A Ray Tracing Solution for Diffuse Interreflection

Gregory J. Ward
Francis M. Rubinstein
Robert D. Clear

Lighting Systems Research
Lawrence Berkeley Laboratory
1 Cyclotron Rd., 90-3111
Berkeley, CA 94720
(415) 486-4757

Abstract

An efficient ray tracing method is presented for calculating interreflections between surfaces with both diffuse and specular components. A Monte Carlo technique computes the indirect contributions to illuminance at locations chosen by the rendering process. The indirect illuminance values are averaged over surfaces and used in place of a constant "ambient" term. Illuminance calculations are made only for those areas participating in the selected view, and the results are stored so that subsequent views can reuse common values. The density of the calculation is adjusted to maintain a constant accuracy, permitting less populated portions of the scene to be computed quickly. Successive reflections use proportionally fewer samples, which speeds the process and provides a natural limit to recursion. The technique can also model diffuse transmission and illumination from large area sources, such as the sky.

General Terms: Algorithm, complexity.

Additional Keywords and Phrases: Caching, diffuse, illuminance, interreflection, luminance, Monte Carlo technique, radiosity, ray tracing, rendering, specular.

1. Introduction

The realistic computer rendering of a geometric model requires the faithful simulation of light exchange between surfaces. Ray tracing is a simple and elegant approach that has produced some of the most realistic images to date. The standard ray tracing method follows light backwards from the viewpoint to model reflection and refraction from specular surfaces, as well as direct diffuse illumination and shadows [15]. Accuracy has been improved with better reflection models [4] and stochastic sampling techniques [6]. Unfortunately, the treatment of diffuse interreflection in conventional ray tracers has been limited to a constant "ambient" term. This approximation fails to produce detail in shadows, and precludes the use of ray tracing where indirect lighting is important.

We present a method for modeling indirect contributions to illumination using ray tracing. A diffuse interreflection calculation replaces the ambient term directly, without affecting the formulas or algorithms used for direct and specular components. Efficiency is obtained with an appropriate mix of view-dependent and view-independent techniques.

 ACM-0-89791-275-6/88/008/0085 $00.75

2. Interreflection in Ray Tracing

Ray tracing computes multiple reflections by recursion (Figure 1). At each level, the calculation proceeds as follows:

1. Intersect the ray with scene geometry.
2. Compute direct contributions from light sources.
3. Compute specular contributions from reflecting surfaces.
4. Compute diffuse contributions from reflecting surfaces.

The complexity of the calculation is closely related to the difficulty of step 1, and the number of times it is executed as determined by the propagation (recursion) of steps 2 through 4. Step 2 requires as many new rays as there are light sources, but the rays do not propagate so there is no growth in the calculation. Step 3 can result in a few propagating rays that lead to geometric growth if unchecked. Methods for efficient specular component computation have been described by [8], [5] and [14]. The diffuse contributions in step 4, however, require many (>100) propagating rays that quickly overwhelm a conventional calculation. Most methods simply avoid this step by substituting a constant ambient term. Our goal is to find an efficient method for computing diffuse interreflection and thereby complete the ray tracing solution. We start with a summary of previous work in this area.

An advanced ray tracing method developed by Kajiya follows a fixed number of paths to approximate global illumination at each pixel [8]. Using hierarchical "importance" sampling to reduce variance, the illumination integral is computed with fewer rays than a naive calculation would require. This brings ray tracing closer to a full solution without compromising its basic properties: separate geometric and lighting models, view-dependence for efficient rendering of specular effects, and pixel-independence for parallel implementations. Unfortunately, the method is not well suited to calculating diffuse interreflection, which still requires hundreds of samples. A high-resolution image simply has too many pixels to compute global illumination separately at each one.

The radiosity method, based on radiative heat transfer, is well suited to calculating diffuse interreflection [12][10][2]. Surfaces are discretized into patches of roughly uniform size, and the energy exchange between patches is computed in a completely view-independent manner. The method makes efficient use of visibility information to compute multiple reflections, and sample points are spaced so that there is sufficient resolution without making the calculation intractable. In areas where illumination changes rapidly, the patches can be adaptively subdivided to maintain accuracy [3]. However, the standard radiosity method models only diffuse surfaces, which limits the realism of its renderings. Immel extended the approach to include non-diffuse environments, adding bidirectional reflectance to the energy equations [7]. Unfortunately, the view-independent solution of specular interreflection between surfaces requires sampling radiated directions over very small (approaching pixel-sized) surface patches. The resulting computation is intractable for all but the simplest scenes.

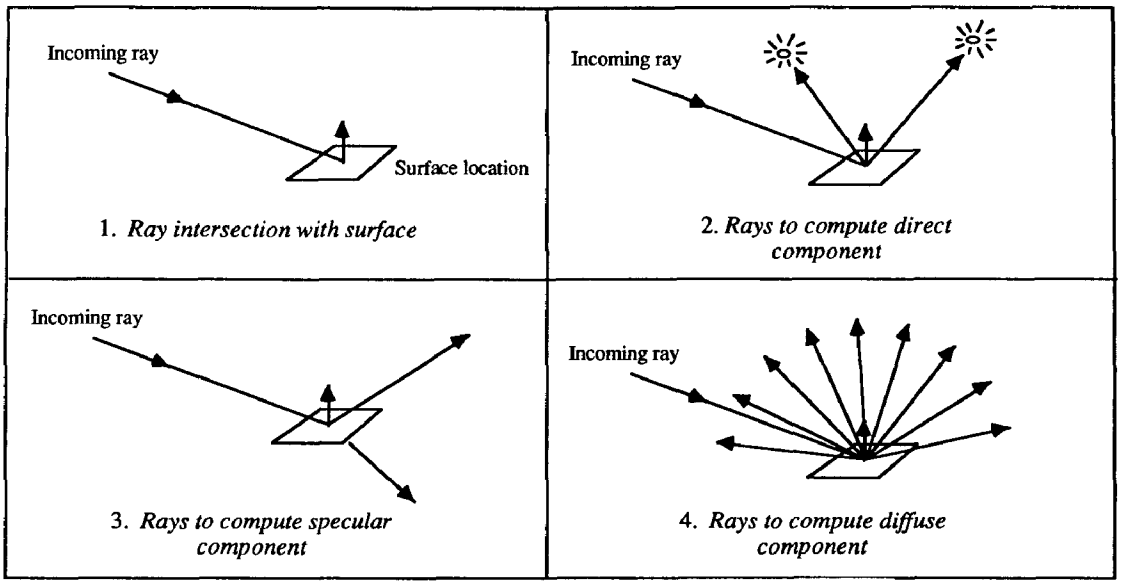

Figure 1: The four steps of ray tracing.

A combined ray tracing and radiosity approach was designed by Wallace to take advantage of the complementary properties of the two techniques [13]. Wallace divides energy transport into four "mechanisms:" diffuse-diffuse, specular-diffuse, diffuse-specular, and specular-specular. He then proceeds to account for most of these interactions with clever combinations of ray tracing and radiosity techniques. Unfortunately, there are really an infinite number of transport mechanisms, such as specular-specular-diffuse, which are neglected by his calculation. The generalization Wallace suggests for his approach is equivalent to *view-independent* ray tracing, which is even more expensive than general radiosity [7].

3. Diffuse Indirect Illumination

Our development of an efficient ray tracing solution to diffuse interreflection is based on the following observations:

- Because reflecting surfaces are widely distributed, the computation of diffuse indirect illumination requires many sample rays.
- The resulting "indirect illuminance" value† is view-independent by the Lambertian assumption [9].
- The indirect illuminance tends to change slowly over a surface because the direct component and its associated shadows have already been accounted for by step 2 of the ray tracing calculation.

For the sake of efficiency, indirect illuminance should not be recalculated at each pixel, but should instead be averaged over surfaces from a small set of computed values. Computing each value might require many samples, but the number of values would not depend on the number of pixels, so high resolution images could be produced efficiently. Also, since illuminance does not depend on view, the values could be reused for many images.

How can we benefit from a view-independent calculation in the inherently view-dependent world of ray tracing? We do not wish to limit or burden the geometric model with illuminance information, as required by the surface discretization of the radiosity method. By the same token, we do not wish to take view-independence too far, calculating illuminance on surfaces that play no part in the desired view. Instead we would like to take our large sample of rays only when and where it is necessary for the accurate computation of an image, storing the result in a separate data structure that puts no constraints on the surface geometry.

†We define indirect illuminance as the light flux per unit area arriving at a surface location via non-self-luminous surfaces.

In our enhancement of the basic ray tracing technique, indirect illuminance values are cached in the following manner:

```
If one or more values is stored near this point
    Use stored value(s)
Else
    Compute and store new value at this point
```

The computation of a new value uses the "primary method." The technique for finding and using stored values is called the "secondary method." The primary method is invoked to calculate a new value the first time it is needed, which is when the secondary method fails to produce a usable estimate from previous calculations (Figure 2). Determining the appropriate range and presenting a surface-independent storage technique are the two main points of this paper. Before we explore these issues, we present a basic computation of indirect illuminance.

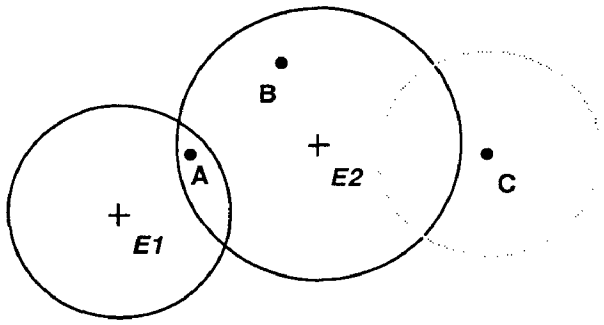

Figure 2: Illuminances *E1* and *E2* were calculated previously using the primary method. Test point **A** uses an average of *E1* and *E2*. Point **B** uses *E2*. Point **C** results in a new indirect illuminance value at that location.

3.1. The Illuminance Integral

Illuminance is defined on a surface as the integral of luminance over the projected hemisphere [9]:

$$E = \int_0^{2\pi}\int_0^{\frac{\pi}{2}} L(\theta, \phi) \cos\theta \sin\theta \, d\theta \, d\phi \qquad (1)$$

where θ = polar angle

ϕ = azimuthal angle

$L(\theta, \phi)$ = luminance from direction (θ, ϕ)

In our primary method for calculating indirect illuminance, the integral is approximated with a discrete set of sample rays that do not intersect light sources. A uniform segmented Monte Carlo distribution is derived by standard transformation methods [11]:

$$E \approx \frac{\pi}{2n^2} \sum_{j=1}^{n} \sum_{k=1}^{2n} L(\theta_j, \phi_k) \qquad (2)$$

where: $\theta_j = \sin^{-1}\left(\sqrt{\frac{j - X_j}{n}}\right)$

$\phi_k = \pi \frac{(k - Y_k)}{n}$

X_j, Y_k = uniform random numbers between 0 and 1

$2n^2$ = total number of samples

In general, a better approximation to (1) may be obtained with fewer rays using hierarchical sampling techniques [8]. The particular method chosen does not affect the remainder of this discussion.

3.2. Illuminance Averaging

The secondary method performs two functions alternatively. It either approximates illuminance by averaging between primary values, or determines that a new primary value is needed. To maintain a constant accuracy with a minimum of primary evaluations, it is necessary to estimate the illuminance gradient on each surface. Where the illuminance changes slowly, as in flat open areas, fewer values are required. Where there is a large gradient, from high surface curvature or nearby objects, more frequent primary evaluations are necessary. Our method uses an estimate of the change in illuminance over a surface based on scene geometry. The inverse of this change serves as the weight for each primary value during averaging. If none of the values has a weight above a specified minimum, the primary method is invoked at that location.

We introduce a simple model to relate the illuminance gradient to scene geometry based on the assumption that narrow concentrations of luminance can be neglected. (Such localized sources should be included in the direct component calculation, since Monte Carlo sampling is a bad way to find them.) A surface element is located at the center of a sphere (Figure 3). Half of the sphere is bright, the other half is dark. The surface element faces the dividing line between the two halves. The "split sphere" has the largest gradient possible for an environment without concentrated sources.

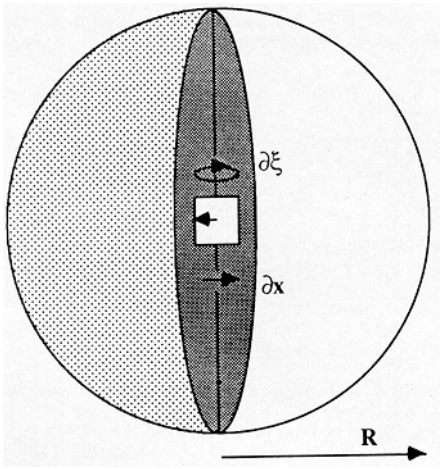

Figure 3: The split sphere model. A surface element is located at the center of a half-dark sphere.

An approximate bound to the change in illuminance in the split sphere, ϵ, is given by the first order Taylor expansion for a function of two variables:

$$\epsilon \le \left| \frac{\partial E}{\partial x}(x - x_o) + \frac{\partial E}{\partial \xi}(\xi - \xi_o) \right| \qquad (3a)$$

Because the illuminance at the center is proportional to the projected area of the bright half of the hemisphere, the partial differentials with respect to x and ξ are proportional to the partial changes in this projection. In terms of x, the differential change over the projected area is $\frac{2R\,\partial x}{\frac{1}{2}\pi R^2}$, which is $\frac{4\partial x}{\pi R}$. In terms of ξ, the ratio is $\frac{\frac{1}{2}\pi R^2 \partial\xi}{\frac{1}{2}\pi R^2}$, or simply $\partial\xi$. Combining these results with the triangle inequality, we get:

$$\epsilon \le \frac{4}{\pi}\frac{E_o}{R}\left|x - x_o\right| + E_o\left|\xi - \xi_o\right| \qquad (3b)$$

Note that the change in illuminance with respect to location is inversely proportional to the radius, R, while the change with respect to orientation does not depend on the sphere geometry. We can extend our approximation to more complicated geometries by replacing x and ξ with vector-derived values:

$$\epsilon(\vec{P}) \le E_o\left[\frac{4}{\pi}\frac{\|\vec{P} - \vec{P}_o\|}{R_o} + \sqrt{2 - 2\vec{N}(\vec{P})\cdot\vec{N}(\vec{P}_o)}\right] \qquad (4)$$

where: $\vec{N}(\vec{P})$ = surface normal at position $\vec{P}$

$\vec{P}_o$ = surface element location

E_o = illuminance at $\vec{P}_o$

R_0 = "average" distance to surfaces at $\vec{P}_0$

The change in x becomes the distance between two points, and the change in ξ becomes the angle between two surface normals. This equation is used to estimate the relative change in illuminance for any geometry. Both the points and the surface normals are determined by the ray intersection calculation. R_0 is the harmonic mean (reciprocal mean reciprocal) of distances to visible surfaces, which can be computed from ray lengths during primary evaluation.

The inverse of the estimated error is used in a weighted average approximation of illuminance:

$$E(\vec{P}) = \frac{\sum_{i \in s} w_i(\vec{P})\, E_i}{\sum_{i \in s} w_i(\vec{P})} \qquad (5)$$

where: $$w_i(\vec{P}) = \frac{1}{\frac{\|\vec{P} - \vec{P_i}\|}{R_i} + \sqrt{1 - \vec{N}(\vec{P}) \cdot \vec{N}(\vec{P_i})}}$$

E_i = computed illuminance at $\vec{P_i}$

R_i = harmonic mean distance to objects visible from $\vec{P_i}$

S = $\{ i : w_i(\vec{P}) > 1/a \}$

a = user selected constant

The approximate illuminance, $E(\vec{P})$, is given by the weighted mean of all "adjacent" illuminance values. The weight of a value is equal to the inverse of its estimated error, without the constant terms that are valid only for the split sphere ($4/\pi$ and $\sqrt{2}$). An illuminance value with an error of zero ($\vec{P_i}$ equal to $\vec{P}$) will have infinite weight. All values with an estimated error less than a will be included in the set of adjacent illuminances, S. If S is empty, a new primary illuminance value must be calculated at $\vec{P}$. (An efficient method for determining the members of S is given in the next section.)

The constant a is directly related to the maximum approximation error. When the approximation is applied to the split sphere, the error is less than $1.4a\bar{E}$, where $\bar{E}$ is a straight average of E_i over S. In general, the illuminance gradient may be larger or smaller than the split sphere, but it will always be roughly proportional to a. It is interesting to note that for a less than or equal to 1, S will not contain any value farther than the average spacing or with a surface normal more than 90 degrees from the test location. Intuitively, such a value would be expected to have 100% error.

In practice, additional tests are required to restrict the values included in S. The ray recursion depth must be considered so that values computed after one or more bounces are not substituted for final illuminances. This is easily prevented by keeping separate value lists at each recursion level. A different problem arises from our generalization of the split sphere model. Equation (4) assumes that motion in any direction is equivalent to motion in x. As a result, the set S can include illuminance values that lie on objects shadowing the test point, $\vec{P}$ (Figure 4). We therefore introduce a test to reject illuminance values that are "in front" of $\vec{P}$:

$$d_i(\vec{P}) = \left(\vec{P} - \vec{P_i}\right) \cdot \left[\vec{N}(\vec{P}) + \vec{N}(\vec{P_i})\right] / 2 \qquad (6)$$

If $d_i(\vec{P})$ is less than zero, then $\vec{P_i}$ is in front of $\vec{P}$ so the value is excluded.

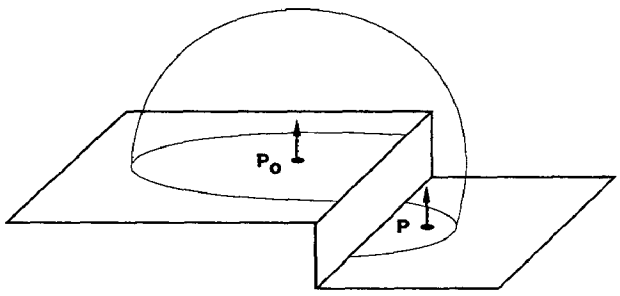

Figure 4: $\vec{P}_0$ sees few close-by surfaces, so its estimated error at $\vec{P}$ is small. But $\vec{P}$ is shadowed by the surface under $\vec{P}_0$, and the true illuminance is different.

Caching indirect illuminance is simple and efficient. The error estimate results in a minimum of primary evaluations and a nearly constant accuracy. Sections of the scene that do not contribute to the image, directly or indirectly, are not examined since no rays reach them. Areas where the indirect illuminance varies rapidly, from changing surface orientation or the influence of nearby objects, will have a higher concentration of values. Flat areas without nearby influences will have only a few values. Dynamic evaluation obviates surface discretization and presampling, so scene representation is not restricted.

Figure 5a shows three colored, textured blocks on a table illuminated by a low-angle light source. Figure 5b shows the placement of indirect illuminance values. Note that the values crowd around inside corners, where surfaces are in close visual contact, and outside corners, where the surface curvature is large. Also, the space between and immediately surrounding the blocks is more densely populated than the background, where only a few values are spread over a wide area. This distribution is different from the standard radiosity technique, which computes values at grid points on each surface. By selecting value locations based on the estimated illuminance gradient, a more accurate calculation is obtained with fewer samples.

Averaging illuminance values over surfaces results in lower pixel variance than produced by standard ray tracing techniques. Figure 5c was produced by a pure Monte Carlo computation that used as many rays as the calculation of Figure 5a. The speckling results from inadequate integration of the indirect contributions at each pixel. Since every pixel requires a separate calculation, only a few diffuse samples are possible over the hemisphere. If a sample happens to catch a bright reflection, the illuminance computed at that point will be disproportionately large. Caching permits a better integration to be performed less frequently, thereby obtaining a more realistic rendering than is feasible with pixel-independent ray tracing.

3.3. Illuminance Storage

For the secondary method to be significantly faster than the primary method, we need an efficient technique for finding the members of S (Equation 5). Without placing any restrictions on scene geometry, an octree permits efficient range searching in three dimensions [1]. A global cube is identified that encompasses all finite surfaces in the scene. When the primary method calculates a new indirect illuminance at a scene location, the global cube is subdivided as necessary to contain the value. Each illuminance, E_i, is stored in the octree node containing its position, $\vec{P_i}$, and having a size (side length) greater than twice but not more than four times the appropriate "valid domain," aR_i. This guarantees that the stored illuminance value will satisfy the condition for S in no more than eight cubes on its own octree level, and a value with a small valid domain will only be examined in close-range searches. Each node in the octree will contain a (possibly empty) list of illuminance values, and a (possibly nil) pointer to eight children. (A two-dimensional analogy is given in Figure 6.) To search the tree for values whose valid domain may contain the point $\vec{P}$, the following recursive procedure is used:

```
For each illuminance value at this node
    If w_i(P) > 1/a and d_i(P) ≥ 0
        Include value
For each child
    If P is within half the child's size of its cube boundary
        Search child node
```

This algorithm will not only pick up the nodes containing $\vec{P}$, but will also search nodes having boundaries within half the cube size of $\vec{P}$. In this way, all lists that might have an illuminance value whose valid domain contains $\vec{P}$ will be examined. The worst case performance of this algorithm is $O(N)$, where N is the number of values. Performance for a uniform distribution is $O(log(N))$.

The scale of the sorting algorithm can be changed so that the octree cubes are either larger or smaller than the domains of the values they contain. If the cubes are smaller, each examined list will be more likely to contain usable values. However, many of the cubes will be empty. If the cubes are larger, more of the values will have to be examined, but less searching through the tree will be necessary. In any case, changing the scale does not affect the functioning of the algorithm, only its performance in a given situation.

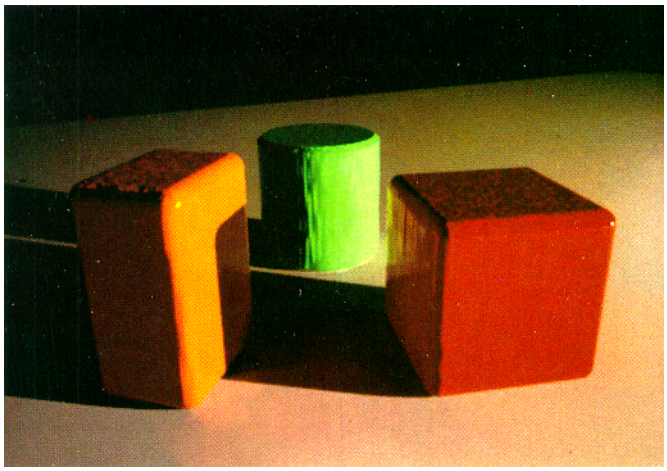

Figure 5a: Colored blocks with diffuse indirect calculation.

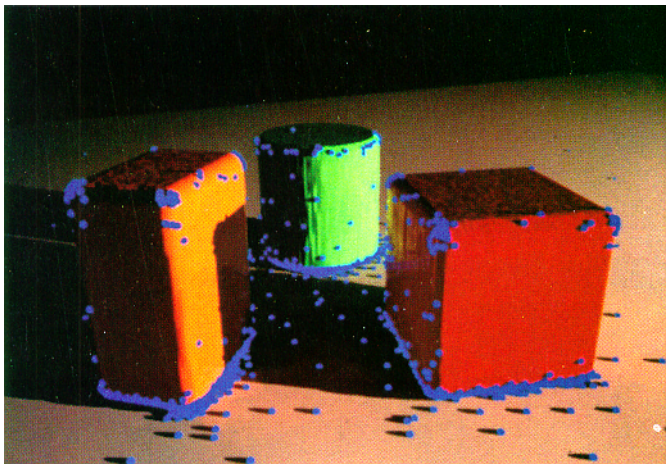

Figure 5b: Blocks with illuminance value locations in blue.

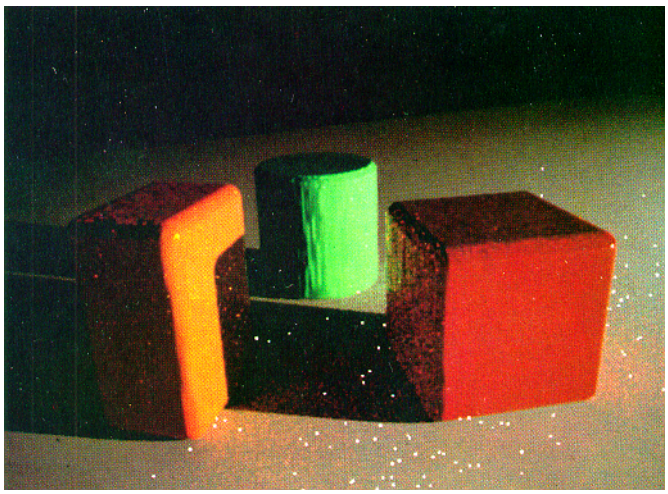

Figure 5c: Blocks using conventional ray tracing techniques.

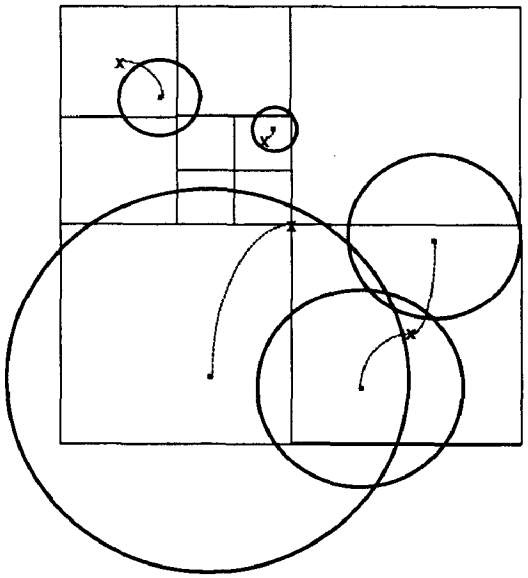

Figure 6: Five indirect illuminance values are shown with their respective domains (circles) linked by dotted lines to the appropriate nodes (squares).

Writing illuminance values to a file permits their reuse in subsequent renderings. By reusing old values, the indirect calculation will not only proceed more quickly, it will be more accurate since the illuminance is already calculated in some areas of the scene. Normally, the secondary method takes the estimated error right to its tolerance level before calculating a new value. Where the necessary values are precalculated, the tolerance is never reached because all points are within one or more valid domains.

3.4. Multiple Diffuse Reflections

It is often desirable to limit the calculation of diffuse reflections separately from the direct and specular components. A record is kept of how many diffuse bounces have occurred, and this is checked against a user-specified limit. When the limit is reached, a constant ambient value is substituted for the calculation. (This value can be zero.)

The first ray traced in a computation with multiple diffuse reflections begins a cascade of illuminance values (Figure 7). The initial primary evaluation uses many ray samples, and these rays in turn produce many more samples, with the last recursion level exhibiting the densest sampling. As the higher recursion levels become filled with primary illuminance values, fewer rays propagate in the calculation. The computation of multiple diffuse reflections therefore begins slowly, and speeds up as fewer recursion levels require primary evaluation. This process is similar to the "solution" stage of a radiosity technique, which calculates energy transfer between all surfaces before rendering is possible. Producing different views of the same scene is then relatively quick. The thrifty computation of multiple views is also present in our method, with an additional savings from ignoring surfaces that do not contribute to the desired images.

In the computation of multiple reflections, a simple optimization reduces the number of samples required for a given accuracy. If the mean surface reflectance is 50 percent, twice as much error can be tolerated in the calculation of each successive bounce. By increasing the value of a by 40 percent and decreasing the Monte Carlo sampling by 50 percent, each reflection uses one quarter as many rays as the last, with the same contribution to error. The total number of sample rays is then a bounded series, which can serve as a soft limit to recursion when accuracy is critical.

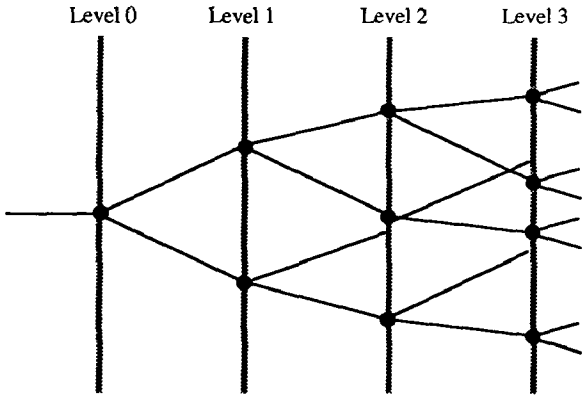

Figure 7: The lines represent rays, and the points represent primary evaluations. The rays that reuse computed values do not propagate.

4. Results

The accuracy of the secondary method was tested with an analytical solution of a sphere resting on an infinite plane with seventy percent reflectance, and a parallel light source overhead (Figure 8). The illumination on the sphere due to direct light plus first bounce was determined. Since the scene is radially symmetric, the sphere illuminance is completely described by a one dimensional function of the angle below the horizontal (γ), as measured from the center of the sphere. A closed form for illuminance was found for the upper half of the sphere, and an analytical function was integrated numerically for the lower half. The illuminance caching calculation was then applied to the problem, and the mean and maximum errors of the secondary method were found for different values of a. In each case, the distribution of error was relatively uniform over the sphere, though the density of primary evaluations varied by several orders of magnitude. The mean error was about one fourth, and the maximum error was about twice the estimated error for the split sphere. The relationship between error and a had the expected linear correlation.

Figures 9a, 9b and 9c show a daylit office space with direct only, first bounce, and seven bounces, respectively. The blind-covered window was modeled as six area sources with precalculated distributions accounting for solar and sky components. The images took 25, 40, and 70 hours in separate calculations on a VAX 11/780.

Figure 10 shows an ice cream store illuminated by indirect cove lighting. The total computation took about 30 hours on a Sun 3/60. We estimate the image would have taken more than 500 hours using pixel-independent ray tracing, or 100,000 hours for an accurate radiosity solution. Although a combined radiosity and ray tracing approach would be comparable to our method in computation time, it would not model many of the interactions shown in this image, such as the illumination under the parfait glass.

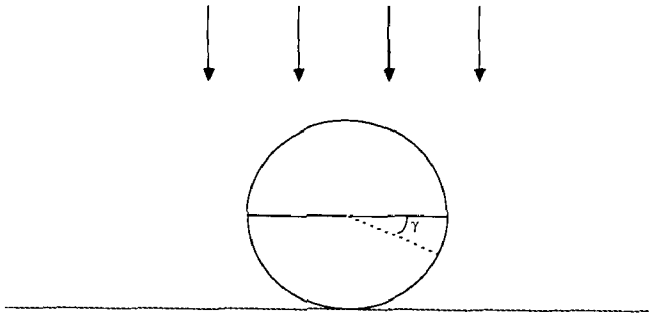

Figure 8: A sphere on an infinite plane, used to validate the secondary method.

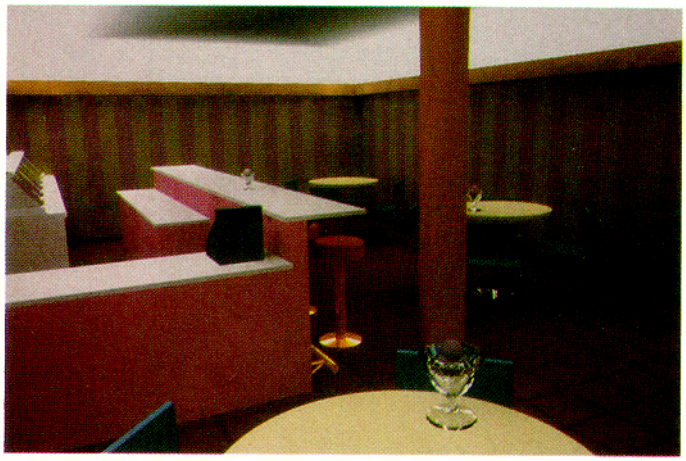

Figure 10: Ice cream store with indirect cove lighting.

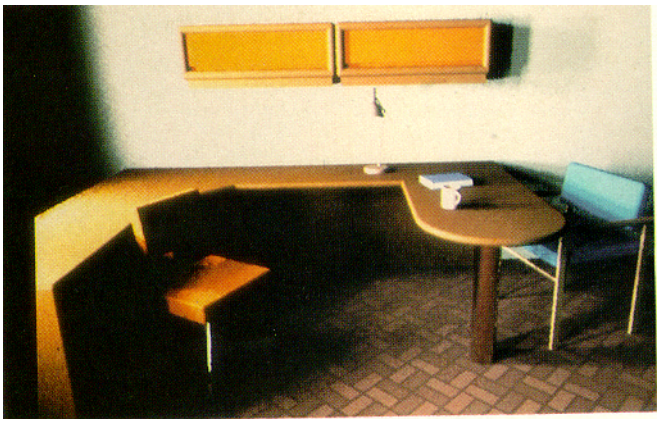

Figure 9a: Daylit office, direct only calculation.

Figure 9b: Daylit office, first bounce calculation.

Figure 9c: Daylit office, seven bounce calculation.

5. Discussion

Because our averaging technique was derived from a simplified model (the split sphere), it is important to study its performance in situations where the model is not predictive. Two such cases are illustrated in Figure 11. They are both related to bright, localized reflections, such as those that might result from a spotlight or mirror. If a bright spot is partially hidden by an occluding surface, or on the horizon, then small changes in element location and orientation can result in large changes in illuminance. The averaging technique we have developed will not respond appropriately, and the error related to a will be much larger than the original split sphere model. However, bright spots also make trouble for the Monte Carlo calculation, which requires a higher sample density to find and integrate such luminance spikes. There is no known lighting calculation that can track these small "secondary sources" efficiently. Our technique uses a smaller value for a together with a higher Monte Carlo sample density to model these effects, with a corresponding increase in complexity.

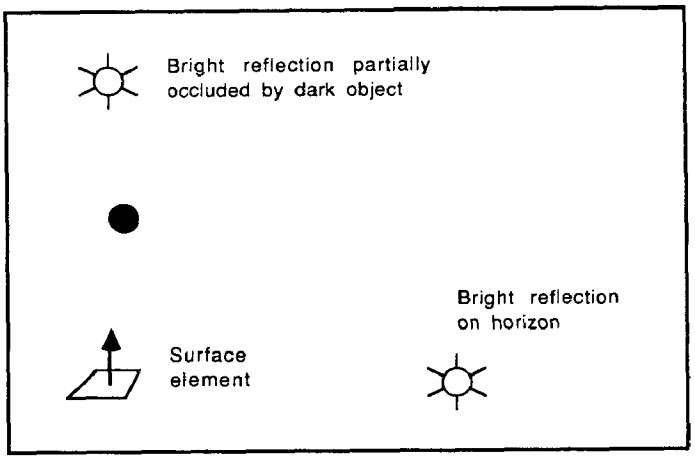

Figure 11: Two cases of indirect illumination that are difficult to model.

Besides diffuse interreflection, the caching technique can also be applied to illumination from large sources, such as a window or the sky. Wide area sources present a problem for conventional ray tracing calculations because they are difficult to sample adequately. Normally, when a ray participating in the diffuse interreflection calculation hits a light source, it is ignored. This prevents counting light sources twice, since they participate in a separate direct component calculation. By moving a large source from the direct to the indirect step of the computation, it is possible to obtain a more accurate sampling of its area. We have found this approach effective for sources with a solid angle greater than 1 steradian.

The calculation of diffuse transmission can also be accelerated by caching. Translucent surfaces become more difficult to model with conventional point sampling techniques as they become more nearly diffuse. The indirect calculation can be used to obtain a more accurate integral of light striking a translucent surface on either side. If the transmission function is not purely diffuse, scattered specular rays can be used to supplement the Monte Carlo calculation, just as they are for reflection.

6. Conclusion

We have developed an efficient ray tracing method for calculating diffuse interreflection, which when combined with standard computations of direct and specular contributions results in a complete simulation of global illumination. Only those illuminance computations required for accurate rendering are performed, and the values can be reused in other images. Thus the method provides an good mix of view-dependent and view-independent qualities. The criterion for evaluation of diffuse interreflection is an estimate of the illuminance gradient from convenient measures of scene geometry. The separation of lighting and geometric models is a basic strength of ray tracing, and it is preserved in this technique. The method can also model diffuse transmission and illumination from large area sources.

7. Acknowledgements

Our thanks go to the reviewers for their comments, and to Sam Berman for his continuing support. We would also like to thank the LBL Computer Science Research Department and the UCB Center for Environmental Design for the use of their equipment. Our special thanks go to Bill Johnston and Paul Heckbert. This work was supported by the Assistant Secretary for Conservation and Renewable Energy, Office of Building Energy Research and Development, Buildings Equipment Division of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

8. References

1. Bentley, Jon Louis and Jerome Friedman, "Data Structures for Range Searching," *ACM Computing Surveys*, Vol. 11, No. 4, 1979, pp. 397-409.

2. Cohen, Michael and Donald Greenberg, "A Radiosity Solution for Complex Environments," *Computer Graphics*, Vol. 19, No. 3, July 1985, pp. 31-40.

3. Cohen, Michael, Donald Greenberg, David Immel, Phillip Brock, "An Efficient Radiosity Approach for Realistic Image Synthesis," *IEEE Computer Graphics and Applications*, Vol. 6, No. 2, March 1986, pp. 26-35.

4. Cook, Robert L. and Kenneth E. Torrance, "A Reflection Model for Computer Graphics," *ACM Transactions on Graphics*, Vol. 1, No. 1, January 1982, pp. 7-24.

5. Cook, Robert, Thomas Porter, Loren Carpenter, "Distributed Ray Tracing," *Computer Graphics*, Vol. 18, No. 3, July 1984, pp. 137-147.

6. Cook, Robert L., "Stochastic Sampling in Computer Graphics," *ACM Transactions on Graphics*, Vol. 5, No. 1, January 1986, pp. 51-72.

7. Immel, David S., Donald P. Greenburg, Michael F. Cohen, "A Radiosity Method for Non-Diffuse Environments," *Computer Graphics*, Vol. 20, No. 4, August 1986, pp. 133-142.

8. Kajiya, James T., "The Rendering Equation," *Computer Graphics*, Vol. 20, No. 4, August 1986, pp. 143-150.

9. Kaufman, John, *IES Lighting Handbook*, Reference Volume, IESNA, New York, NY, 1981.

10. Nishita, Tomoyuki and Eihachiro Nakamae, "Continuous Tone Representation of Three-Dimensional Objects Taking Account of Shadows and Interreflection," *Computer Graphics*, Vol. 19, No. 3, July 1985, pp. 23-30.

11. Rubenstein, R.Y., *Simulation and the Monte Carlo Method*, J. Wiley, New York, 1981.

12. Siegel, R. and J. R. Howell, *Thermal Radiation Heat Transfer*, Hemisphere Publishing Corp., Washington DC., 1981.

13. Wallace, John R., Michael F. Cohen, Donald P. Greenburg, "A Two-Pass Solution to the Rendering Equation: A Synthesis of Ray Tracing and Radiosity Methods," *Computer Graphics*, Vol. 21, No. 4, July 1987, pp. 311-320.

14. Weghorst, Hank, Gary Hooper, Donald P. Greenburg. "Improved computational methods for ray tracing" *ACM Transactions on Graphics*, Vol. 3, No. 1, January 1984, pp. 52-69.

15. Whitted, Turner, "An Improved Illumination Model for Shaded Display," *Communications of the ACM*, Vol. 23, No. 6, June 1980, pp. 343-349.