RADIOSITY: An Illuminating Perspective

Steven M. Drucker

Media Laboratory, MIT.
Written Doctoral Exam
Professor David Zeltzer

Abstract

This paper presents a summary of the current state of research in the area of radiosity. Besides discussing the basic formulation for radiosity, we also examine several extensions of radiosity; inclusion of specular effects, adaptation to dynamic environments, and acceleration methods.

I. Introduction:

Throughout its history, one interest of computer graphics has been to produce interesting realistic pictures quickly and accurately. There have been several methods introduced in the literature that attempt to model the interactions between light and the surfaces of a scene to achieve this result. These methods have progressively moved from techniques which merely produce good pictures to techniques which have a valid physical basis.

Early attempts, characterized as "incremental methods" by Hall [Hall86], transformed the geometry of a scene using perspective projections into a flat image and then used scan-line rendering to compute the color of the surfaces. These techniques are called incremental because each pixel on the screen is based on the color of the previous pixel or scanline. The typical lighting model used in incremental algorithms is as follows:

$$I(\lambda) = f(d) \times (\text{ambient} + \text{diffuse} + \text{specular})$$

where $I(\lambda)$ is the intensity for a particular wavelength, $f(d)$ is a function based on the distance from the viewer, and the ambient light is an empirical term added to account for all the light in the scene that is not represented by direct influence from identified light sources.

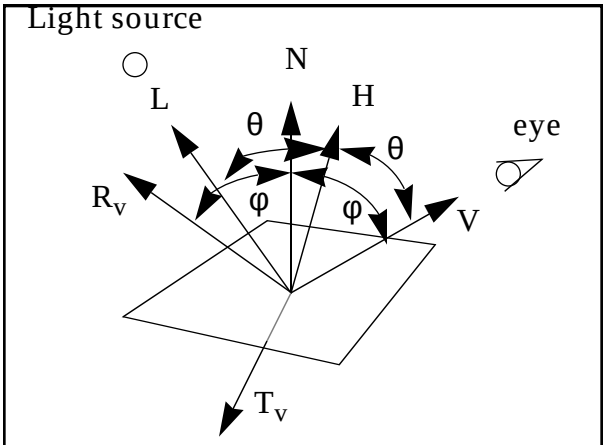


Fig. 1: vector notation for illumination models

Using the vectors in Fig. 1, we can further define the diffuse and specular terms so that the general lighting model for the incremental rendering techniques is:

$$I(\lambda) = f(d) \times \left(K_a \lambda I_a + K_d \lambda \sum_{j=1}^{\text{lights}} N \cdot L_j + K_s \lambda \sum_{j=1}^{\text{lights}} (R_v \cdot L_j)^{Ns}\right) \quad (1)$$

This is just one of the incremental shading models, known as Phong Shading, for a more complete description, see [Hall87]. Besides problems with perspective distortions, these models do not accurately depict the complex effects of light interacting with the objects in a scene; no shadows are generated, and shading is based only on the initial position of the

lights. No reflection or transmission is taken into account.

Raytracing was introduced in the mid 1970's and corrected many of these deficits. Raytracing calculates both the geometry and the color for every pixl of the image. Raytracing can account for shadowing and interreflections, but besides being computationally expensive, it works well only on smooth, highly specular surfaces. This is because the number of rays that must be traced becomes prohibitively large when diffusely reflecting surfaces are modeled. Methods have been introduced in raytracing that both increase the computational efficiency of the algorithms, and add to their capability for handling diffuse scenes, but at the present time, however, another method is much better suited to this very important type of environment (some say that diffuse objects make up more than 70% of a typical scene [Greenberg86]). This method is known as "radiosity."

Around 1984, methods of radiative heat transfer were introduced into the computer graphics literature and the name "radiosity" was coined to describe this new method for generating photorealistic images [Goral84]. Radiosity is particularly suited to diffuse environments, and the algorithm is based in object space rather than image space. This means that the colors of objects are calculated before they are projected onto the screen thus once the initial calculations have been made, they can be rendered from any point of view using fast hardware rendering techniques. This aspect alone makes research in radiosity extremely important as interest in realistic "virtual environments" becomes much more prevalent.

This paper will describe the basic radiosity method and discuss extensions of various kinds, including techniques for incorporating specular effects in radiosity, adaptation to dynamic environments, and parallel implementations. Not only can radiosity create extremely realistic pictures, but it can also accurately simulate the complex electromagnetic interactions that exist in closed spaces. This makes other applications possible besides photorealism.

II. Basic Radiosity Formulation:

The derivation of radiosity described in this section draws from [Kajiya90], [Hall90], and [Cohen90].

Before deriving radiosity, we must first establish what various terms mean. Since light is electromagnetic radiation, to measure the intensity of this radiation, we must measure energy flow, the rate of radiant energy per time, which is also called *flux*:

$$\Phi_L = \frac{dQ_L}{dt} \tag{2}$$

To measure a flow, we must define a region of space across which we measure that flow. Essentially we define an area across which we measure the rate of energy and continually shrink this area until we are measuring what is called the *flux density* or *energy density* which leaves a surface; for diffuse surfaces, this is also known as the radiosity: B.

$$B = \frac{d\Phi_L}{dA} \tag{3}$$

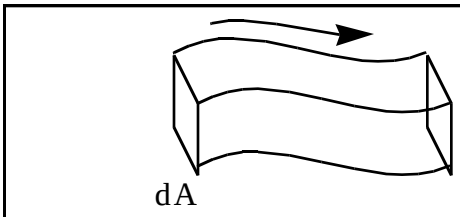


Fig. 2: Flow across a differential area.

A light might not radiate flux uniformly in all directions; if we measure the flux density

at a unit distance away, we must specify at what solid angle we measure. We define the intensity I as the energy flux per unit projected area of the source per solid angle ω, through which the source radiates, see Fig. 3:

$$I = \frac{d\Phi_L}{dA d\omega} \tag{4}$$

In terms of intensity, the energy flux is the intensity integrated over the solid angle of the source and the area of the receiving surface:

$$\Phi = \iint_{\omega A} I dA d\omega \tag{5}$$

Lastly, we define the *bidirectional reflectance* R_{bd} as:

$$I_R = \frac{R_{bd}\Phi_L}{A} \tag{6}$$

and the *bidirectional transmittance* as:

$$I_T = \frac{T_{bd}\Phi_L}{A} \tag{7}$$

These terms describe the reflection and transmission as a function of geometry, surface roughness characteristics and wavelength and intensity of the incoming light.

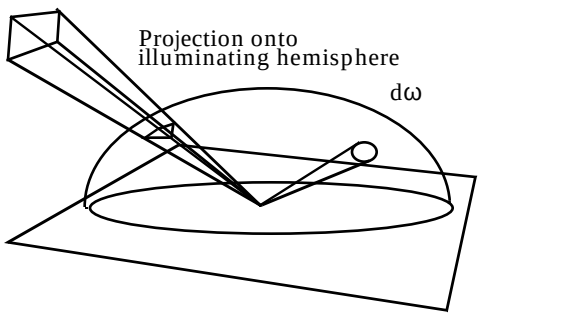


Fig. 3: The illuminating hemisphere.

Radiosity is inherently an energy balance formulation, so to start the derivation, we begin with the equilibrium at the surface relating the reflected and transmitted energy from the definition for flux; the flux from a surface is equal to the amount of flux either being reflected from the surface or transmitted through it:

$$\Phi_L = \int_{2\pi} I_R A \cos\theta_R d\omega_R + \int_{2\pi} I_T A \cos\theta_T d\omega_T \tag{8}$$

Because reflective and refractive relationships are independent of direction, this equation can be inverted so that the intensity transmitted in a certain direction V, is given as a function of incident intensity I_L as:

$$I_V = \int_{2\pi} I_L R_{bd} \cos\theta_R d\omega_L + \int_{2\pi} I_L T_{bd} \cos\theta_L d\omega_L \tag{9}$$

For opaque surfaces, there is no transmittance and for light sources, the emissive intensity, ε, must be added. Thus this equation becomes:

$$I_V = \varepsilon_V + \int_{2\pi} I_L R_{bd} \cos\theta_L d\omega_L \tag{10}$$

This equation is also known as the Rendering Equation [Kajiya86]. The physical interpretation of this equation is straightforward. The light that comes from any surface in the environment consists of the light which that surface itself emits, and the light which that surface reflects from all the other incoming lights in the environments.

In radiosity, we make the assumption that all the surfaces are diffuse and for diffuse surfaces, the bidirectional reflectance is constant and can be taken out of the integral resulting in:

$$I_V = \varepsilon_V + R_{bd} \int_{2\pi} I_L \cos\theta_L d\omega_L \tag{11}$$

From equations (6) & (7), and the fact that the integral of the projected angle over

the hemisphere is:

$$\int_{2\pi} \cos\theta d\omega = \pi \tag{12}$$

We get the following relations; intensity, emissive intensity and reflected energy all related to the radiosity:

$$I = \frac{B}{\pi}, \varepsilon = \frac{E}{\pi}, R_{bd} = \frac{\rho}{\pi} \tag{13}$$

Thus we get the general equation for radiosity as:

$$B_{dA} dA = E_{dA} dA + \rho_{dA} \int_{2\pi} \frac{B_L \cos\theta_L d\omega_L}{\pi} \tag{14}$$

In other words, as in the rendering equation, the radiosity for a particular surface is related to the amount of energy emitted from that surface, and the amount of energy that is reflected from the incoming energy in all directions. More commonly however, radiosity is expressed as a relation between the surfaces in an environment so we can express equation (14) as an integral over all the surfaces in the environment:

$$B_{dA_i} dA_i = E_{dA_i} dA_i + \rho_{dA_i} \int_j B_{dA_j} F_{dA_j - dA_i} dA_j \tag{15}$$

where $F_{dAj\text{-}dAi}$, called the Form-Factor from dA_j to dA_i, represents the fraction of energy leaving dA_j and arriving at dA_i. The Form-Factor will be discussed extensively in section III. We now need to recast equation (15) in a way that is computationally feasible. Since we can't manage to find the separate radiosity for each differential area, we begin by discretizing the environment into small patches of uniform radiosity. Turning the environment into discrete patches is discussed in section V. This turns the integral equation into a system of linear equations:

$$B_{A_i} A_i = E_{A_i} A_i + \rho_{A_i} \sum_j B_{A_j} F_{A_j - A_i} A_j \tag{16}$$

If one were to switch the roles of emitters and receivers, the fraction of energy emitted by one and received by the other would be identical to the fraction of energy going the other way. Thus, a reciprococity relationship exists between the form factors such that they are simply related by the ratio of their areas:

$$F_{A_i - A_j} A_i = F_{A_j - A_i} A_j \text{ or } F_{A_i - A_j} = F_{A_j - A_i} \frac{A_j}{A_i} \tag{17}$$

Dividing equation (16) through by A_i and incorporating equation (17) we get:

$$B_{A_i} = E_{A_i} + \rho_{A_i} \sum_j B_{A_j} F_{A_j - A_i} \tag{18}$$

Rearranging to matrix form and realizing that the Form Factor from a patch to itself is 0, gives us the following system of equations:

$$\begin{bmatrix} 1 & -\rho F_{1,2} & -\rho F_{1,3} & \dots & -\rho F_{1,N-1} & -\rho F_{1,N} \\ -\rho F_{2,1} & 1 & -\rho F_{2,3} & \dots & -\rho F_{2,N-1} & -\rho F_{2,N} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ -\rho F_{N-1,1} & -\rho F_{N-1,2} & -\rho F_{N-1,3} & \cdots & 1 & -\rho F_{N-1,N} \\ -\rho F_{N,1} & -\rho F_{N,2} & -\rho F_{N,3} & \cdots & -\rho F_{N,N-1} & 1 \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ . \\ . \\ B_{N-1} \\ B_N \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ . \\ . \\ E_{N-1} \\ E_N \end{bmatrix} \tag{19}$$

This matrix is always strictly diagonally dominant and can be solved by an iterative Gauss-Siedel solution which is guaranteed to converge. One note on the computational complexity of the radiosity solution. The order of the problem is order N for each form factor computation, thus the entire order for finding the full matrix for the radiosity solution is O(N2).

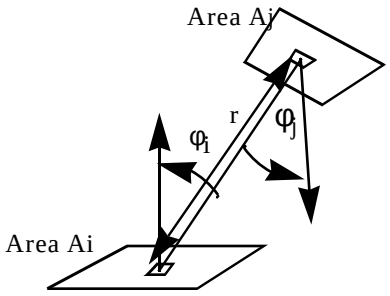


Fig. 4: Form-Factor Geometry

III. Form-Factor Calculation:

To find the form factor we must find the fractional contribution that a single patch makes upon another patch. This term is purely geometric, related only to the size, orientation, distance and visibility between the two patches. The basic geometry for form factor calculation is shown in Fig. 4. If we look at Fig. 5 we see that the area A is related to the projected area, A_p, by $A_p = A\cos\theta$, and the contribution of a projected area A_p is related to the solid angle by $\omega = \frac{A_p}{r^2}$.

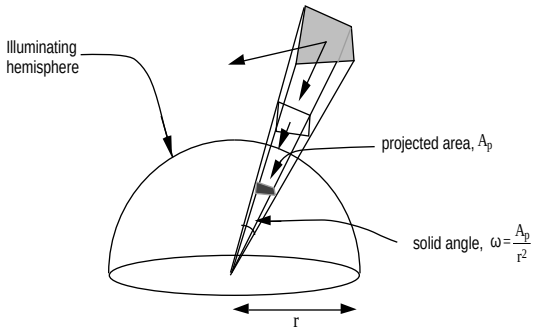


Fig. 5: Projected area onto hemisphere

The expression relating the contribution from one infinitesimal area to another is:

$$F_{dA_i - dA_j} = \frac{\cos\varphi_i \cos\varphi_j dA_j}{\pi r^2} \qquad (20)$$

The contribution from an infinitesimal area to a finite area is found by integrating over the receiving area:

$$F_{dA_i - dA_j} = \int_{A_j} \frac{\cos\varphi_i \cos\varphi_j dA_j}{\pi r^2} \qquad (21)$$

And from a finite patch to another finite patch, we take the area average of the previous equation:

$$F_{dA_i - dA_j} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{\cos\varphi_i \cos\varphi_j dA_j}{\pi r^2} \qquad (22)$$

Contour Integral Evaluation:

There are several different methods for evaluating this integral. In [Goral84], the contour integral is found by transforming the double integral with Stoke's Theorem:

$$F_{dA_i - dA_j} = \frac{1}{A_i} \int_{A_i} \int_{A_j} (\ln(r) dx_i dy_j + \ln(r)\, dy_i dx_j + \ln(r)\, dz_i dz_j) \qquad (23)$$

see appendix I for pseudo code for this evaluation. One limitation of this algorithm is that it doesn't take into account the visibility between one patch and another, another is that it is extremely expensive computationally. Baum et al. [Baum89] also use an analytical approach to find form factors. They integrate the outer integral numerically, while integrating the inner integral analytically by converting it into a contour integral. They then calculate the contour integral by piecewise summation, from Fig. 6.

$$F_{dA_j A_i} = \frac{1}{2\pi} \sum_{G_i} N_j \cdot \Gamma_g \qquad (24)$$

where: G_i is the set of edges in surface i.
N_j is the surface normal for the differential surface j.
Γ_g is a vector with magnitude equal to the angle gamma illustrated in Fig. 6and direction given by the cross product of vectors R_g and R_{g+1}.

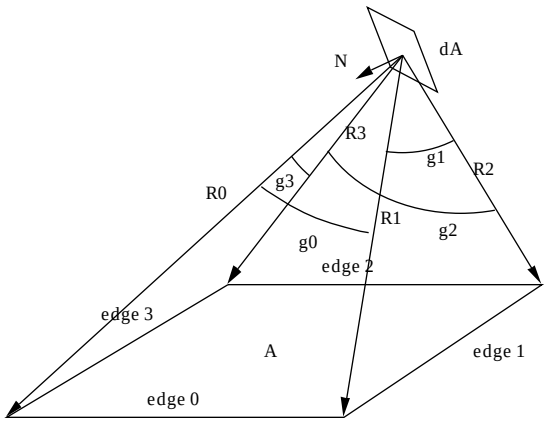


Fig. 6: Geometry for contour integral.

Baum et al. use this approach when evaluation by the more efficient hemi-cube method of evaluation described in the following sub-section is geometrically inappropriate for form factor evaluation. They also incorporate an extra term to account for the visibility between surfaces.

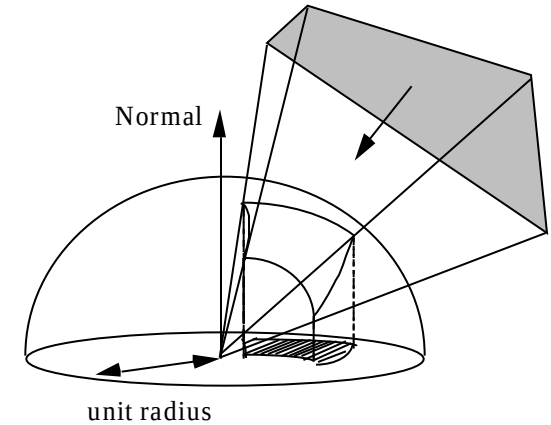


Fig. 7: Nusselt's Analog

Hemi-cube evaluation:

The hemi-cube approach for evaluating form factors was introduced by [Cohen85]. It is motivated by examining the geometry in Nusselt's Analog shown in Fig. 7. First, a patch is projected onto the hemisphere surrounding a patch. This projection accounts for the cosine of the angle between the normal of the projected patch and the hemisphere as well as one 1/r term. This projected patch is now projected onto the base of the hemisphere accounting for another 1/r and cosine term. The area at the base is equivalent to the form factor. Many areas are identical to the projected area on the hemisphere, and several of these lend themselves to calculation in a more straightforward fashion. See Fig. 8.

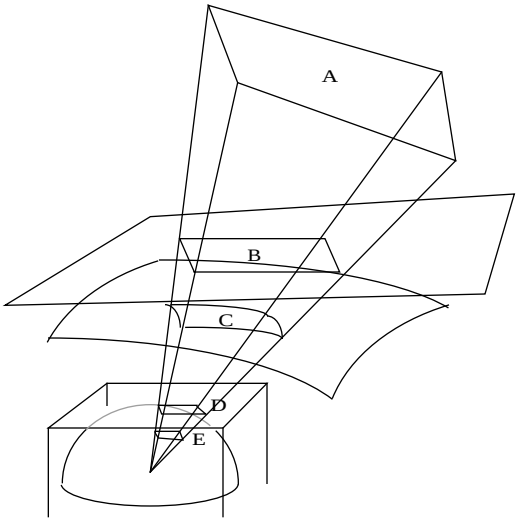


Fig. 8: A,B,C,D, & E all have same form factor.

Instead of projecting the environment onto a hemisphere, a cube is placed with the receiving patch at the center (see Fig. 9) and each surface of the cube is divided into a set number of pixels. The contribution of each pixel on the cube's surface to the form factor value can be precalculated since it is only dependent on the pixel location and orientation. The environment is then projected onto the faces of the cube (and half faces). Object visibility can be determined by using simple z-buffer techniques.

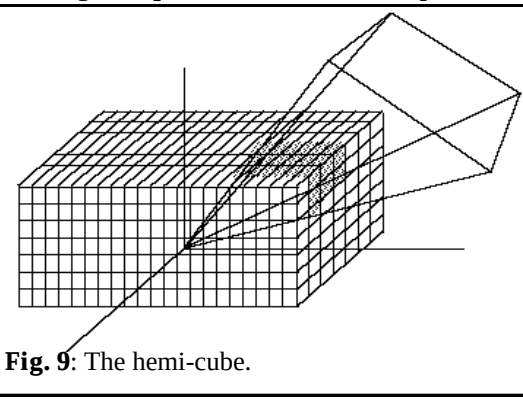


Fig. 9: The hemi-cube.

After determining which patch is visible at each pixel on the hemi-cube, the sum of all the delta form-factors for each pixel occupied by that patch determines the overall form-factor from the patch at the center of the cube. See Appendix II for details on the calculation of form factors via hemicubes.

Two primary problems exist with the hemi-cube formulation. Since only the integral over the sending patch is performed, inaccuracies can occur if the size of the patch is large relative to the distance between the patches. This problem can be alleviated by averaging the results of several hemicubes distributed over the receiving patch or by subdividing the environment into smaller patches or by using the contour integral method discussed above. A second problem exists because the hemi-cube samples the environment at regularly spaced solid angles, thus aliasing problems can occur. Small patches in the environment might never be seen while others may only be sampled by one or two hemi-cube grid cells. One solution to solving these problems is to use a different visibility algorithm; ray tracing.

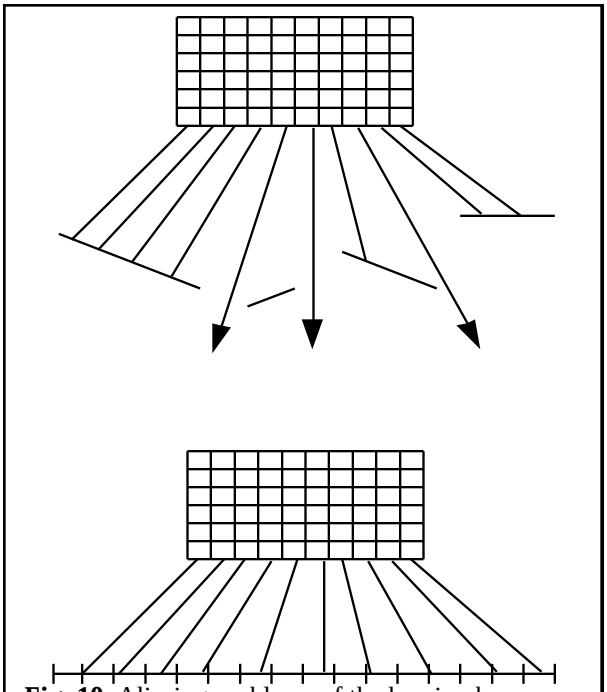


Fig. 10: Aliasing problems of the hemi-cube.

Raytracing for finding Form Factors:

Using ray tracing to find form factors has a number of attractions. First, since ray tracing can be done adaptively and stochastically, many of the aliasing problems in hemicubes can be overcome. Second, raytracing can handle a wide variety of geometries so that the objects which act as the "masks" need not be discretized into patches and the original geometries for the objects may be used. Although raytracing is not easily implemented in silicon, a number of acceleration techniques have been found for raytracing and in fact, for complicated scenes, algorithms have been found that actually perform better than the software z-buffers [Fujimoto86]. Finally, ray tracing techniques can play an important role for the inclusion of specular effects as will be discussed in section VI.

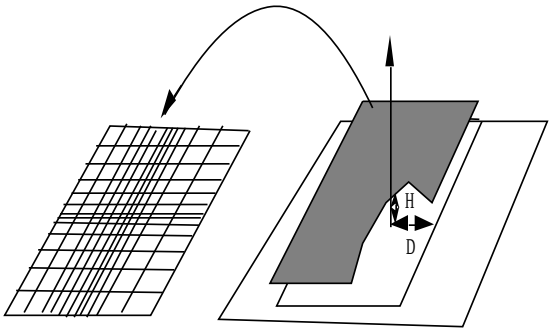


Fig. 11: Sillion's proxels for form factors

Sillion used a hybrid approach between hemi-cubes and ray tracing by placing a plane close to a receiving patch and sampling a rectangular grid on that plane centered about the patch. Some light is lost beyond the edges of the rectangular grid, but that amount can be made arbitrarily small by adjusting the distance of the plane from the patch. The grid is divided up into uneven elements (called proxels - projection elements) based on each elements contribution to the form factor and visibility is calculated by Warnock's algorithm. Once the patch visibility is calculated from each window, the

contribution number of proxels is summed and the form factor is calculated. Raytracing is then used on the corners of the window to verify that it has been subdivided properly, and for the generation of extended form factors which will be discussed in a sectionVI. [Sillion 89].

One straightforward approach in using raytracing (actually ray casting since only object visibility information is used), is to sample the hemisphere around a patch uniformly in all directions. However doing so wastes effort by sampling areas that do not contribute a great deal. Directions close to the normal will have a large effect while those at the periphery a smaller effect. Malley uses this method by jittering a number of samples in a distribution about the normal [Malley 88].

Another way of using ray tracing for form factor calculation was used by Wallace in [Wallace88]. Here, instead of computing the entire environments contribution to a single patch, the computation is turned around and the contribution of a single patch to every differential area in the environment is found. This is particularly suited to the progressive radiosity method discussed in section IV. Essentially a patch is divided into a number of smaller regions ΔA's and each region is sampled by a ray from some point in the environment. In other words, equation (13) is discretized and calculated for each subregion:

$$F_{dA_i - A_j} = \sum_{k=1}^{n} \frac{\cos\varphi_{ik}\cos\varphi_{jk}}{\pi r^2}\Delta A_j \qquad (25)$$

This however still does not overcome the problem when ΔA is large relative to r^2. To do so, one can discretize the environment into ever smaller patches, or treat the ΔA's as finite areas and use an analytical expression between a differential area and a finite area to sum the form factors.

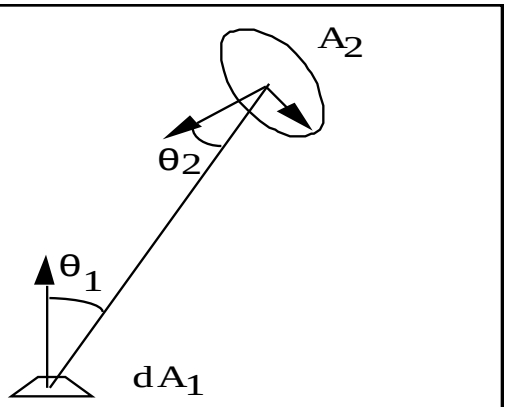


Fig. 12: Differential area to disk form-factor.

If each ΔA is considered to be a small disk of area A_2 the form factor from a differential area to a directly opposing disk of area A_2 is:

$$F_{dA_i - A_j} = \frac{A_2}{\pi r^2 + A_2} \qquad (26)$$

Taking into account the relative orientations of the source and receiving areas makes:

$$F_{dA_i - A_j} = \frac{A_2\cos\varphi_i\cos\varphi_j}{\pi r^2 + A_2} \qquad (27)$$

and finally if a patch of area A2 is subdivided into n delta areas:

$$F_{dA_i - A_j} = \frac{A_2}{n}\sum_{k=1}^{n} \frac{\cos\varphi_{ik}\cos\varphi_{jk}}{\pi r^2 + A_2} \qquad (28)$$

IV. Progressive Radiosity:

Harkening back to equation (19) we already discussed how the system of equations can be solved iteratively by a Gauss-Siedel iteration method. In a landmark paper, Cohen et al.[Cohen88] suggested a faster approach to solving the radiosity

equation. Solving the matrix by Gauss-Siedel, converges to the correct solution one row at a time. This is equivalent to saying that the i^{th} row of the equations provides an estimate of the radiosity of patch i based on the current estimates of all the other patches.

$$B_i = E_i + \rho_i \sum_{j=1}^{n} B_j F_{ij} \tag{29}$$

This is commonly interpreted as gathering in the light from the rest of the environment at a single patch. In a similar fashion, the solution can be arrived at by distributing the light from a single patch to all the other patches in the environment. This can continue until all the energy in the environment is redistributed.

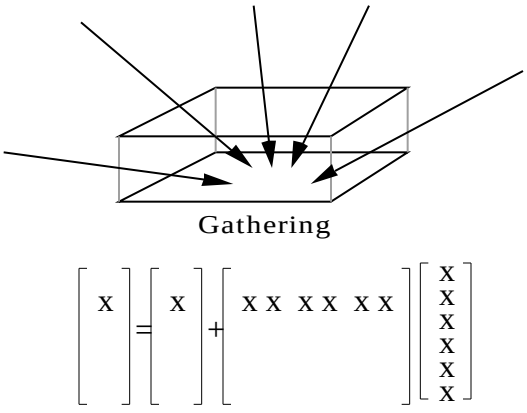


Fig. 13: Gathering

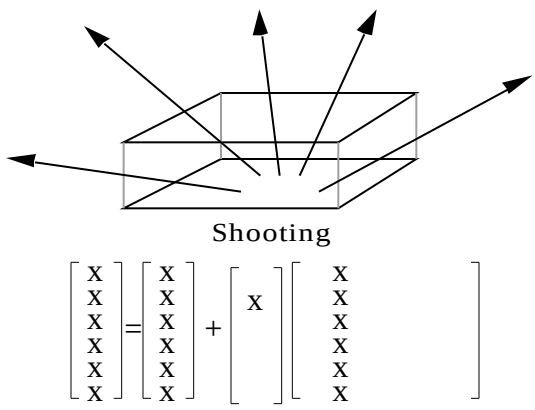


Fig. 14: Shooting

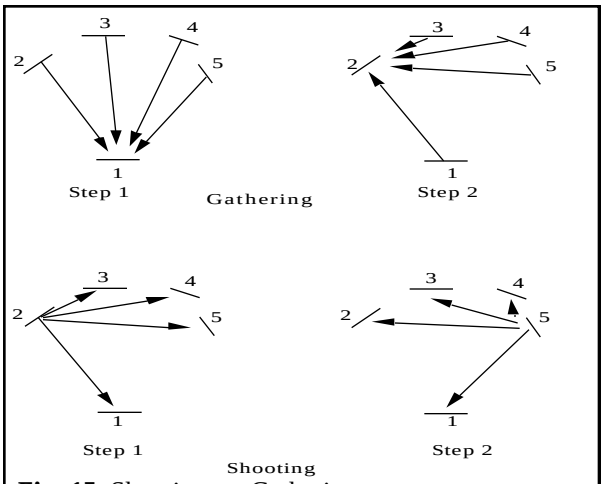


Fig. 15: Shooting vs. Gathering

Mathematically, shooting is equivalent to the following:

$$B_j = B_j + B_i(\rho_j F_{ji}) \tag{30}$$

where $F_{ji} = F_{ij}\frac{A_i}{A_j}$ as by reciprococity (equation (17)). These two interpretations are shown in Fig. 13 and Fig. 14. If the brightest patch in the environment is shot first, the solution converges very rapidly to the final solution. An inexact solution is often good enough so that the most lengthy part of the computation (the calculation of the form factors) need only be performed for a limited number of shoots. To display the results intermediately, an ambient term must be calculated to approximate the total lighting in the environment. To do so, the form factor is approximated from each element to another as a simple area weighted ratio between the area of the patch and the area of the entire environment:

$$F_{*j} = \frac{A_j}{\sum_{j=1}^{n} A_j} \tag{31}$$

and an average reflectivity for the environment:

$$\rho_{ave} = \frac{\sum_{i=1}^{n} \rho_i A_i}{\sum_{i=1}^{n} A_i} \tag{32}$$

Thus the overall intereflection factor is simply a geometric sum:

$$R = 1 + \rho_{ave} + \rho_{ave}^2 + \quad = \frac{1}{1-\rho_{ave}} \tag{33}$$

and the overall ambient term is the total energy which has not yet been shot via form factor computation times the reflection factor R:

$$\text{Ambient} = R \sum_{j=1}^{n} \Delta B_j F_{*j} \tag{34}$$

So, at rendering time, the radiosity for any patch is augmented by the ambient term times the reflectivity for the patch (only for display purposes):

$$B'_i = B_i + \rho_i (\text{Ambient}) \tag{35}$$

Sorting the patches so as to always shoot the brightest patch also increases convergence. Since the solution converges much more rapidly to the real solution for the environment, every patch need not shoot for the overall solution to be within acceptable tolerance from a final result. Thus all the form factors from every patch to every other patch need not be computed and the computational order of the radiosity solution is no longer O(N2); it is nearly O(N). Convergence times for gathering, shooting, shooting with sorting, and shooting, sorting, & ambient light are shown in Fig. 16.

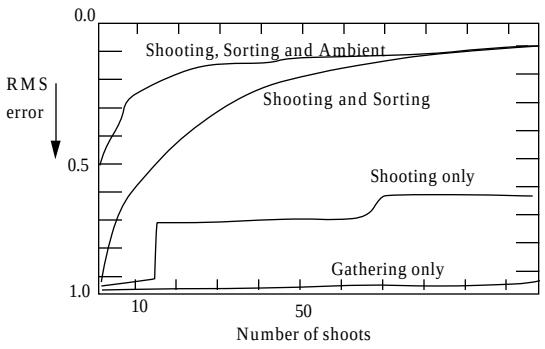


Fig. 16: Convergence Times [Cohen88]

V. Discretizing the Environment:

One major concern in radiosity research is the subdivision of the environment into appropriate sized patches. If patches are too large, then the radiosity solution is not only inaccurate, but patches that lie along large radiosity changes are inaccurately represented. If patches are too small however, the computational burden imposed by the $O(N^2)$ problem becomes overwhelming. One solution to this problem is "substructuring" [Cohen88]. Here, patches are subdivided into elements, which receive radiosity for a more accurate solution. However elements do not send out radiosity, the radiosities of all the elements that make up a patch are summed together to compute the radiosity of the patch that will then be distributed to the rest of the environment. In this fashion, hemicubes (or other methods such as raytracing) can be placed on the sending patch and form factors computed to all other elements but this calculation only needs to be performed at the patches, and not at the elements. This patch-element subdivision can be performed before any radiosity calculation is done, or it can be done adaptively based on high radiosity gradients during the solution. Whenever a patch's radiosity differs by too much at the vertices, that patch can be subdivided into elements and the radiosity can be reshot. This will result in

substructuring as pictured in Fig. 17.

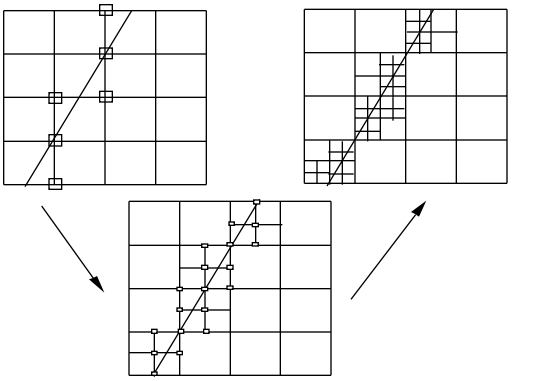


Fig. 17: Adaptive subdivision

Adaptive subdivision still suffers from several drawbacks. One problem is that the patches must initially be small enough for the high radiosity gradients to exist in the first place. If a patch is too large, a small object casting a shadow will not be seen at all at the patches, thus no substructuring will take effect. Also, since emitting surfaces are not subdivided (in other words, patches are point sampled), soft shadows, or penumbras, are not seen. Another method generates patch-element subdivisions based on shadow boundaries [Campbell90]. The shadow boundaries are computed based on BSP trees as described in [Chin89]. In their work, form factor calculation was performed by the raytracing method, but visibility was calculated with the BSP tree structure that was already formed for the shadow boundaries.

VI. Incorporation of Specular Effects:

The radiosity calculations described in the preceding sections have all been for solely diffuse environments. Several extensions have been suggested for the radiosity method to include specular effects, some still retaining view independence, while other add a view dependent step.

View Independent Approaches:

Immel et al. [Immel86] suggested extending radiosity by including directional information at each patch besides just diffuse illumination. The algorithm proceeds by propagating the incoming light outwards in the corresponding outgoing direction. Since each patch retains all outgoing information, the final viewpoint can incorporate all specular information for every patch independent of where that view is. The pseudocode for the algorithm is given in appendix IV. The major problem with the algorithm is that enormous amounts of information must be stored and computed. A finite number of directions must be chosen to be retained for each patch, but that number must be extremely large for specular objects since they may receive and reflect light along any direction. The imposing requirements for storage and computation make this algorithm unfeasible at the present time, but may be intriguing in the future for large scale parallel implementations.

Shao et al [Shao88] take a similar but less computationally intensive approach. Essentially, they modify the concept of the form factor to include not only geometric information between two patches, but general radiant information as well. If a patch is a specular patch, the form factor $F_{1,2}$ between patches 1 and 2 in Fig. 18 will be dependent on the amount of light being received from patch 3. The algorithm starts by assuming a diffuse approximation to the solution using a typical radiosity solution. This solution is then used to iteratively adjust all the form factors in the solution so that non-diffuse patches form factors with other patches in the environment account for the incoming and outgoing information. Although this method produces directional information from each pass, the authors had to add a final rendering of the scene to capture

quickly changing specular patches.

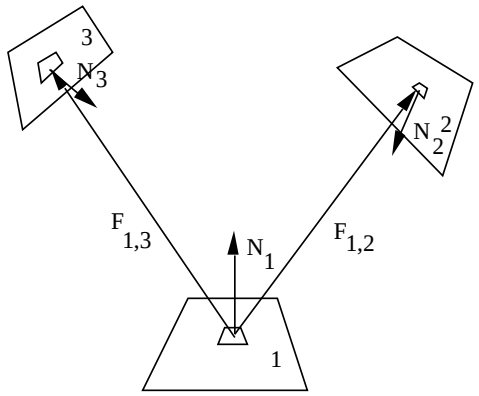


Fig. 18: Form Factors for Shao et al [Shao88]

Two pass methods:

Several researchers have suggested combining the strengths of two different types of calculations; raytracing and radiosity. Raytracing is well suited to specular environments since "importance" sampling in the reflected direction is possible [Kajiya86] (that is, to send the most rays in the direction which will have the greatest affect on the perceived intensity at a pixel). Radiosity, as we have seen above, is well suited to diffuse environments. Extensions have been made to each method that try to include aspects of the other. Kajiya used "path-tracing" to stochastically sample paths through the environment, but an imposing number of rays must be shot to attempt to calculate the effects of diffuse transmission. Immel et al [Immel86] (as discussed previously) extended the radiosity method to include information about specular transmission and reflection, but specular surfaces often needed to be subdivided to a level that make computation infeasible. Essentially, two pass methods use radiosity as a first pass to compute the effects of diffuse light, and a post process pass of raytracing or z-buffering adds specular contributions. The details of this method are described below in the following section.

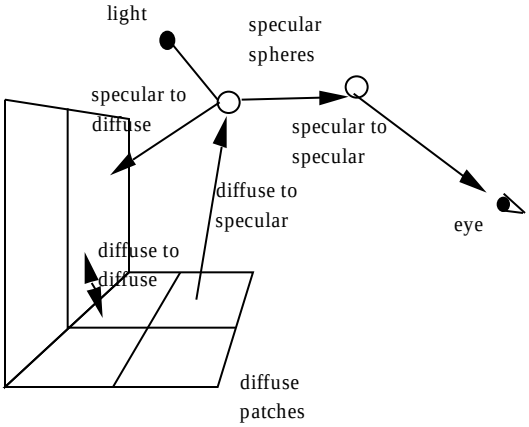


Fig. 19: Transport mechanisms for light

Wallace et. al [Wallace88] characterized the "transport" of light via four mechanisms; (1) diffuse to diffuse, (2) specular to diffuse, (3) diffuse to specular, and (4) specular to specular (shown in Fig. 19). Radiosity will account for only the first transport mechanism, diffuse to diffuse. A post process step can be added which traces rays from the eye point into the scene, recursively following specular objects while stopping as soon as a diffuse object is hit. This however does not account for specular to diffuse transport. To do this, the basic formulation for radiosity must extended to account for this mode of transport.

Rushmeier and Torrance [Rushmeier91] outline the method for extending the calculation of form factors to include diffuse transmission, specular transmission and specular reflectance. Diffuse transmission is accomplished by the simple expedient of calculating the form factors on the hemisphere at the back of the surface (they placed a hemi-cube centered at the patch but facing in the direction of transmission. Specular transmission was calculated by considering the specularly transmitting patch as a window into the environment through which additional form factors are calculated (see figure Fig.

20).

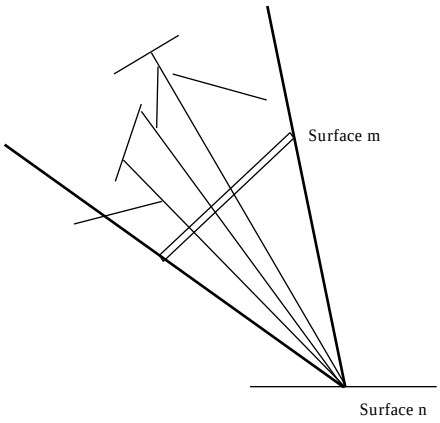


Fig. 20: Specular Transmitting Form-Factors (formfactor at n from specular surface m is due to all surfaces seen at n through m)

Finally, both Wallace and Rushmeier et al. treat specularly reflecting surfaces as ideal mirrors through which a duplicate of the environment is visible. Form factors are then calculated in a similar fashion as through the specularly transmissive factors. Sillion and Puech [Sillion89] extended the concept of form factors by considering them not just as the direct contribution between one patch to another, but the proportion of energy leaving patch i and reaching patch j via any number of specular reflections or refractions. This is similar to the Wallace or Rushmeier models, but any specular surface, not just mirrors may be used because they use ray tracing to calculate the form factors.

Raytracing the resulting environment is quite a bit more efficient than typical raytracing since no shadow rays are present (generally accelerating the process and making it independent of the number of lights in the scene), and the shading model is trivial; simply the value of the radiosity interpolated to the intersection point. Finally, specular surfaces do not need to be as finely subdivided as in Shao's or Immel's methods.

VII. Dynamic Environments:

Once the computationally intensive form factor computations have been performed, and the solution is solved by whatever method, the scene can be rendered from any point of view by simply redisplaying. However the moment the geometry within the environment changes, the radiosity solution is no longer accurate and new calculations must be performed. Since the calculation of the entire scene is extremely computationally intensive, it would be beneficial to leverage the results of the previously calculated solution to generate a new solution. Two methods that have been introduced in the literature to deal with this problem are those of Baum et al [Baum86] and of Chen [Chen90] or closely related to Chen's method, that of [George90].

Back-Buffer Algorithm:

Baum proposed the first method in [Baum86]. This method requires that the movement of any objects in the scene is known before hand. The first part of the algorithm is to make a generalized volume through which the dynamic objects travel in time. Then, all the static objects calculate their form factors with all the other static objects in the environment. Objects that are occluded by the swept volume from a certain patch's point of view are stored in a structure called a "back-buffer".

The initial solution is calculated completely. When it comes time to update the solution, only the form factors between static objects that are behind the swept volume need to be recalculated based on the current position of the moving object. In addition, only the form factors between the dynamic object and the static patches in the environment that can "see" the swept volume need to be calculated. In this fashion, most of the previously calculated form

factors can be reused and the solution can proceed to the iterative solution stage.

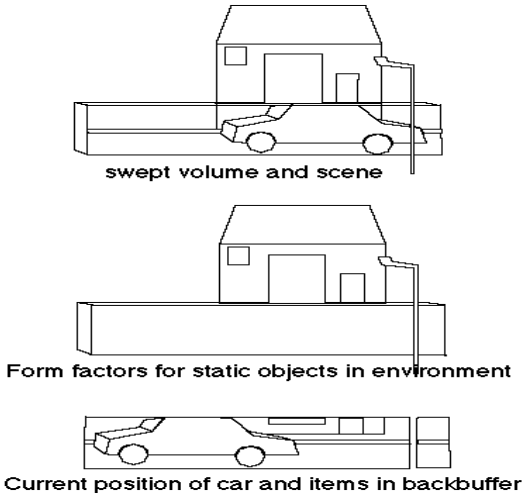


Fig. 21: The Back Buffer

Incremental radiosity & redistribution:

Not all the motion in an environment might be known beforehand. Two essentially identical methods for dynamic environments proposed by [Chen90] and [George90] involve redistribution of the changed radiosity throughout the environment. This redistribution has been dubbed incremental radiosity by Chen. The incremental radiosity method is essentially an extension of the progressive radiosity method discussed in section IV.. Instead of just shooting radiosity energy throughout the environment, however, corrections in the radiosity are redistributed into the environment as well. When the attributes of a patch are changed, the patch's incremental radiosity (the difference in radiosity as a result of the attribute change) is computed first. This incremental radiosity is then distributed to the rest of the environment. For example, if a light in a scene is suddenly turned off, the resulting incremental radiosity will be negative, and this negative radiosity will be distributed throughout the scene, making each surface that it reached before correspondingly darker. Eventually, all the changes will propagate through the scene for the proper solution. The mathematical derivation is as follows (from [Chen90]). Remembering equation (18), the old radiosity for a patch is:

$$B_i = E_i + \rho_i \sum_j B_j F_{j-i} \tag{36}$$

if we let E_i' and ρ_i' be the new emission and reflectance of patch i. The new radiosity is:

$$B'_i = E'_i + \rho'_i \sum_j B_j F_{j-i} \tag{37}$$

and the incremental radiosity is simply their difference:

$$\Delta B_i = E'_i - E_i + (\rho'_i - \rho_i) \sum_j B_j F_{j-i} \tag{38}$$

or incorporating equation (36) again:

$$\Delta B_i = E'_i - E_i + \frac{(\rho'_i - \rho_i)\ (B_i - E_i)}{\rho_i} \tag{39}$$

Once this incremental radiosity is computed, it is added to the radiosity and unshot radiosity of the patch. The progressive radiosity can proceed as before. If an object in the environment moves, the form factors between many of the objects in the environment must be recalculated, though fortunately, object coherence may be used to limit this computation, see Fig. 22.

Here the new radiosity is based on a change in the form factors:

$$B''_j = B_j + \rho_j (B_i^s \Delta F_{ji} + B_i^u F'_{ji}) \tag{40}$$

What this means is that all the previously shot information must be reshot based on the change in the form factors ΔF_{ji} and previously shot radiosity. Thus

patches that previously saw light from a certain patch now must receive negative radiosity from that patch combined with the unshot radiosity from other patches based on its current form factor. Care must be taken to either reshoot all the previously shot patches before any new movements, or to keep track of which patches have shot depending on which geometry. Fig. 23 shows the dangers of adding an object, and then removing it before all patches reshot their radiosity based on a change in the patch form factors. A hole is created by light shooting negative radiosity to a place that is no longer in shadow.

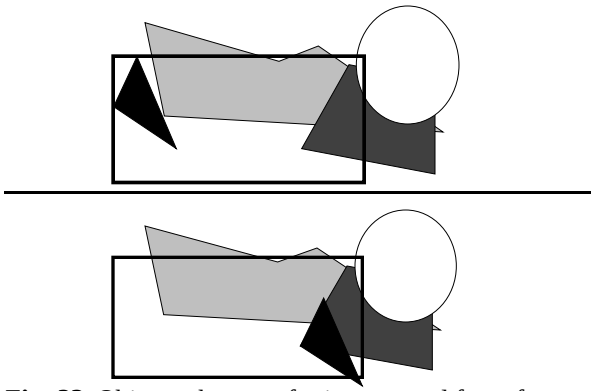


Fig. 22: Object coherence for incremental form-factors

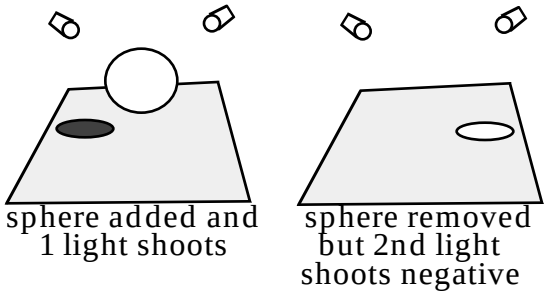


Fig. 23: incremental radiosity dangers.

This general technique for adding and subtracting radiosity based on changes in the environment shows great promise for eventually generating real time interactive radiosity based pictures. Another method that might produce real time radiosity solution is the incorporation of parallel processing.

VIII. Parallel Processing:

The radiosity method lends itself to parallelization quite conveniently and several researchers have explored techniques to accomplish this. The primary method, utilized by Puech et al. [Puech90], Recker et al. [Recker90], and Baum et al[Baum90] distribute the calculation of form factors to many different processors which each contain a copy of the entire database. One server process maintains contact with all the clients and sends patches to each client to compute the form factors. The server than receives updates from the clients and in turn updates a display. Bottlenecks can result when the server cannot communicate with more than a certain amount of clients without the clients waiting for another patch to calculate.

Baum et al take a slightly different approach to parallelizing the radiosity algorithm. Since the hemi-cube algorithm is essentially a rasterization problem, they use the graphics hardware already present in the SGI GTX platform. Using the already parallel geometry pipeline they get improvements of over 40 times the performance of non-dedicated platforms. They distribute as parallel tasks the conversion of the hemicube item buffers into form factors and the redistribution of light into the environment. Shared memory is used to retain information about the radiosities of all the patches in the environment. Bottlenecks will still result when the geometry pipeline is completely full and clients need to wait for information to process. They also use a double buffered approach to reduce idle time for an unfilled geometry pipeline while the item buffers are being processed. The pseudocode for this algo-

rithm is shown in appendix IV.

Other researchers have designed VLSI architectures to achieve a high speed solution. The architecture proposed by Bu et al [Bu89] divides the system into two parts, the first subsystem calculates the form factors while the second subsystem implements a parallel Gauss Siedel iterative solver. The calculation of form factors is done by parallel raytracing, each form factor being computed independently. This concept can also be implemented on more general SIMD parallel processing machines such as the Connection Machine.

Drucker has implemented a radiosity algorithm on the Connection Machine [Drucker91]. The algorithm is based on the calculation of form factors by ray tracing. Essentially, all the form factors to a single shooting patch can be computed at the same time. Each vertex calculates a new radiosity value based on the calculated form factor and then each patch to which this vertex belongs is updated simultaneously. Since parallel processing is used, there is no speedup for the incorporation of more elements to which the radiosity is computed. Currently the algorithm uses a simplistic serial based ray tracing algorithm at the core process, thus the system is O(N) in computing form factors. More sophisticated raytracing algorithms are currently being explored. Preliminary performance results are extremely encouraging in comparison with other methods of radiosity calculation.

IX. Conclusions:

Radiosity techniques have a number of advantages over other forms of rendering. Greater realism, view independence, accurate simulation of physical phenomena, and now, competing speed versus other methods of global illumination (ray tracing). Realism, though important in several applications, (such as architectural visualization) serves little benefit if the user can not interact conveniently with the results. Because of the view independence of conventional radiosity formulations, (and growing speed at which radiosity solutions can be achieved), interactive radiosity is almost a reality. Coarse grain parallelism has already brought this technique to the threshold of practicality and fine-grain parallelism holds promising possibilities for even greater improvements. The pursuit for greater and greater realism is perhaps always an elusive goal to which some feel too much attention has already been devoted. “Rendering is a solved problem” some feel, with faster machines just making more complex environments possible. Yet radiosity as an algorithm did not exist in the computer graphics literature 6 years ago, and the pace at which innovations and modifications to the algorithm have taken it from a mere curiosity to perhaps the best way to simulate interior environments, is astonishing. Photorealistic pictures computed at interactive update rates is not yet a reality, but the results in radiosity research show that improvement is not only in the domain of faster hardware but also basic new algorithms.

Radiosity has the added benefit of accurately simulating the physical situation of the environment. Extensions might be made to deal with other physical phenomena based on the techniques that have only now become clear, already some papers are attempting to model sound using formulations similar to radiosity. The techniques used to solve one problem may have great effects on other problems as well.

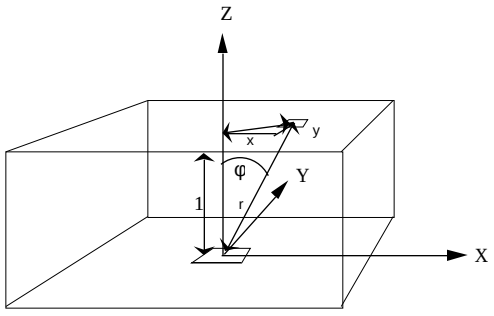
Finally, I feel that research into radiosity can be a motivating force in examining the desirability of certain architectures. Already the SLALOM benchmark uses the

radiosity problem as a standard at which to rate several machines with completely different architectures and capabilities. Examining how course and fine grain parallelism can be used to solve the radiosity problem has lead to some rethinking about the benefits or drawbacks that are inherent in SIMD architectures over MIMD ones.

Appendix I: Pseudocode for Contour Evaluation of Form Factor:

```
FF := 0
for each segment of the perimeter of
element i:
{
    for each segment of the perimeter
     of element j:
    {
       evaluate the distance between
        the segments;
       take the natural log of the
        distance;
       evaluate the lengths of the
        segments along each axis (dxi,
        dxj, dyi, dyj, dzi, dzj);
       multiply the natural log by
        (dxidxj + dyidyj + dzidzj);
       add the result to FF;
    }
}
divide FF by 2π times the area of
element i;
```

Appendix II: Hemi-cube calculation of Form Factors:



$$r = \sqrt{(x^2 + y^2 + 1)}$$

$$\cos\varphi_i = \cos\varphi_j$$

$$\cos\varphi = \frac{1}{\sqrt{(x^2 + y^2 + 1)}}$$

$$\Delta\text{Form-Factor} = \frac{\cos\varphi_i \cos\varphi_j}{\pi r^2}\Delta A$$

Fig. 24: Hemi-cube calculations

ΔForm-Factor is the delta form-factor associate with pixel q on the hemi-cube

R is the number of hemi-cube pixels covered by projection of patch j onto the hemi-cube centered at patch i.

Appendix III: Progressive Refinement Pseudocode

```
for each iteration, for patch i:
{
    calculate the form-factors Fij
     using a hemi-cube at patch i;
    for each patch j:
    {
       ΔRad = ρΔBiFijAi/Aj;
       /* update change since last time
        patch j shot light */
       ΔBj = ΔBj + ΔRad;
       /* update total radiosity of
        patch j */
       Bj = Bj + ΔRad;
       /* reset unshot radiosity for
        patch i to zero */
       ΔBi=0.;
    }
}
```

Appendix IV: Pseudocode for Non-diffuse Radiosity environments:

```
/* Initial estimate for directional
intensities */

for each direction d and patch p:
    intensity[p][d] = emission[p][d];
end

for each iteration:
    for each patch p:
       for each direction:
            newintensity[p][d] =
             emission[p][d]
       end
       for each direction in:
           energy[in] =
             intensity[Cube[p][in]]
             [d-in] cosθω[in]
           for each direction out:
              newintensity[p][d] +=
```

```
                r''(p, in, out) x
                Energy[in]
            end
        end
    /* check for convergence and
     replace old intensity with new*/
     for each direction d:
      compare intensity[p][d] with
       newintensity[p][d]
      intensity[p][d] =
       newintensity[p][d]
    end
end
```

Appendix V: Pseudo for Parallel Implementation (Baum et al):

```
/* Initial estimate for directional
intensities */
Producer()
{
    /* find first patch to shoot */
    until (done) do:

    /* use graphics hardware to compute
     hemi-cubes */
     render_polys_to_hemi_cube();
    /* give rest of computation
     (including form factors and
     distribution of light to consumer
     process */
    /* must wait while consumer is busy
     */
    while (Consumer_Buff == FULL)
     wait;
    /* free up the consumer task to
     process item buffers */
    Set_state(consumer_buff) = FULL;
    /* get next patch to shoot (from
     consumer) */
    patch =
     get_shooting_patch(consumer_buff
     );
    end
}

Consumer()
{
    until(done) {
      /* wait until item buffer is
       present in shared memory */
      while(get_buf_state(consumer_bu
       f) == EMPTY) wait;
      /* find out which patch was shot
       for current item buf */
      get_shooting_patch(consumer_buf
       );
      /* fork computation off to
       several sub processors */
      m_fork(Compute_Form_Factors);
      /* for out distribution of light
       energy to env */
      m_form(Shoot_Light_Energy);
      /* find next patch to shoot */
      next_to_shoot=(Unselected
       patch w/highest delta energy);
      /* make this patch the next one
       selected by the producer*/
      set_shooting_patch(consumer_buf
       f, next_to_shoot);
      set_buf_state(consumer_buf) =
       EMPTY;
    }
}
```

X. Bibliography:

[Baum86]Baum, D.R., Wallace, J.R., Cohen, M.F., and Greenberg, D.P. The back-buffer algorithm: an extension of the radiosity method to dynamic environments. The Visual Computer. 2(5):298-306, June 1986.

[Baum89]Baum, D.R., Rushmeier, H.E., Winget, J.M. Improving radiosity solutions through the use of analytically determined form-factors. Computer Graphics, 23(3):325-334, July 1989. Proceedings SIGGRAPH 1989 in Boston.

[Baum90]Baum, D.R. Winget, J.M. Real Time Radiosity through Parallel Processing and Hardware Acceleration. Computer Graphics (Special Issue on 1990 Symposium on Interactive 3D Computer Graphics) 24(2):67-75, March, 1990.

[Bu89]Bu, J. Deprettere, E.F, A VLSI system architecture for high-speed radiative transfer 3D Image synthesis. The Visual Computer 5:121-133. 1989.

[Campbell90]Campbell, A.T., Fussel, D.S., Adaptive mesh generation for global diffuse illumination, 24(4):155-164, August 1990. Proceedings SIGGRAPH 1990 in Dallas.

[Chen90]Chen, S.E. Incremental Radiosity: An extension of Progressive Radiosity to an Interactive Image Synthesis System. Computer Graphics, 24(4):135-144, August 1990. Proceedings SIGGRAPH 1990 in Dallas.

[Chin89]Chin, N., Feiner, S. Near Real-time shadow generation using BSP trees, 23(3):31-40, July 1989. Proceedings SIGGRAPH 1989 in Boston.

[Cohen85] Cohen, M.F, Greenberg, D.F. The Hemi-cube: A radiosity solution for complex environments. Computer Graphics, 19(3):31-40, July 1985. Proceedings SIGGRAPH 1985 in San Francisco.

[Cohen86] Cohen, M.F., Greenberg, D.F., Immel, D.S., Brock, P.J. An efficient radiosity approach for realistic image synthesis. IEEE C.G & A, 6(2):26-35 January, 1986.

[Cohen88]Cohen, M.F., Chen, S.E., Wallace, J.R., Greenberg, D.P. A progressive refinement approach to fast radiosity image generation, Computer Graphics, 22(4):75-84, July 1988. Proceedings SIGGRAPH 1988 in Atlanta.

[Drucker91]Drucker, S.M. Radiosity on the Connection Machine System ™, In Progress.

[Fujimoto86]Fujiomot, A, Tanaka, T., Iwata, K., ARTS: Accelerated Ray Tracing System. IEEE C.G. & A. 6(4):16-26.April 1986.

[Goral84] Goral, C.M, Torrance, K.E., Greenberg, D.P., Battaile, B., Modeling the interaction of light between diffuse surfaces. Computer Graphics, 18(3):213-222, July 1984. Proceedings SIGGRAPH 1984 in Boston.

[Greenberg86]Greenberg, D.P. Cohen, M.F., Torrance, K.E., Radiosity: a method for computing global illumination. The Visual Computer (1986). 2:291-297.

[George90]George, D.W. Sillion, F.X., Greenberg, D.P. Radiosity redistribution for dynamic environments. Course Notes on Radiosity, Tutorial at SIGGRAPH 1990. Dallas.

[Hall86]Hall, R, A characterization of illumination models and shading techniques, The Visual Computer (1986) 2: 268-277.

[Hall89]Hall, R, Illumination and Color in Computer Generated Imagery. Springer Verlag, New York, 1989.

[Immel86]Immel, D.S., Cohen, M.F., Greenberg, D.P. A Radiosity Method for Non-Diffuse environments, Computer Graphics, 20(4):133-142, August 1986. Proceedings SIGGRAPH 1986 in Dallas.

[Kajiya86]Kajiya, J.T.1986. The Rendering Equation, Computer Graphics, 20(4):143-150, July 1986. Proceedings SIGGRAPH 1986 in Boston.

[Kajiya1990]Kajiya, J.T. 1990. Radiometry and Photometry for Computer Graphics. SIGGRAPH '90 Course Notes on Advanced Techniques for Raytracing.

[Recker90]Recker, R.J., George, D.W. Greenberg, D.P. Acceleration Techniques for progressive refinement radiosity, Computer Graphics (Special Issue on 1990 Symposium on Interactive 3D Computer Graphics) 24(2):59-64, March, 1990.

[Rushmeier90]Rushmeier, H. Torrance, K.E., Extending the Radiosity Method to Include Specularly Reflecting and Translucent Materials. Transactions on Computer Graphics. 1990.

[Sillion89]Sillion, F., Puech, C., A General Two-Pass Method Integration Specular and Diffuse Reflection. Computer Graphics, 23(3):335-344, July 1989. Proceedings SIGGRAPH 1989 in Boston.

[Shao88]Shao, M.Z., Peng,Q.S, Lian, Y.D, A new radiosity approach by procedural refinements for realistic image synthesis, Computer Graphics, 22(4):93-101, July 1988. Proceedings SIGGRAPH 1988 in Atlanta.

[Wallace87]Wallace, J.R., Cohen, M.F., Greenberg, D.P., A two pass solution to the rendering equation: A synthesis of ray tracing and radiosity methods. Computer Graphics, 21(4):311-320, July 1987. Proceedings SIGGRAPH 1987 in San Anaheim.

[Wallace89]Wallace, J.R., Elmquist, K.A., Haines, E.A. A raytracing algorithm for progressive radiosity, Computer Graphics, 23(3):315-324, July 1989. Proceedings SIGGRAPH 1989 in Boston.