On Convergence and Complexity of Radiosity Algorithms

László Szirmay-Kalos and Gábor Márton

Department of Process Control, Technical University of Budapest,

Budapest, Műegyetem rkp. 11, H-1111, HUNGARY

szirmay@fsz.bme.hu

ABSTRACT: This paper evaluates and compares the convergence and complexity characteristics of radiosity algorithms with a special emphasis on randomized methods.

Key Words: Radiosity method, complexity evaluation, randomized algorithms

Introduction

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 that these patches are small and they have only diffuse reflection, their intensity distribution over the surface can be approximated by a constant value. 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.

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 $(E_i \cdot dA_i)$ and the diffuse reflection of the radiance coming from other surfaces.

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:

$$B_i \cdot dA_i = E_i \cdot dA_i + \varrho_i \cdot \int B_j \cdot F_{ji} \cdot dA_j \tag{1}$$

Taking advantage of the homogeneous property of the surface patches, the integral can be replaced by a finite sum:

$$B_i \cdot \Delta A_i = E_i \cdot \Delta A_i + \varrho_i \cdot \sum_j B_j \cdot F_{ji} \cdot \Delta A_j \tag{2}$$

Using the reciprocity relationship stating that $F_{ji} \cdot \Delta A_j = F_{ij} \cdot \Delta A_i$, we get:

$$B_i = E_i + \varrho_i \cdot \sum_j B_j \cdot F_{ij} \tag{3}$$

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

$$\begin{bmatrix} 1-\varrho_1 F_{11} & -\varrho_1 F_{12} & \dots & -\varrho_1 F_{1N} \\ -\varrho_2 F_{21} & 1-\varrho_2 F_{22} & \dots & -\varrho_2 F_{2N} \\ \vdots & & & \\ -\varrho_N F_{N1} & -\varrho_N F_{N2} & \dots & 1-\varrho_N F_{NN} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_N \end{bmatrix} \tag{4}$$

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

$$(\mathbf{1} - \mathbf{R}) \cdot \mathbf{B} = \mathbf{E} \qquad \mathbf{1} \text{ stands for the unit matrix} \tag{5}$$

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 calculated B_i radiosities represent the light density of the surface on a given wavelength. 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.

3. Solve the linear equation for representative wavelengths, yielding $B_i^{\lambda_1}$, $B_i^{\lambda_2}$ ... $B_i^{\lambda_n}$.

4. 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.

In practical circumstances the number of elemental surface patches can be very large, making the form factor computation and the solution of the linear equation rather time consuming. Thus special tricks are badly needed to speed up the method and to make it appropriate even for very complex object scenes. The tricks are usually based on iteration or on randomization. This paper evaluates and compares the various radiosity calculation algorithms with respect to their convergence and complexity, and highlights the merits of the randomized approaches as well.

Form factor computation

The direct application of the described algorithm requires the calculation of the N^2 number of form factors first, then the solution of an $N \times N$ linear equation. The time and space requirements of the form factor calculation, if all of them are needed, cannot be less then $O(N^2)$ time and space because the output size of the solution is exactly N^2.

A class of form factor calculation algorithms, called geometric methods, aims at computing all the form factors. Thus, in the optimal case we can expect a geometric algorithm to have $O(N^2)$ complexity. Randomized methods, however, take advantage of the fact that many of the form factors are very small, which are not worth computing and storing. This means that only the relevant form factors are estimated, which results in an algorithm having more appealing complexity.

Geometric methods

Geometrical form factor calculation methods apply the following approximation:

$$F_{ij} = \frac{1}{\Delta A_i} \cdot \int\limits_{\Delta A_i} \int\limits_{\Delta A_j} H_{ij} \cdot \frac{dA_i \cdot \cos\phi_i \cdot dA_j \cdot \cos\phi_j}{\pi \cdot r^2} \approx \int\limits_{\Delta A_j} H_{ij} \cdot \frac{\cos\phi_i \cdot \cos\phi_j}{\pi \cdot r^2} \, dA_j \qquad (6)$$

where $\cos\phi_i$, $\cos\phi_j$ and r are the angles of the surface normals and the direction vector pointing from dA_i and dA_j and the length of this vector respectively.

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, 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**.

This means that a single row of the form factor matrix can be determined by solving a visibility problem, where the "window" is a half sphere and the "eye" is in the center of surface i, followed by an area computation. It is well known that only image space visibility algorithms, such as z-buffer algorithm and ray tracing for instance, are capable of doing this in linear, that is in $O(N \cdot P^2)$, time, where P^2 is the number of "pixels" used to discretize the continuous image space. Area computation over a discretized space can be done in $O(P^2)$ time. The complicated form of the "window" can be simplified if the hemisphere is replaced by other immediate surfaces, such as a **hemicube** [CG85] or a **cubic tetrahedron** [BKP91]. In these cases the modification of the geometry must be compensated by appropriate weighting functions in area computation. This modification does not effect the inherent visibility computation phase, thus their computation time can differ only in a scalar factor. This scalar factor, however, can be really significant. For hemi-cube and hemishpere algorithms, the window surface consists of normal rectangles, which can be exploited by the built in transformation and scan-conversion hardware of graphics workstations.

Using z-buffer method and supposing a hemicube to be placed over the surfaces, the form factor calculation algorithm is shown below, proving that it really has $O(N^2 \cdot P^2)$ time complexity.

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) // *requires* $O(N)$ *time*
    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

Randomized form factor calculation

Probabilistic or randomized approaches can often reduce the complexity for the price of generating the correct result only with a given probability that can be made very close to 1. A randomized approach to form factor calculation 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 if the underlying probability distributions are defined as following:

1. Assume the origin of the particle on the surface to be selected randomly by uniform distribution.

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.

This probability 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, $\hat{F}_{ij} = k_j/n$, since k_j is a random variable of binomial distribution with expectation value nF_{ij} and square variance $nF_{ij}(1-F_{ij})$. Thus the expectation value and square variance of the relative frequency are:

$$E[\hat{F}_{ij}] = F_{ij}, \qquad \sigma^2[\hat{F}_{ij}] = \frac{F_{ij}(1-F_{ij})}{n} \leq \frac{1}{4n} \tag{7}$$

This means that if the square variance of the form factor estimation is expected to be less than some D, then n must be greater than $1/4D$.

Summarizing the pseudo code of the randomized form factor calculation is as follows:

for $i = 1$ **to** N **do for** $j = 1$ **to** N **do** $F_{ij} = 0$;
for $i = 1$ **to** N **do for** $j = 1$ **to** n **do**
  $\vec{p}$ = a random point on surface i by uniform distribution
  $\vec{d}$ = a random direction from $\vec{p}$ by cosine distribution
  if $ray(\vec{p}, \vec{d})$ hits surface s first **then** F_{is} += $1/n$;
endfor

The only step which requires $O(N^2)$ time is the initialization of the form factor matrix by zeros, the rest part is only an $O(N \cdot n)$ algorithm. Since n is independent of N, the randomized form factor method can calculate the form factors in $O(N)$ time. This means that if $n < N$, then the randomized algorithm can estimate only a subset of the form factors, those having very small values are ignored by the random simulation.

The $O(N^2)$ initialization phase can be improved by sparse matrix techniques, which store only non-zero matrix elements with their indices in appropriate data structures. The handling of these data structures, however, can pose additional computational problem which must be taken into consideration. If the non-zero elements and their indices are stored in a balanced binary tree, for instance, then the update of a single matrix element is an $O(\log N)$ process, making the form factor computation have $O(N \log N)$ complexity.

Solution of the linear equation

The most obvious way to solve a linear equation is to apply the Gauss elimination method. Unfortunately it fails to solve the radiosity equation for more complex models effectively, since it has $O(N^3)$ time 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 side. 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:

$$\mathbf{B}(m+1) = \mathbf{R} \cdot \mathbf{B}(m) + \mathbf{E} \tag{8}$$

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

$$\mathbf{B}(m+1) - \mathbf{B}(m) = \mathbf{R} \cdot (\mathbf{B}(m) - \mathbf{B}(m-1)) = \mathbf{R^m} \cdot (\mathbf{B}(1) - \mathbf{B}(0)) \tag{9}$$

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's use the $\|\mathbf{R}\|_\infty$ norm defined as the maximum of absolute row sums, and a vector norm that is compatible with it:.

$$\|\mathbf{R}\|_\infty = \max_i \{\sum_j F_{ij} \cdot \varrho_i\}, \qquad \|\mathbf{b}\|_\infty = \max_i \{|b_i|\} \tag{10}$$

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

$$\|\mathbf{B}(m+1) - \mathbf{B}(m)\| = \|\mathbf{R^m} \cdot (\mathbf{B}(1) - \mathbf{B}(0))\| \leq \|\mathbf{R}\|^m \cdot \|\mathbf{B}(1) - \mathbf{B}(0)\| = q^m \cdot \|\mathbf{B}(1) - \mathbf{B}(0)\| \tag{11}$$

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.

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.

Denoting the lower triangle matrix and the upper triangle matrix of $\mathbf{R}$ by $\mathbf{L}$ and $\mathbf{U}$ respectively, the Gauss-Seidel iteration can be defined in matrix form:

$$\mathbf{B}(m+1) = \mathbf{L} \cdot \mathbf{B}(m+1) + \mathbf{U} \cdot \mathbf{B}(m) + \mathbf{E} \tag{12}$$

As for the normal iteration, the equation representing step $m-1$ of the iteration is subtracted from this, resulting in:

$$\mathbf{B}(m+1) - \mathbf{B}(m) = \mathbf{L} \cdot (\mathbf{B}(m) - \mathbf{B}(m-1)) + \mathbf{U} \cdot (\mathbf{B}(m+1) - \mathbf{B}(m)) \tag{13}$$

Denoting $\mathbf{B}(m+1) - \mathbf{B}(m)$ by $\delta\mathbf{B}(m+1)$ we get:

$$\delta\mathbf{B}(m+1) = \mathbf{L} \cdot \delta\mathbf{B}(m+1) + \mathbf{U} \cdot \delta\mathbf{B}(m) \tag{14}$$

Let α_i and β_i be the sum of absolute values of the elements in the row i of $\mathbf{L}$ and $\mathbf{U}$ respectively, that is:

$$\alpha_i = \sum_{j=1}^{n} |\mathbf{L}_{ij}| = \sum_{j=1}^{i-1} |\mathbf{R}_{ij}|, \qquad \beta_i = \sum_{j=1}^{n} |\mathbf{U}_{ij}| = \sum_{j=i+1}^{n} |\mathbf{R}_{ij}| \tag{15}$$

Using the definition of $\|\mathbf{R}\|_\infty$ we have: $\alpha_i + \beta_i \leq \|\mathbf{R}\|_\infty = q$. Let us estimate the absolute value of the element i of $\delta\mathbf{B}(m+1)$ using equation 14:

$$|\delta\mathbf{B}(m+1)|_i \leq \alpha_i \cdot ||\delta\mathbf{B}(m+1)|| + \beta_i \cdot ||\delta\mathbf{B}(m)|| \tag{16}$$

Expressing the norm of $\delta\mathbf{B}(m+1)$ from this, we are left with:

$$||\delta\mathbf{B}(m+1)|| \leq \max_i\{\alpha_i \cdot ||\delta\mathbf{B}(m+1)|| + \beta_i \cdot ||\delta\mathbf{B}(m)||\} \tag{17}$$

Let $\alpha = \alpha_i$ and $\beta = \beta_i$ for that i which maximizes the right hand side of this inequality. Rewriting this inequality with these two new variables:

$$||\delta\mathbf{B}(m+1)|| \leq \alpha \cdot ||\delta\mathbf{B}(m+1)|| + \beta \cdot ||\delta\mathbf{B}(m)|| \tag{18}$$

Expressing $||\delta\mathbf{B}(m+1)||$ we have:

$$||\delta\mathbf{B}(m+1)|| \leq \frac{\beta}{(1-\alpha)} \cdot ||\delta\mathbf{B}(m)|| \tag{19}$$

The iteration is convergent if $\beta/(1-\alpha) < 1$. Since $\alpha + \beta < q$ where q is the norm of $\mathbf{R}$ and $q < 1$, we can write:

$$\frac{\beta}{1-\alpha} \leq \frac{q-\alpha}{1-\alpha} = q - \frac{\alpha(1-q)}{1-\alpha} < q \tag{20}$$

This means that Gauss-Seidel iteration is always convergent for radiosity equations and its speed of convergence is better than that of the normal iteration.

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 $\delta\mathbf{B} = \mathbf{B}(m+1) - \mathbf{B}(m)$ showing the magnitude of difference, as well as the direction of the improvement in the 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:

$$\mathbf{B}^{*}(m+1) = \mathbf{B}(m) + \omega \cdot \delta\mathbf{B} \tag{21}$$

The determination of ω is a crucial problem. 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.

Progressive refinement

The previously discussed radiosity method determined the form factor matrix first, then solved the linear equation by iteration. 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:

$$B_i(m+1) = E_i + \varrho_i \sum_j B_j(m) \cdot F_{ij} = E_i + \sum_j (B_i \text{ due to } B_j(\mathrm{m})) \tag{22}$$

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. 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:

$$B_i^{new} = B_i^{old} + \varrho_i \cdot (\Delta B_j) \cdot F_{ij} \tag{23}$$

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 23, 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. This reduces the burden of the storage from the $N \times N$ matrix elements to only a single column containing N elements, but necessitates the recalculation of the form factors. Let us realize that equation 23 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:

$$F_{ji} \cdot \Delta A_j = F_{ij} \cdot \Delta A_i \implies F_{ij} = F_{ji} \cdot \frac{\Delta A_j}{\Delta A_i} \quad (i = 1, ..., N) \tag{24}$$

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;
  $U_j = 0$;
  $error = \max\{U_1, U_2, ..., U_N\}$;
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 the maximal in step m, and using the notation $q = \|\mathbf{R}\|_\infty$ again:

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

$$\leq (\sum_i^N U_i(m)) - (1-q) \cdot U_j \leq (1 - \frac{1-q}{N}) \cdot \sum_i^N U_i(m) = q^* \cdot \sum_i^N U_i(m) \tag{25}$$

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. Its effect can be evaluated by determining the number of iterations needed to make

$$error = \max\{U_1, U_2, ..., U_N\} < \varepsilon$$

which stops the algorithm. Since after the mth step of iteration

$$\max\{U_1(m), U_2(m), ..., U_N(m)\} \leq \sum_i^N U_i(m) \leq (1 - \frac{1-q}{N})^m \cdot \sum_i^N U_i(0), \tag{26}$$

the stopping condition is satisfied if

$$(1 - \frac{1-q}{N})^m < \frac{\varepsilon}{\sum_i^N U_i(0)} = C \tag{27}$$

Expressing m we get:

$$m > \frac{\log C}{\log(1-(1-q)/N)} \approx \frac{-\log C}{((1-q)/N)} = \frac{-\log C \cdot N}{1-q} \tag{28}$$

Thus the slow down of the convergence is approximately a factor of N, which makes 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. Interestingly, progressive refinement does not decrease the $O(N^2)$ time complexity, but it can achieve $O(N)$ space complexity instead of the $O(N^2)$ behavior obtained by the original method.

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 += $\varrho_i \cdot U_j/n$;
  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.

Let us realize that this algorithm is always convergent since the total unshot radiosity decreases in each step by a factor less than 1 as for the case of deterministic progressive refinement. Using the considerations made when the speed of the convergence in the progressive refinement was calculated, we can conclude that the number of the required iteration steps is proportional to $N \cdot n$. This means that this algorithm also requires $O(n \cdot N^2)$ time to determine the radiosities. This algorithm has very appealing storage complexity. Since it does not require the form factors to be calculated and stored (the random shooting distributes the energy proportionally to the form factors), the storage complexity is $O(N)$.

Since this is a randomized algorithm, the radiosities to which this is converging approximate the real radiosities with only a given probability. The probability can be controlled by the factor n. The definition of this factor is a crucial problem since if it is small, then the result will probably be inaccurate, but if it is large, then the algorithm will be very slow. It can be shown that the expectation of the calculated radiosities will be equal to the real radiosities and the square variance will be lower then $E^2/(4n - N)$ where E is the total energy of the system. Thus by increasing n the variance can be kept under control.

Conclusions

This paper reviewed the radiosity calculation algorithms which can be classified to classical methods which separate the steps of form factor calculation and the solution of the linear equation and direct methods which do not require a separate form factor calculation step. Due to the output size of the problem, the form factor calculation cannot be better than $O(N^2)$, both in space and time, if all the form factors are needed but optimal algorithm based on geometric considerations doing this job in $O(N^2 \cdot P^2)$ time are available. However, if the negligible form factors need not be computed even better algorithms can be constructed using randomized techniques. If we can assume the form factor matrix to be initialized by zeros, an $O(n \cdot N)$ method can be elaborated. If no such initialized matrix is available, sparse matrix techniques result in $O(n \cdot N \log N)$ time and $O(N)$ space algorithms. These methods come to the foreground if N is very large.

Classical methods require the solution of a linear equation which requires $O(N^3)$ time if Gauss elimination is used, but only $O(N^2)$ time if iteration is applied. It has been shown that both the normal and the Gauss-Seidel iteration are always convergent for radiosity equations, and that the Gauss-Seidel iteration's speed of convergence is always better.

Direct methods do not separate the form factor calculation from the radiosity determination. The progressive refinement has thus $O(N)$ storage complexity but, as has been proven, has still $O(N^2)$ time complexity. Its randomized version does not use the concept of form factors at all, which resulted in an extremely simple algorithm, but its complexity characteristics could not be made better that that of the progressive refinement.

References

[BKP91] Jeffrey C. Beran-Koehn and Mark J. Pavicic. A cubic tetrahedra adaption of the hemicube algorithm. In James Arvo, editor, *Graphics Gems II*, pages 299–302. Academic Press, Boston, 1991.

[CG85] Michael Cohen and Donald Greenberg. The hemi-cube, a radiosity solution for complex environments. In *Proceedings of SIGGRAPH '85*, pages 31–40, 1985.

[CGIB86] Michael F. Cohen, Donald P. Greenberg, David S. Immel, and Phillip J. Brock. An efficient radiosity approach for realistic image synthesis. *IEEE Computer Graphics and Applications*, 6(3):26–35, 1986.

[SH81] Robert Siegel and John R. Howell. *Thermal Radiation Heat Transfer*. Hemisphere Publishing Corp., Washington, D.C., 1981.