|^2}{r^2}. \tag{10.37}$$

The correction term, to provide the internal variable in the form factor integral, is:

$$\frac{dA_j \cdot \cos\phi_j \cdot \cos\phi_i}{\pi \cdot r^2} = dA'_j \cdot \frac{\cos\phi_i \cdot \cos\theta}{\pi \cdot |\vec{R}|^2} = dA'_j \cdot w(\vec{R}). \tag{10.38}$$

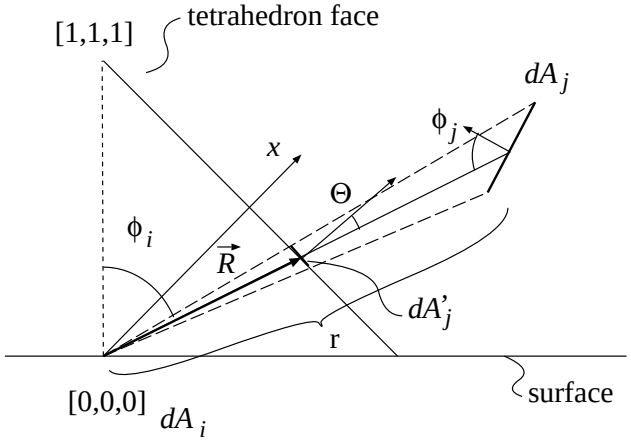

Figure 10.7: Projection to the cubic tetrahedron

Expressing the cosine of angles by a scalar product with $\vec{R}$ pointing to the projected area:

$$\cos\theta = \frac{\vec{R}\cdot[1,0,0]}{|\vec{R}|}, \qquad \cos\phi_i = \frac{\vec{R}\cdot[1,1,1]}{|\vec{R}|\cdot|[1,1,1]|}. \tag{10.39}$$

Vector $\vec{R}$ can also be defined as the sum of the vector pointing to the apex of the pyramid ($[1,1,1]$) and a linear combination of side vectors of pyramid face perpendicular to x axis:

$$\vec{R} = [1,1,1] + (1-u)\cdot[0,-1,0] + (1-v)\cdot[0,0,-1] = [1,u,v]. \tag{10.40}$$

This can be turned to the previous equation first, then to the formula of the correction term:

$$w(u,v) = \frac{u+v+1}{\pi\cdot\sqrt{3}\cdot(u^2+v^2+1)^2}. \tag{10.41}$$

Because of symmetry, the values of this weight function — that is the delta form factors — need to be computed and stored for only one-half of any face when the delta form factor table is generated. It should be mentioned that cells located along the base of the tetrahedron need special

treatment, since they have triangular shape. They can either be simply ignored, because their delta form factors are usually very small, or they can be evaluated for the center of the triangle instead of the center of the rectangular pixel.

10.2 Solution of the linear equation

The most obvious way to solve a linear equation is to apply the Gauss elimination method [PFTV88]. Unfortunately it fails to solve the radiosity equation for more complex models effectively, since it has $O(N^3)$ complexity, and also it accumulates the round of errors of digital computers and magnifies these errors to the extent that the matrix is close to singular.

Fortunately another technique, called **iteration**, can overcome both problems. Examining the radiosity equation,

$$B_i = E_i + \varrho_i \sum_j B_j \cdot F_{ij}$$

we will see that it gives the equality of the energy which has to be radiated due to emission and reflection (right side) and the energy really emitted (left side). Suppose that only estimates are available for B_j radiosities, not exact values. These estimates can be regarded as right side values, thus having substituted them into the radiosity equation, better estimates can be expected on the left sides. If these estimates were exact — that is they satisfied the radiosity equation —, then the iteration would not alter the radiosity values. Thus, if this iteration converges, its limit will be the solution of the original radiosity equation.

In order to examine the method formally, the matrix version of the radiosity equation is used to describe a single step of the iteration:

$$\mathbf{B}(m+1) = \mathbf{R} \cdot \mathbf{B}(m) + \mathbf{E}. \tag{10.42}$$

A similar equation holds for the previous iteration too. Subtracting the two equations, and applying the same consideration recursively, we get:

$$\mathbf{B}(m+1) - \mathbf{B}(m) = \mathbf{R} \cdot (\mathbf{B}(m) - \mathbf{B}(m-1)) = \mathbf{R^m} \cdot (\mathbf{B}(1) - \mathbf{B}(0)). \tag{10.43}$$

The iteration converges if

$$\lim_{m \to \infty} \|\mathbf{B}(m+1) - \mathbf{B}(m)\| = 0, \quad \text{that is if} \quad \lim_{m \to \infty} \|\mathbf{R^m}\| = 0$$

for some matrix norm. Let us use the $\|\mathbf{R}\|_\infty$ norm defined as the maximum of absolute row sums

$$\|\mathbf{R}\|_\infty = \max_i \{\sum_j F_{ij} \cdot \varrho_i\} \tag{10.44}$$

and a vector norm that is compatible with it:

$$\|\mathbf{b}\|_\infty = \max_i \{|b_i|\}. \tag{10.45}$$

Denoting $\|\mathbf{R}\|$ by q, we have:

$$\|\mathbf{B}(m+1) - \mathbf{B}(m)\| = \|\mathbf{R}^\mathbf{m} \cdot (\mathbf{B}(1) - \mathbf{B}(0))\| \leq \|\mathbf{R}\|^m \cdot \|\mathbf{B}(1) - \mathbf{B}(0)\| =$$
$$q^m \cdot \|\mathbf{B}(1) - \mathbf{B}(0)\| \tag{10.46}$$

according to the properties of matrix norms. Since F_{ij} represents the portion of the radiated energy of surface i, which actually reaches surface j, $\sum_j F_{ij}$ is that portion which is radiated towards any other surface. This obviously cannot exceed 1, and for physically correct models, diffuse reflectance $\varrho_i < 1$, giving a norm that is definitely less than 1. Consequently $q < 1$, which provides the convergence with, at least, the speed of a geometric series.

The complexity of the iteration solution depends on the operations needed for a single step and the number of iterations providing convergence. A single step of the iteration requires the multiplication of an N dimensional vector and an $N \times N$ dimensional matrix, which requires $O(N^2)$ operations.

Concerning the number of necessary steps, we concluded that the speed of the convergence is at least geometric by a factor $q = \|\mathbf{R}\|_\infty$. The infinite norm of $\mathbf{R}$ is close to being independent of the number of surface elements, since as the number of surface elements increases, the value of form factors decreases, sustaining a constant sum of rows, representing that portion of the energy radiated by surface i, which is gathered by other surfaces, multiplied by the diffuse coefficient of surface i. Consequently, the number of necessary iterations is independent of the number of surface elements, making the iteration solution an $O(N^2)$ process.

10.2.1 Gauss–Seidel iteration

The convergence of the iteration can be improved by the method of Gauss–Seidel iteration. Its basic idea is to use the new iterated values immediately

when they are available, and not to postpone their usage until the next iteration step. Consider the calculation of B_i in the normal iteration:

$$B_i(m+1) = E_i + R_{i,1} \cdot B_1(m) + R_{i,2} \cdot B_2(m) + ... + R_{i,N} \cdot B_N(m). \quad (10.47)$$

During the calculation of $B_i(m+1)$, values $B_1(m+1), ..., B_{i-1}(m+1)$ have already been calculated, so they can be used instead of their previous value, modifying the iteration, thus:

$$B_i(m+1) = E_i + R_{i,1} \cdot B_1(m+1) + \ldots + R_{i,i-1} \cdot B_{i-1}(m+1) +$$
$$R_{i,i+1} \cdot B_{i+1}(m) + \ldots + R_{i,N} \cdot B_N(m) \quad (10.48)$$

(recall that $R_{i,i} = 0$ in the radiosity equation).

A trick, called **successive relaxation**, can further improve the speed of convergence. Suppose that during the mth step of the iteration the radiosity vector $\mathbf{B}(m+1)$ was computed. The difference from the previous estimate is:

$$\Delta\mathbf{B} = \mathbf{B}(m+1) - \mathbf{B}(m) \quad (10.49)$$

showing the magnitude of difference, as well as the direction of the improvement in N dimensional space. According to practical experience, the direction is quite accurate, but the magnitude is underestimated, requiring the correction by a relaxation factor ω:

$$\mathbf{B}^*(m+1) = \mathbf{B}(m) + \omega \cdot \Delta\mathbf{B}. \quad (10.50)$$

The determination of ω is a crucial problem. If it is too small, the convergence will be slow; if it is too great, the system will be unstable and divergent. For many special matrices, the optimal relaxation factors have already been determined, but concerning our radiosity matrix, only practical experiences can be relied on. Cohen [CGIB86] suggests that relaxation factor 1.1 is usually satisfactory.

10.3 Progressive refinement

The previously discussed radiosity method determined the form factor matrix first, then solved the linear equation by iteration. Both steps require $O(N^2)$ time and space, restricting the use of this algorithm in commercial

applications. Most of the form factors, however, have very little effect on the final image, thus, if they were taken to be 0, a great amount of time and space could be gained for the price of a negligible deterioration of the image quality. A criterion for selecting unimportant form factors can be established by the careful analysis of the iteration solution of the radiosity equation:

$$B_i(m+1) = E_i + \varrho_i \sum_j B_j(m) \cdot F_{ij} = E_i + \sum_j (B_i \text{ due to } B_j(m))$$

$$(B_i \text{ due to } B_j) = \varrho_i \cdot B_j \cdot F_{ij}. \tag{10.51}$$

If B_j is small, then the whole column i of $\mathbf{R}$ will not make too much difference, thus it is not worth computing and storing its elements. This seems acceptable, but how can we decide which radiosities will be small, or which part of matrix $\mathbf{R}$ should be calculated, before starting the iteration? We certainly cannot make the decision before knowing something about the radiosities, but we can definitely do it during the iteration by calculating a column of the form factor matrix only when it turns out that it is needed, since the corresponding surface has significant radiosity.

Suppose we have an estimate B_j allowing for the calculation of the contribution of this surface to all the others, and for determining a better estimate for other surfaces by adding this new contribution to their estimated value. If an estimate B_j increases by ΔB_j, due to the contribution of other surfaces to this radiosity, other surface radiosities should also be corrected according to the new contribution of B_j, resulting in an iterative and progressive refinement of surface radiosities:

$$B_i^{\text{new}} = B_i^{\text{old}} + \varrho_i \cdot (\Delta B_j) \cdot F_{ij}. \tag{10.52}$$

Note that, in contrast to the previous radiosity method when we were interested in how a surface gathers energy from other surfaces, now the direction of the light is followed focusing on how surfaces shoot light to other surfaces. A radiosity increment of a surface, which has not yet been used to update other surface radiosities, is called **unshot radiosity**. In fact, in equation 10.52, the radiosity of other surfaces should be corrected according to the unshot radiosity of surface j. It seems reasonable to select for shooting that surface which has the highest unshot radiosity. Having selected a

surface, the corresponding column of the form factor matrix should be calculated. We can do that on every occasion when a surface is selected to shoot its radiosity. This reduces the burden of the storage of the $N \times N$ matrix elements to only a single column containing N elements, but necessitates the recalculation of the form factors. Another alternative is to store the already generated columns, allowing for reduction of the storage requirements by omitting those columns whose surfaces are never selected, due to their low radiosity. Let us realize that equation 10.52 requires $F_{1j}, F_{2j}, \ldots, F_{Nj}$, that is a single column of the form factor matrix, to calculate the radiosity updates due to ΔB_j. The hemicube method, however, supports "parallel" generation of the rows of the form factor matrix, not of the columns. For different rows, different hemicubes have to be built around the surfaces. Fortunately, the reciprocity relationship can be applied to evaluate a single column of the matrix based on a single hemicube:

$$F_{ji} \cdot \Delta A_j = F_{ij} \cdot \Delta A_i \quad \Longrightarrow \quad F_{ij} = F_{ji} \cdot \frac{\Delta A_j}{\Delta A_i} \quad (i = 1, ..., N) \qquad (10.53)$$

These considerations have formulated an iterative algorithm, called **progressive refinement**. The algorithm starts by initializing the total (B_i) and unshot (U_i) radiosities of the surfaces to their emission, and stops if the unshot radiosity is less than an acceptable threshold for all the surfaces:

for $j = 1$ **to** N **do** $B_j = E_j$; $U_j = E_j$
do
    j = Index of the surface of maximum U_j;
    Calculate F_{j1}, F_{j2} ,..., F_{jN} by a single hemicube;
    for $i = 1$ **to** N **do**
        $\Delta B_i = \varrho_i \cdot U_j \cdot F_{ji} \cdot \Delta A_j / \Delta A_i$;
        U_i += ΔB_i;
        B_i += ΔB_i;
    endfor
    $U_j = 0$;
    $error = \max\{U_1, U_2, ..., U_N\}$;
while $error > threshold$;

This algorithm is always convergent, since the total amount of unshot energy decreases in each step by an attenuation factor of less than 1. This

statement can be proven by examining the total unshot radiosities during the iteration, supposing that U_j was maximal in step m, and using the notation $q = \|\mathbf{R}\|_\infty$ again:

$$\sum_i^N U_i(m+1) = \sum_{i \neq j}^N U_i(m) + U_j \cdot \sum_i^N \varrho_i \cdot F_{ij} = (\sum_i^N U_i(m)) - U_j + U_j \sum_i^N \varrho_i \cdot F_{ij} \leq$$

$$\leq (\sum_i^N U_i(m)) - (1-q) \cdot U_j \leq (1 - \frac{1-q}{N}) \cdot \sum_i^N U_i(m) = q^* \cdot \sum_i^N U_i(m) \quad (10.54)$$

since $q = \max_i \{\sum_i^N \varrho_i \cdot F_{ij}\} < 1$ and $U_j \geq \sum_i^N U_i/N$, because it is the maximal value among U_i-s.

Note that, in contrast to the normal iteration, the attenuation factor q^* defining the speed of convergence now does depend on N, slowing down the convergence by approximately N times, and making the number of necessary iterations proportional to N. A single iteration contains a single loop of length N in progressive refinement, resulting in $O(N^2)$ overall complexity, taking into account the expected number of iterations as well. Interestingly, progressive refinement does not decrease the $O(N^2)$ time complexity, but in its simpler form when the form factor matrix is not stored, it can achieve $O(N)$ space complexity instead of the $O(N^2)$ behavior obtained by the original method.

10.3.1 Application of vertex-surface form factors

In the traditional radiosity and the discussed progressive refinement methods, the radiosity distributions of the elemental surfaces were assumed to be constant, as were the normal vectors. This is obviously far from accurate, and the effects need to be reduced by a bilinear interpolation of Gouraud shading at the last step of the image generation. In progressive refinement, however, the linear radiosity approximation can be introduced earlier, even during the phase of the calculation of radiosities. Besides, the real surface normals in the vertices of the approximating polygons can be used resulting in a more accurate computation.

This method is based on the examination of energy transfer between a differential area (dA_i) around a vertex of a surface and another finite surface (ΔA_j), and concentrates on the radiosity of vertices of polygons instead of

the radiosities of the polygons themselves. The normal of dA_i is assumed to be equal to the normal of the real surface in this point. The portion of the energy landing on the finite surface and the energy radiated by the differential surface element is called the **vertex-surface form factor** (or vertex-patch form factor).

The vertex-surface form factor, based on equation 10.6, is:

$$F_{ij}^{v} = \frac{1}{dA_i} \cdot \int_{dA_i} \int_{\Delta A_j} H_{ij} \cdot \frac{dA_i \cdot \cos\phi_i \cdot dA_j \cdot \cos\phi_j}{\pi \cdot r^2} = \int_{\Delta A_j} H_{ij} \cdot \frac{\cos\phi_i \cdot \cos\phi_j}{\pi \cdot r^2} \, dA_j. \tag{10.55}$$

This expression can either be evaluated by any discussed method or by simply firing several rays from dA_i towards the centers of the patches generated by the subdivision of surface element ΔA_j. Each ray results in a visibility factor of either 0 or 1, and an area-weighted summation has to be carried out for those patches which have visibility 1 associated with them.

Suppose that in progressive refinement total and unshot radiosity estimates are available for all vertices of surface elements. Unshot surface radiosities can be approximated as the average of their unshot vertex radiosities. Having selected the surface element with the highest unshot radiosity (U_j), and having also determined the vertex-surface form factors from all the vertices to the selected surface (note that this is the reverse direction), the new contributions to the total and unshot radiosities of vertices are:

$$\Delta B_i^v = \varrho_i \cdot U_j \cdot F_{ij}^v. \tag{10.56}$$

This has modified the total and unshot radiosities of the vertices. Thus, estimating the surface radiosities, the last step can be repeated until convergence, when the unshot radiosities of vertices become negligible. The radiosity of the vertices can be directly turned to intensity and color information, enabling Gouraud's algorithm to complete the shading for the internal pixels of the polygons.

10.3.2 Probabilistic progressive refinement

In probabilistic form factor computation, rays were fired from surfaces to determine which other surfaces can absorb their radiosity. In progressive refinement, on the other hand, the radiosity is shot proportionally to the

precomputed form factors. These approaches can be merged in a method which randomly shoots photons carrying a given portion of energy. As in progressive refinement, the unshot and total radiosities are initialized to the emission of the surfaces. At each step of the iteration a point is selected at random on the surface which has the highest unshot radiosity, a direction is generated according to the directional distribution of the radiation (cosine distribution), and a given portion, say $1/n$th, of the unshot energy is delivered to that surface which the photon encounters first on its way.

The program of this algorithm is then:

for $j = 1$ **to** N **do** $B_j = U_j = E_j$
do
  j = Index of the surface of maximum U_j
  $\vec{p}$ = a random point on surface j by uniform distribution
  $\vec{d}$ = a random direction from $\vec{p}$ by cosine distribution
  if $ray(\vec{p}, \vec{d})$ hits surface i first **then**
    U_i += $\varrho_i \cdot U_j/n$;
    B_i += $\varrho_i \cdot U_j/n$;
  endif
  U_j -= U_j/n;
  $error = \max\{U_1, U_2, ..., U_N\}$;
while $error > threshold$;

This is possibly the simplest algorithm for radiosity calculation. Since it does not rely on form factors, shading models other than diffuse reflection can also be incorporated.

10.4 Extensions to non-diffuse environments

The traditional radiosity methods discussed so far consider only diffuse reflections, having made it possible to ignore directional variation of the radiation of surfaces, since diffuse reflection generates the same radiant intensity in all directions. To extend the basic method taking into account more

terms in the general shading equation, directional dependence has to be built into the model.

The most obvious approach is to place a partitioned sphere on each elemental surface, and to calculate and store the intensity in each solid angle derived from the partition [ICG86]. This partitioning also transforms the integrals of the shading equations to finite sums, and limits the accuracy of the direction of the incoming light beams. Deriving a shading equation for each surface element and elemental solid angle, a linear equation is established, where the unknown variables are the radiant intensities of the surfaces in various solid angles. This linear equation can be solved by similar techniques to those discussed so far. The greatest disadvantage of this approach is that it increases the number of equations and the unknown variables by a factor of the number of partitioning solid angles, making the method prohibitively expensive.

More promising is the combination of the radiosity method with ray tracing, since the respective strong and weak points of the two methods tend to complement each other.

10.4.1 Combination of radiosity and ray tracing

In its simplest approach, the final, view-dependent step of the radiosity method involving Gouraud shading and usually z-buffering can be replaced by a recursive ray tracing algorithm, where the diffuse component is determined by the surface radiosities, instead of taking into consideration the abstract lightsources, while the surface radiosities are calculated by the methods we have discussed, ignoring all non-diffuse phenomena. The result is much better than the outcome of a simple recursive ray tracing, since the shadows lose their sharpness. The method still neglects some types of coupling, since, for example, it cannot consider the diffuse reflection of a light beam coherently reflected or refracted onto other surfaces. In figure 10.8, for example, the vase should have been illuminated by the light coherently reflected off the mirror, but the algorithm in question makes it dark, since the radiosity method ignores non-diffuse components. A possible solution to this problem is the introduction and usage of **extended form factors** [SP89].

A simplified shading model is used which breaks down the energy radiated by a surface into diffuse and coherently reflected or refracted components.

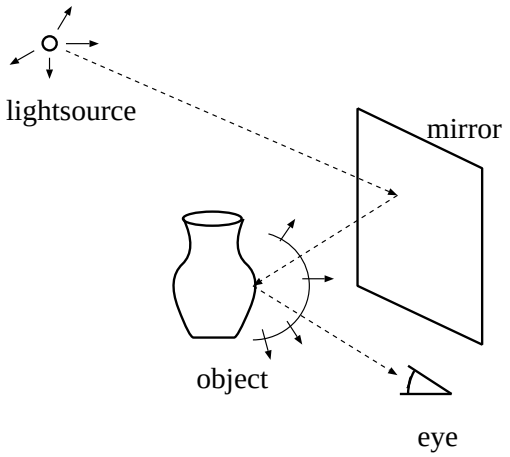

Figure 10.8: Coherent-diffuse coupling: The vase should have been illuminated by the light reflected off the mirror

An extended form factor F^*_{ij}, by definition represents that portion of the energy radiated diffusely by surface i which actually reaches surface j either by direct transmission or by single or multiple coherent reflections or refractions. The use of extended form factors allows for the calculation of the diffuse radiance of patches which takes into account not only diffuse but also coherent interreflections. Suppose diffuse radiance B_i of surface i needs to be calculated. Diffuse radiance B_i is determined by the diffuse radiation of other surfaces which reaches surface i and by those light components which are coherently reflected or refracted onto surface i. These coherent components can be broken down into diffuse radiances and emissions which are later coherently reflected or refracted several times, thus similar expression holds for the diffuse radiance in the improved model as for the original, only the normal form factors must be replaced by the extended ones. The extended radiosity equation defining the diffuse radiance of the surfaces in a non-diffuse environment is then:

$$B_i \cdot dA_i = E_i \cdot dA_i + \varrho_i \cdot \int B_j \cdot F^*_{ji} \cdot dA_j. \tag{10.57}$$

Recursive ray tracing can be used to calculate the extended form factors. For each pixel of the hemicube a ray is generated which is traced backward

finding those surfaces which can be visited along this ray. For each surface found that portion of its diffusely radiated energy which reaches the previous surface along the ray should be computed — this is a differential form factor — then the attenuation of subsequent coherent reflections and refractions must be taken into consideration by multiplying the differential form factor by the product of the refractive and reflective coefficients of the surfaces visited by the ray between the diffuse source and surface i. Adding these portions of possible contribution for each pixel of the hemicube also taking the hemicube weighting function into account, the extended form factors can be generated. Having calculated the extended form factors, the radiosity equation can be solved by the method discussed, resulting in the diffuse radiosities of the surfaces.

Expensive ray-tracing can be avoided and normal form factors can be worked with if only single, ideal, mirror-like coherent reflections are allowed, because this case can be supported by mirroring every single surface onto the reflective surfaces. We can treat these reflective surfaces as windows onto a "mirror world", and the normal form factor between the mirrored surface and another surface will be responsible for representing that part of energy transfer which would be represented by the difference of the extended and normal form factors [WCG87].

If the diffuse radiosities of the surfaces are generated, then in the second, view-dependent phase another recursive ray-tracing algorithm can be applied to generate the picture. Whenever a diffuse intensity is needed this second pass ray-tracing will use the radiosities computed in the first pass. In contrast to the naive combination of ray-tracing and radiosity, the diffuse radiosities are now correct, since the first pass took not only the diffuse interreflections but also the coherent interreflections and refractions into consideration.

10.5 Higher order radiosity approximation

The original radiosity method is based on finite element techniques. In other words, the radiosity distribution is searched in a piecewise constant function form, reducing the original problem to the calculation of the values of the steps.

The idea of piecewise constant approximation is theoretically simple and easy to accomplish, but an accurate solution would require a large number of steps, making the solution of the linear equation difficult. Besides, the constant approximation can introduce unexpected artifacts in the picture even if it is softened by Gouraud shading.

This section addresses this problem by applying a variational method for the solution of the integral equation [SK93].

The variational solution consists of the following steps [Mih70]:

1. It establishes a functional which is extreme for a function (radiosity distribution) if and only if the function satisfies the original integral equation (the basic radiosity equation).

2. It generates the extreme solution of the functional by **Ritz's method**, that is, it approximates the function to be found by a function series, where the coefficients are unknown parameters, and the extremum is calculated by making the partial derivatives of the functional (which is a function of the unknown coefficients) equal to zero. This results in a linear equation which is solved for the coefficients defining the radiosity distribution function.

Note the similarities between the second step and the original radiosity method. The proposed variational method can, in fact, be regarded as a generalization of the finite element method, and, as we shall see, it contains that method if the basis functions of the function series are selected as piecewise constant functions being equal to zero except for a small portion of the surfaces. Nevertheless, we are not restricted to these basis functions, and can select other function bases, which can approximate the radiosity distribution more accurately and by fewer basis functions, resulting in a better solution and requiring the calculation of a significantly smaller linear equation.

Let the diffuse coefficient be $\varrho(p)$ at point p and the visibility indicator between points p and p' be $H(p,p')$. Using the notations of figure 10.9, and denoting the radiosity and emission at point p by $B(p)$ and $E(p)$ respectively, the basic radiosity equation is:

$$B(p) \cdot dA = E(p) \cdot dA + \varrho(p) \cdot \int_A B(p') \; f(p,p') \; dA' \cdot dA \qquad (10.58)$$

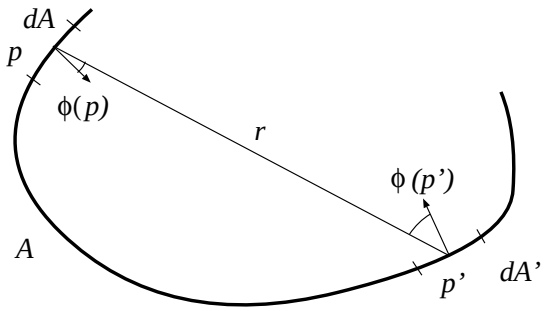

Figure 10.9: Geometry of the radiosity calculation

where $f(p, p')$ is the **point-to-point form factor**:

$$f(p, p') = H(p, p') \frac{\cos \phi(p) \cdot \cos \phi(p')}{r^2 \pi}. \tag{10.59}$$

Dividing both sides by dA, the **radiosity equation** is then:

$$B(p) = E(p) + \varrho(p) \cdot \int_A B(p')\ f(p, p')\ dA'. \tag{10.60}$$

Let us define a linear operator $\mathcal{L}$:

$$\mathcal{L}B(p) = B(p) - \varrho(p) \cdot \int_A B(p')\ f(p, p')\ dA'. \tag{10.61}$$

Then the radiosity equation can also be written as follows:

$$\mathcal{L}B(p) = E(p). \tag{10.62}$$

The solution of the radiosity problem means to find a function B satisfying this equation. The domain of possible functions can obviously be restricted to functions whose square has finite integration over surface A. This function space is usually called $L_2(A)$ space where the scalar product is defined as:

$$\langle u, v \rangle = \int_A u(p) \cdot v(p)\ dA. \tag{10.63}$$

If $\mathcal{L}$ were a symmetric and positive operator, that is, for any u, v in $L_2(A)$,

$$\langle \mathcal{L}u, v \rangle = \langle u, \mathcal{L}v \rangle \tag{10.64}$$

were an identity and

$$\langle \mathcal{L}u, u \rangle \geq 0 \qquad \wedge \qquad \langle \mathcal{L}u, u \rangle = 0 \quad \text{if and only if } u = 0, \tag{10.65}$$

then according to the *minimal theorem of quadratic functionals* [Ode76] the solution of equation 10.62 could also be found as the stationary point of the following functional:

$$\langle \mathcal{L}B, B \rangle - 2\langle E, B \rangle + \langle E, E \rangle. \tag{10.66}$$

Note that $\langle E, E \rangle$ makes no difference in the stationary point, since it does not depend on B, but it simplifies the resulting formula.

To prove that if and only if some B_0 satisfies

$$\mathcal{L}B_0 = E \tag{10.67}$$

for a symmetric and positive operator $\mathcal{L}$, then B_0 is extreme for the functional of equation 10.66, a sequence of identity relations based on the assumption that $\mathcal{L}$ is positive and symmetric can be used:

$$\begin{gathered}
\langle \mathcal{L}B, B \rangle - 2\langle E, B \rangle + \langle E, E \rangle = \langle \mathcal{L}B, B \rangle - 2\langle \mathcal{L}B_0, B \rangle + \langle E, E \rangle = \\
\langle \mathcal{L}B, B \rangle - \langle \mathcal{L}B_0, B \rangle - \langle B_0, \mathcal{L}B \rangle + \langle E, E \rangle = \\
\langle \mathcal{L}B, B \rangle - \langle \mathcal{L}B_0, B \rangle - \langle \mathcal{L}B, B_0 \rangle + \langle \mathcal{L}B_0, B_0 \rangle - \langle \mathcal{L}B_0, B_0 \rangle + \langle E, E \rangle = \\
\langle \mathcal{L}(B - B_0), (B - B_0) \rangle - \langle \mathcal{L}B_0, B_0 \rangle + \langle E, E \rangle.
\end{gathered} \tag{10.68}$$

Since only the term $\langle \mathcal{L}(B - B_0), (B - B_0) \rangle$ depends on B and this term is minimal if and only if $B - B_0$ is zero due to the assumption that $\mathcal{L}$ is positive, therefore the functional is really extreme for that B_0 which satisfies equation 10.62.

Unfortunately $\mathcal{L}$ is not symmetric in its original form (equation 10.61) due to the asymmetry of the radiosity equation which depends on $\varrho(p)$ but not on $\varrho(p')$. One possible approach to this problem is the subdivision of surfaces into finite patches having constant diffuse coefficients, and working

with multi-variate functionals, but this results in a significant computational overhead.

Now another solution is proposed that eliminates the asymmetry by calculating $B(p)$ indirectly through the generation of $B(p)/\sqrt{\varrho(p)}$. In order to do this, both sides of the radiosity equation are divided by $\sqrt{\varrho(p)}$:

$$\frac{E(p)}{\sqrt{\varrho(p)}} = \frac{B(p)}{\sqrt{\varrho(p)}} - \sqrt{\varrho(p)} \int_A \frac{B(p')}{\sqrt{\varrho(p')}} \sqrt{\varrho(p')}\ f(p,p')\ dA'. \tag{10.69}$$

Let us define $B^*(p)$, $E^*(p)$ and $g(p,p')$ by the following formulae:

$$B^*(p) = \frac{B(p)}{\sqrt{\varrho(p)}}, \qquad E^*(p) = \frac{E(p)}{\sqrt{\varrho(p)}}, \qquad g(p,p') = f(p,p')\sqrt{\varrho(p)\varrho(p')}. \tag{10.70}$$

Using these definitions, we get the following form of the original radiosity equation:

$$E^*(p) = B^*(p) - \int_A B^*(p')\ g(p,p')\ dA'. \tag{10.71}$$

Since $g(p,p') = g(p',p)$, this integral equation is defined by a symmetric linear operator $\mathcal{L}^*$:

$$\mathcal{L}^* B^*(p) = B^*(p) - \int_A B^*(p')\ g(p,p')\ dA'. \tag{10.72}$$

As can easily be proven, operator $\mathcal{L}^*$ is not only symmetric but also positive taking into account that for physically correct models:

$$\int_A \int_A B(p') g(p,p')\ dA' dA \le \int_A B(p)\ dA. \tag{10.73}$$

This means that the solution of the modified radiosity equation is equivalent to finding the stationary point of the following functional:

$$I(B^*) = \langle \mathcal{L}^* B^*, B^* \rangle - 2\langle E^*, B^* \rangle + \langle E^*, E^* \rangle =$$
$$\int_A (E^*(p) - B^*(p))^2 dA - \int_A \int_A B^*(p)\ B^*(p')\ g(p,p')\ dA\ dA'. \tag{10.74}$$

This extreme property of functional I can also be proven by generating the functional's first variation and making it equal to zero:

$$0 = \delta I = \frac{\partial I(B^* + \alpha\ \delta B)}{\partial \alpha}\Big|_{\alpha=0}. \tag{10.75}$$

Using elementary derivation rules and taking into account the following symmetry relation:

$$\int_A \int_A B^*(p)\ \delta B(p')\ g(p,p')\ dAdA' = \int_A \int_A \delta B(p)\ B^*(p')\ g(p,p')\ dAdA' \tag{10.76}$$

the formula of the first variation is transformed to:

$$0 = \delta I = \int_A [E^*(p) - B^*(p) + \int_A B^*(p') \cdot g(p,p')\ dA'] \cdot \delta B\ dA. \tag{10.77}$$

The term closed in brackets should be zero to make the expression zero for any δB variation. That is exactly the original radiosity equation, hence finding the stationary point of functional I is really equivalent to solving integral equation 10.71.

In order to find the extremum of functional $I(B^*)$, Ritz's method is used. Assume that the unknown function B^* is approximated by a function series:

$$B^*(p) \approx \sum_{k=1}^{n} a_k \cdot b_k(p) \tag{10.78}$$

where $(b_1, b_2, ..., b_n)$ form a complete function system (that is, any piecewise continuous function can be approximated by their linear combination), and $(a_1, a_2, ..., a_n)$ are unknown coefficients. This assumption makes functional $I(B^*)$ an n-variate function $I(a_1, ..., a_n)$, which is extreme if all the partial derivatives are zero. Having made every $\partial I/\partial a_k$ equal to zero, a linear equation system can be derived for the unknown a_k-s ($k = \{1, 2, ..., n\}$):

$$\sum_{i=0}^{n} a_i [\int_A b_i(p) b_k(p) dA - \int_A \int_A b_k(p) b_i(p') g(p,p')\ dAdA'] = \int_A E^*(p) b_k(p) dA. \tag{10.79}$$

This general formula provides a linear equation for any kind of complete function system $b_1, ..., b_n$, thus it can be regarded as a basis of many different radiosity approximation techniques, because the different selection of basis functions, b_i, results in different methods of determining the radiosity distribution.

Three types of function bases are discussed:

- piecewise constant functions which lead to the traditional method, proving that the original approach is a special case of this general framework,
- piecewise linear functions which, as we shall see, are not more difficult than the piecewise constant approximations, but they can provide more accurate solutions. It is, in fact, a refined version of the method of "**vertex-surface form factors**",
- harmonic (cosine) functions where the basis functions are not of finite element type because they can approximate the radiosity distribution everywhere not just in a restricted part of the domain, and thus fall into the category of global element methods.

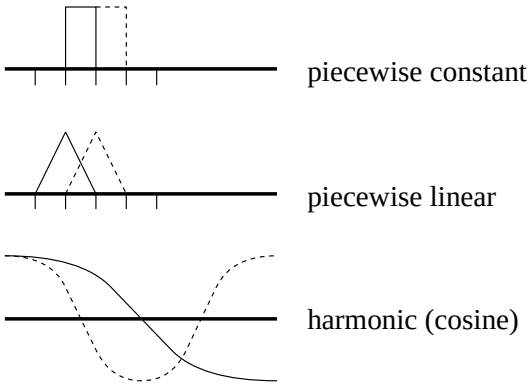

Figure 10.10: One-dimensional analogy of proposed basis functions

10.5.1 Piecewise constant radiosity approximation

Following a finite element approach, an appropriate set of b_k functions can be defined having broken down the surface into ΔA_1, ΔA_2,...,ΔA_n surface elements:

$$b_k(p) = \begin{cases} 1 \text{ if } p \text{ is on } \Delta A_k \\ \\ 0 \text{ otherwise} \end{cases} \tag{10.80}$$

If the emission E and the diffuse coefficient ϱ are assumed to be constant on the elemental surface ΔA_k and equal to E_k and ϱ_k respectively, equation 10.79 will have the following form:

$$a_k \Delta A_k - \sum_{i=0}^{n} a_i [\int_{\Delta A_k} \int_{\Delta A_i} g(p, p') \; dA dA'] = \frac{E_k}{\sqrt{\varrho_k}} \Delta A_k. \tag{10.81}$$

According to the definition of basis function b_k, the radiosity of patch k is:

$$B_k = B_k^* \sqrt{\varrho_k} = a_k \sqrt{\varrho_k}. \tag{10.82}$$

Substituting this into equation 10.81 and using the definition of $g(p, p')$ in equation 10.70, we get:

$$B_k \Delta A_k - \varrho_k \sum_{i=0}^{n} B_i [\int_{\Delta A_k} \int_{\Delta A_i} f(p, p') \; dA dA'] = E_k \Delta A_k. \tag{10.83}$$

Let us introduce the **patch-to-patch form factor** as follows:

$$F_{ki} = \frac{1}{\Delta A_k} \int_{\Delta A_k} \int_{\Delta A_i} f(p, p') \; dA dA'. \tag{10.84}$$

Note that this is the usual definition taking into account the interpretation of $f(p, p')$ in equation 10.59.

Dividing both sides by ΔA_k, the linear equation is then:

$$B_k - \varrho_k \sum_{i=0}^{n} B_i F_{ki} = E_k. \tag{10.85}$$

This is exactly the well known linear equation of original radiosity method (equation 10.10). Now let us begin to discuss how to define and use other, more effective function bases.

10.5.2 Linear finite element techniques

Let us decompose the surface into planar triangles and assume that the radiosity variation is linear on these triangles. Thus, each vertex i of the triangle mesh will correspond to a "tent shaped" basis function b_i that is 1

at this vertex and linearly decreases to 0 on the triangles incident to this vertex.

Placing the center of the coordinate system into vertex i, the position vector of points on an incident triangle can be expressed by a linear combination of the edge vectors $\vec{a}, \vec{b}$:

$$\vec{p} = \alpha\vec{a} + \beta\vec{b} \tag{10.86}$$

with $\alpha, \beta \geq 0 \quad \wedge \quad \alpha + \beta \leq 1$.

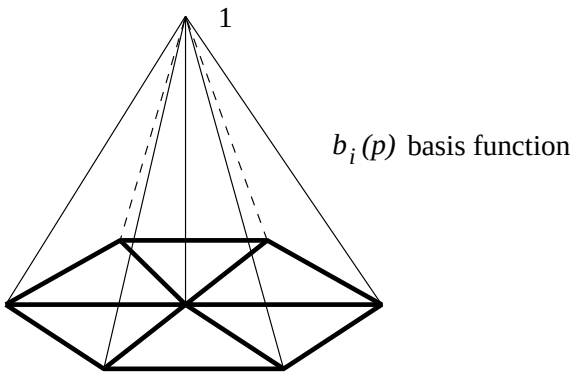

Figure 10.11: Linear basis function in three dimensions

Thus, the surface integral of some function F on a triangle can be written as follows:

$$\int_{\Delta A} F(\vec{p})dA = \int_{\beta=0}^{1} \int_{\alpha=0}^{1-\beta} F(\alpha,\beta)|\vec{a} \times \vec{b}|d\alpha d\beta =$$

$$2\Delta A \int_{\beta=0}^{1} \int_{\alpha=0}^{1-\beta} F(\alpha,\beta)\ d\alpha d\beta. \tag{10.87}$$

If $F(\alpha, \beta)$ is a polynomial function, then its surface integration can be determined in closed form by this formula.

The basis function which is linearly decreasing on the triangles can be conveniently expressed by α, β coordinates:

$$\begin{aligned} &b_k(\alpha,\beta) = 1-\alpha-\beta, \\ &b_{k'}(\alpha,\beta) = \alpha, \\ &b_{k''}(\alpha,\beta) = \beta, \\ &b_i = 0 \qquad\qquad \text{if } i \neq k, k', k'' \end{aligned} \tag{10.88}$$

where k, k' and k'' are the three vertices of the triangle.

Let us consider the general equation (equation 10.79) defining the weights of basis functions; that is the radiosities at triangle vertices for linear finite elements. Although its integrals can be evaluated directly, it is worth examining whether further simplification is possible. Equation 10.79 can also be written as follows:

$$\int_A [\sum_{i=0}^{n} a_i \{b_i(p) - \int_A b_i(p') g(p,p')\ dA'\} - E^*(p)] \cdot b_k(p)\ dA = 0 \tag{10.89}$$

The term enclosed in brackets is a piecewise linear expression according to our assumption if E^* is also linear. The integration of the product of this expression and any linear basis function is zero. That is possible if the term in brackets is constantly zero, thus an equivalent system of linear equations can be derived by requiring the closed term to be zero in each vertex k (this implies that the function will be zero everywhere because of linearity):

$$a_k - \sum_{i=0}^{n} a_i \int_A b_i(p') g(p_k,p')\ dA' = E_k^*, \qquad k = \{1,2,...n\} \tag{10.90}$$

As in the case of piecewise constant approximation, the diffuse coefficient $\varrho(p)$ is assumed to be equal to ϱ_k at vertex k, and using the definitions of the normalized radiosities we can conclude that:

$$a_k = B_k^* = \frac{B_k}{\sqrt{\varrho_k}}, \qquad E_k^* = \frac{E_k}{\sqrt{\varrho_k}}. \tag{10.91}$$

Substituting this into equation 10.90 and taking into account that b_i is zero outside ΔA_i, we get:

$$B_k - \varrho_k \sum_{i=0}^{n} B_i [\int_{\Delta A_i} b_i(p') f(p_k,p') \sqrt{\frac{\varrho(p')}{\varrho_i}}\ dA'] = E_k. \tag{10.92}$$

Let us introduce the **vertex-patch form factor** P_{ki}:

$$P_{ki} = \int_{\Delta A_i} b_i(p') f(p_k, p') \sqrt{\frac{\varrho(p')}{\varrho_i}}\, dA'. \tag{10.93}$$

If the diffuse coefficient can be assumed to be (approximately) constant on the triangles adjacent to vertex i, then:

$$P_{ki} \approx \int_{\Delta A_i} b_i(p') f(p_k, p')\, dA'. \tag{10.94}$$

The linear equation of the vertex radiosities is then:

$$B_k - \varrho_k \sum_{i=0}^{n} B_i P_{ki} = E_k. \tag{10.95}$$

This is almost the same as the linear equation describing the piecewise constant approximation (equation 10.85), except that:

- Unknown parameters $B_1, ..., B_k$ represent now vertex radiosities rather than patch radiosities. According to Euler's law, the number of vertices of a triangular faced polyhedron is half of the number of its faces plus two. Thus the size of the linear equation is almost the same as for the number of quadrilaterals used in the original method.
- There is no need for double integration and thus the linear approximation requires a simpler numerical integration to calculate the form factors than constant approximation.

The vertex-patch form factor can be evaluated by the techniques developed for patch-to-patch form factors taking account also the linear variation due to b_i. This integration can be avoided, however, if linear approximation of $f(p_k, p')$ is acceptable. One way of achieving this is to select the subdivision criterion of surfaces into triangles accordingly.

A linear approximation can be based on point-to-point form factors between vertex k and the vertices of triangle $\Delta A'$. Let the $f(p_k, p)$ values of the possible combinations of point p_k and the vertices be F_1, F_2, F_3 respectively. A linear interpolation of the point-to-point form factor between p_k and $p' = \alpha'\vec{a}' + \beta'\vec{b}'$ is:

$$f(p_k, p') = \alpha' F_1 + \beta' F_2 + (1 - \alpha' - \beta') F_3. \tag{10.96}$$

Using this assumption the surface integral defining P_{ki} can be expressed in closed form.

10.5.3 Global element approach — harmonic functions

In contrast to previous cases, the application of harmonic functions does not require the subdivision of surfaces into planar polygons, but deals with the original geometry. This property makes it especially useful when the view-dependent rendering phase uses ray-tracing.

Suppose surface A is defined parametrically by a position vector function, $\vec{r}(u, v)$, where parameters u and v are in the range of $[0, 1]$.

Let a representative of the basis functions be:

$$b_{ij} = \cos(i\pi u) \cdot \cos(j\pi v) = C_u^i C_v^j \tag{10.97}$$

(C_u^i substitutes $\cos(i\pi u)$ for notational simplicity). Note that the basis functions have two indices, hence the sums should also be replaced by double summation in equation 10.79. Examining the basis functions carefully, we can see that the goal is the calculation of the Fourier series of the radiosity distribution.

In contrast to the finite element method, the basis functions are now non-zero almost everywhere in the domain, so they can approximate the radiosity distribution in a wider range. For that reason, approaches applying this kind of basis function are called **global element methods**.

In the radiosity method the most time consuming step is the evaluation of the integrals appearing as coefficients of the linear equation system (equation 10.79). By the application of cosine functions, however, the computational time can be reduced significantly, because of the orthogonal properties of the trigonometric functions, and also by taking advantage of effective algorithms, such as Fast Fourier Transform (FFT).

In order to illustrate the idea, the calculation of

$$\int_A E^*(p) b_{kl}(p)\ dA$$

for each k, l is discussed. Since $E^*(p) = E^*(\vec{r}(u, v))$, it can be regarded as a function defined over the square $[0, 1]^2$. Using the equalities of surface

integrals, and introducing the notation $J(u,v) = |\partial\vec{r}/\partial u \times \partial\vec{r}/\partial v|$ for surface element magnification, we get:

$$\int_A E^*(p) b_{kl}(p)\ dA = \int_0^1 \int_0^1 E^*(\vec{r}(u,v)) b_{kl}(u,v) J(u,v)\ du dv. \tag{10.98}$$

Let us mirror the function $E^*(\vec{r}) \cdot J(u,v)$ onto coordinate system axes u and v, and repeat the resulting function having its domain in $[-1,1]^2$ infinitely in both directions with period 2. Due to mirroring and periodic repetition, the final function $\hat{E}(u,v)$ will be even and periodic with period 2 in both directions. According to the theory of the Fourier series, the function can be approximated by the following sum:

$$\hat{E}(u,v) \approx \sum_{i=0}^{m} \sum_{j=0}^{m} E_{ij} C_u^i C_v^j. \tag{10.99}$$

All the Fourier coefficients E_{ij} can be calculated by a single, two-dimensional FFT. (A D-dimensional FFT of N samples can be computed by taking DN^{D-1} number of one-dimensional FFTs [Nus82] [PFTV88].)

Since $\hat{E}(u,v) = E^*(\vec{r}) \cdot J(u,v)$ if $0 \le u, v \le 1$, this Fourier series and the definition of the basis functions can be applied to equation 10.98, resulting in:

$$\int_A E^*(p) b_{kl}(p)\ dA = \int_{u=0}^1 \int_{v=0}^1 \sum_{i=0}^{m} \sum_{j=0}^{m} E_{ij} C_u^i C_v^j \cdot b_{kl}(u,v)\ du dv =$$

$$\sum_{i=0}^{m} \sum_{j=0}^{m} E_{ij} \int_0^1 C_u^i C_u^k du \int_0^1 C_v^j C_v^l dv = \begin{cases} E_{0,0} & \text{if } k = 0 \text{ and } l = 0 \\ E_{0,l}/2 & \text{if } k = 0 \text{ and } l \neq 0 \\ E_{k,0}/2 & \text{if } k \neq 0 \text{ and } l = 0 \\ E_{k,l}/4 & \text{if } k \neq 0 \text{ and } l \neq 0 \end{cases} \tag{10.100}$$

Consequently, the integral can be calculated in closed form, having replaced the original function by Fourier series. Similar methods can be used to evaluate the other integrals. In order to compute

$$\int_A b_{ij}(p) b_{kl}(p) dA$$

$J(u,v)$ must be Fast Fourier Transformed.

To calculate

$$\int_A \int_A b_k(p) b_i'(p) g(p,p')\, dA dA'$$

the Fourier transform of

$$g(p(u,v),p'(u',v')) \cdot J(u,v)J(u',v')$$

is needed. Unfortunately the latter requires a 4D FFT which involves many operations. Nevertheless, this transform can be realized by two two-dimensional FFTs if $g(p,p')$ can be assumed to be nearly independent of either p or p', or it can be approximated by a product form of p and p' independent functions.

Finally, it should be mentioned that other **global function bases** can also be useful. For example, Chebyshev polynomials are effective in approximation, and similar techniques to FFT can be developed for their computation.