A new technique for computer simulation of illuminated spaces

Gregory J. Ward
Francis M. Rubinstein
Lighting Systems Research Group
Lawrence Berkeley Laboratory
Bldg. 90, Rm. 3111
1 Cyclotron Rd.
Berkeley, CA 94720
(415) 486-4757

ABSTRACT

An advanced ray-tracing technique for computing luminances in unempty spaces is presented. The method is a hybrid of finite element analysis and Monte Carlo techniques. Interreflection between specular surfaces of complex geometries can be computed to any desired accuracy. Applications to synthetic imaging for lighting design are highlighted.

Introduction

Recent research in illuminating engineering has demonstrated the power of the computer for predicting illumination levels in interior spaces[1]. Practical techniques based on finite element analysis have been developed for computing direct illumination and interreflection in rooms with simple geometry and Lambertian surfaces. Finite element analysis divides surfaces into sections that are characterized by their influence on one another[2]. The problem of flux transfer between diffuse surfaces can then be transformed into a set of solvable linear equations. Although the matrix is often too large for direct solution, iterative techniques provide an accurate approximation of final illuminance levels[3]. Efforts to extend this approach to the solution of non-Lambertian environments have proven to be extremely difficult[4]. Unempty spaces are also a problem for flux transfer techniques because of the difficulty in determining which surface elements "see" each other, and the relation of complexity to the square of the number of surface elements. Intervening objects must also be broken into surface elements for such a lighting calculation, and elements on all surfaces must be smaller than those used in the unobstructed case to attain the same level of accuracy.

Monte Carlo techniques have been applied to lighting calculation in an effort to overcome some of the limitations of finite element analysis[5,6]. In a pure Monte Carlo method, light "particles" are followed stochastically through the environment. Light is emitted, reflected and refracted according to probabilities defined by the characteristics of each surface. This technique places few constraints on the behavior of light or its interaction with the environment. Specular surfaces and intervening objects are handled in the same manner as diffuse surfaces and simple geometry. As a statistical method,

the accuracy of the calculation is proportional to the square root of the number of samples. Unfortunately, the number of samples required for acceptable accuracy is usually large, and the corresponding calculation is slow.

The method we present borrows from both finite element analysis and Monte Carlo techniques to provide a powerful approach to the simulation of illuminated spaces. It is based on a technique originally developed for computer graphics that we refer to in this paper as "ray-tracing"[7], popular for its realistic display of geometric models. Although accuracy is not strictly a requirement for realism in computer graphics, the two are certainly related. Realism without accuracy provides a convincing display that is meaningless (eg. office furniture with faked illumination), and accuracy without realism provides a meaningful display that is not interesting (eg. an empty room). We obtain both accuracy and realism using an enhanced ray-tracing technique that predicts both diffuse and specular interreflection in unempty spaces.

Background

Ray-tracing is a method for computing luminance by following light backwards from the point of measurement to the source(s). Each "ray" can be thought of as the luminance value that results either directly from an emitting surface, or indirectly from a reflecting surface*. Computing luminance from a reflecting surface requires convolving the surface reflectance-distribution function[8] with the luminous flux arriving at the surface. New rays are traced to determine the necessary luminance values, making the process recursive. Because ray-tracing does not impose any restrictions on the surface reflectance-distribution, specularity and other phenomena can be readily incorporated in the model. From luminance, most other lighting metrics, such as illuminance, visibility level (VL), equivalent sphere illumination (ESI), contrast and visual comfort probability (VCP) can be derived.

In this paper, we focus on a particular application of computed luminance -- synthetic imaging. The synthetic image, a two-dimensional map of calculated luminance values, is a new and important development in computer-aided lighting design[9]. The distribution of light in a complex environment with specularity and intervening objects is difficult to evaluate for lighting quality, even if it can be calculated. By using the computer to simulate a photograph of the proposed lighting environment, the evaluation can be left to the lighting designer. A photograph contains a tremendous amount of information, yet it is easily digested by both lighting experts and their clients.

Computer simulation of illuminated spaces involves two main areas of modeling, surface geometry and surface reflectance. In addition, synthetic imaging requires the modeling of a camera. Ray-tracing offers advantages in each of these areas.

Surface geometry

The comparatively simple operation required for geometric modeling with ray-tracing is the intersection of a surface and a line. The parametric equations for the ray are substituted into the surface equation, resulting in an equation of one variable (distance). The smallest positive root of this equation is the intersection of the ray with

* The issue of color will be addressed in a later section.

that surface. There are many useful shapes with easily-solved ray-surface intersection equations. A plane reduces to a simple linear equation, and quadric surfaces (spheres, ellipsoids, cones, etc.) result in a quadratic form. Boundary restraints often supplement the basic surface equation (eg. polygons rather than planes). The greater variety of surface shapes permitted by ray-tracing allows more compact and therefore simpler geometric descriptions.

Surface reflectance

Ray-tracing places few restrictions on the interaction of light with surfaces. For diffuse reflection, outgoing luminance has a simple relation to illuminance, which can be computed from rays traced to the light source(s). Rays traced in the reflected direction can be used to compute specular components. Following light backwards permits accurate computations of luminance because the measurement location and direction determine the appropriate rays to follow.

Camera

The perspective projection of a scene into two dimensions is straightforward in a ray-tracing scheme. A single point acts as the focus for luminance values (rays) on a two-dimensional map (the image plane). The location of the focal point and image plane determine the perspective view for the image. Each luminance value on the image plane is determined by a ray traced from that location through the focal point into the scene. The projection into two dimensions is implicit in this process. It is not necessary to transform or clip objects in the scene, or perform any separate hidden-surface removal. Ray-tracing also permits more sophisticated modeling of the camera, such as depth of field and motion blur[10].

Method

The basic method for tracing a ray is deceptively simple:

1. Determine which surface the ray intersects.
2. Calculate luminance in the ray's direction, possibly tracing new rays.
3. Return the computed value.

The lighting calculation takes place in one or more routines that determine outgoing luminance based on incoming luminance. The difficulty of the calculation and the number of incoming luminance values (rays) needed depends on the choice of reflectance and lighting models. A basic scene with the rays necessary to compute a single luminance value is shown in Figure 1.

Ray intersection

Regardless of the lighting model used, the speed of overall computation depends heavily on ray-scene intersection. The intersection calculation determines what surface shapes can be represented, and how long it takes to trace each ray.

For a scene composed of multiple surfaces, an intersection test must be performed for each surface in the ray's path, until the closest point of intersection is found. Several methods for quickly identifying which surfaces are in the path of a ray have been studied[11]. We use an octree sorting method developed by Glassner[12].

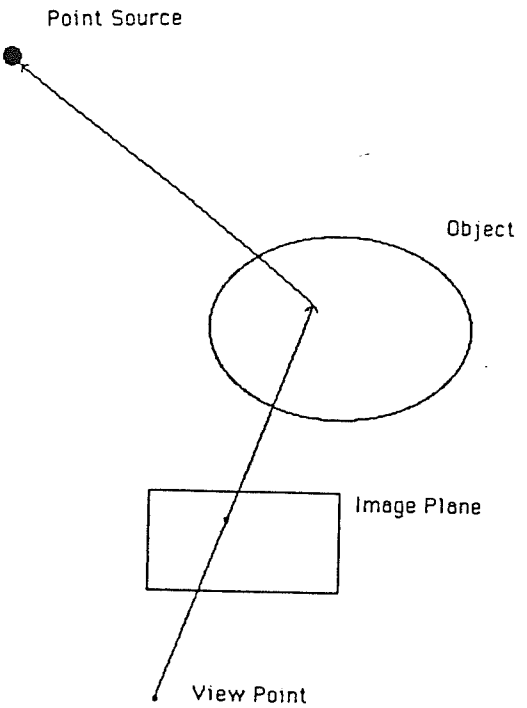

Figure 1

The scene space is recursively divided into cubes. Each tree cube branches into eight sub-cubes, hence the name octree. A leaf cube (a cube that has not been subdivided) is called a voxel, and contains a set indicating which surfaces in the scene penetrate it. To build the octree, we start with an empty cube encompassing the entire scene, and add to it one surface at a time. The algorithm for adding a surface to a cube is as follows:

```
AddSurface(Cube, Surface):
    if Intersect(Cube, Surface) then
        if not Leaf(Cube) then
            foreach SubCube in Cube do
                AddSurface(SubCube, Surface)
        else
            if SetSize(Cube.Set) < N or CannotDivide(Cube) then
                AddElement(Cube.Set, Surface)
            else
                SubDivide(Cube)
                foreach SubCube in Cube do
                    foreach Element in Cube.Set do
                        AddSurface(SubCube, Element)
                    AddSurface(SubCube, Surface)
```

The test for whether a cube contains (any part of) a surface, Intersect(Cube, Surface), will be different for each surface type. The user-selected integer N is the maximum set

size for a leaf. Another test, CannotDivide(Cube), prevents cubes from being made smaller than a certain size. This avoids infinite recursion at points where more than N surfaces intersect (eg. the tip of a pyramid).

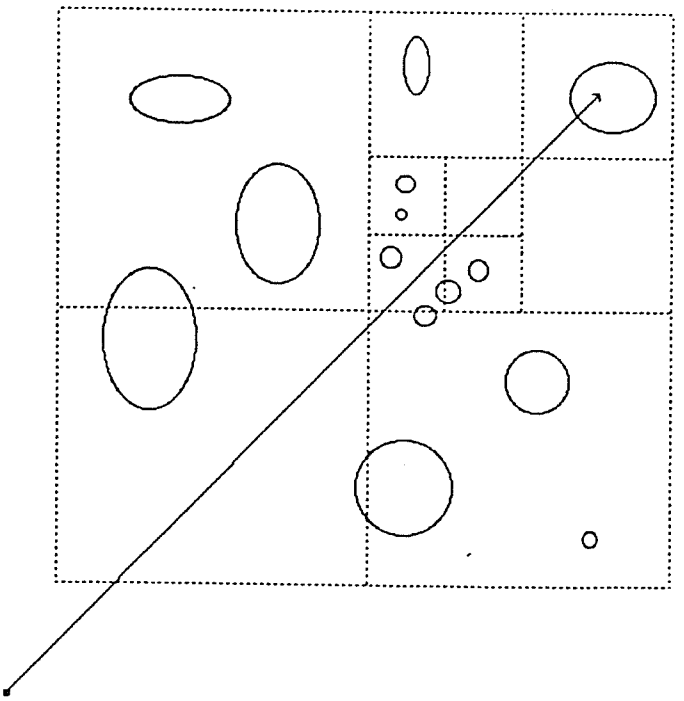

Figure 2

Using the octree, rays are followed through the voxels along its path and tested against the set of surfaces in each (Figure 2). The total number of voxels is proportional to the number of surfaces, and the voxels along an arbitrary linear path is roughly the cube root of the total. Since most of the voxels contain fewer than N surfaces, an average intersection calculation involves a constant times the cube root of the total number of surfaces. This means an average intersection calculation in a scene with 1000 surfaces will take only twice as long as a scene with 125 surfaces. Using this method, extremely complex environments can be ray-traced efficiently.

Direct illumination

To compute the contribution of direct illumination to the outgoing luminance at a surface element, we use the locations and sizes of the light sources in the scene to trace a ray to each source and determine whether the surface element is in shadow. If the surface element is illuminated by a source, the size, distance, brightness, and direction of the source together with the surface reflectance-distribution determine that source's direct contribution to the outgoing luminance. If the surface element is completely blocked from seeing a source (in shadow), that source will have zero contribution towards outgoing luminance. If the surface element is partially occluded, the

contribution to outgoing luminance will be fractional, resulting in a penumbra. To accurately model area sources, a Monte Carlo technique is used. Instead of sending rays directly to the center of the light source, rays are randomly distributed over its disk[10]. In the partially occluded case, such a ray will be blocked with a probability exactly proportional to the degree of occlusion. Therefore, an average of these rays will approach the correct illumination value in accordance with the central-limit theorem.

Indirect illumination

Indirect illumination is the component of luminous flux arriving at a surface that did not travel directly from any light source. We will first present a simple calculation of indirect illumination, and then show how to make it efficient.

The basic method for computing indirect illumination is to sample *reradiated* luminance values over a hemisphere defined by the surface element position and normal. The luminance integral will be convolved with the bidirectional reflectance-distribution function for the surface:

$$L_r(\theta_r, \phi_r) = \int_0^{2\pi} \int_0^{\frac{\pi}{2}} L_i(\theta_i, \phi_i)\, f(\theta_i, \phi_i; \theta_r, \phi_r) \cos\theta_i \sin\theta_i \, d\theta_i \, d\phi_i \qquad (1)$$

where: $L_r(\theta_r, \phi_r)$ = luminance radiated in direction (θ_r, ϕ_r)

$L_i(\theta_i, \phi_i)$ = luminance incident from direction (θ_i, ϕ_i)

$f(\theta_i, \phi_i; \theta_r, \phi_r)$ = bidirectional reflectance distribution function

θ's are polar angles from surface normal

ϕ's are azimuth angles in the surface plane

The bidirectional reflectance-distribution function can be broken into diffuse and specular (Lambertian and non-Lambertian) components[13]:

$$f(\theta_i, \phi_i; \theta_r, \phi_r) = f_s(\theta_i, \phi_i; \theta_r, \phi_r) + \frac{\rho_D}{\pi} \qquad (2)$$

where: $f_s(\theta_i, \phi_i; \theta_r, \phi_r)$ = specular bidirectional reflectance distribution function

ρ_D = diffuse reflectance

Thus the luminance integral (1) can also be split into diffuse and specular terms. The specular term can usually be computed with a few rays, as we will show in a later section.

Diffuse interreflection

The diffuse term is equivalent to an illuminance calculation, which requires many rays to be sampled in all directions. Since the integrand is weighted by $\cos\theta$, the best a priori distribution of samples is uniform over the projected hemisphere (Figure 3).

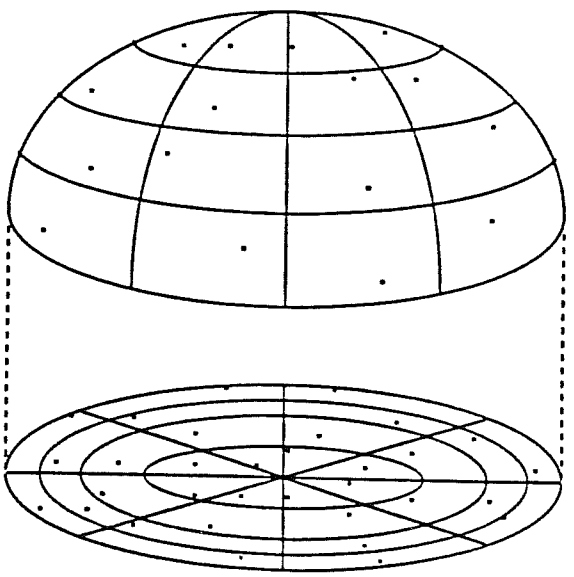

Figure 3

Each ray will carry equal weight, and an average of the incident luminance values will approach the correct illuminance value as the number of samples tends to infinity:

$$E = \frac{\pi}{2n^2} \sum_{j=1}^{n} \sum_{k=1}^{2n} L(\theta_j, \phi_k) \tag{3}$$

where: E = illuminance

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

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

X = uniform random variable between 0 and 1

$2n^2$ = total number of samples

Since this is a Monte Carlo technique, the error in illuminance is inversely proportional to n in the above equation. Dividing the hemisphere in this fashion is called stratified sampling[14,15], and stabilizes the integral for faster convergence.

For simple scenes with uniform lighting and few obstructions, fewer samples are required (ie. less than 50). In contrast, complex scenes may require several hundred samples for convergence. If this sampling is done during every computation of luminance, the time for a synthetic image will be measured in weeks.

We can reduce the number of illuminance calculations by storing values and reusing them locally. Indirect illumination tends to vary more slowly than the direct component because reradiating surfaces are more evenly distributed than light sources[1]. Therefore, interpolation of indirect illumination over surfaces will generally be satisfactory, since sharp boundaries (shadows) are not a concern. Storing and interpolating indirect values adds complexity to the method, but has the potential for greatly improving efficiency.

Many different schemes for interpolation are feasible. The surfaces could be divided into sections in a gridlike fashion, and the indirect contribution to illuminance could be computed at each grid point. The values could then be interpolated bilinearly over the surfaces. This would be analogous to a finite element method of calculation where each surface element is made a certain size. There are two main disadvantages to this scheme. The first disadvantage is that it is difficult to sensibly divide surfaces whose geometry is non-planar, or whose size is small and quantity large. The second disadvantage is that surfaces or surface sections that are near other surfaces should have a finer grid than surfaces which see no nearby objects. This is because the indirect illumination on a surface is affected primarily by neighboring objects. If those objects are close, the shadows and reflections they cause will change on a smaller scale.

Our interpolation method dynamically places indirect illuminance values on surfaces in a way that is not dependent on surface representation. An estimate of the expected change in indirect illuminance due to a change in surface element location

serves as a basis for value placement and interpolation. The estimate is based upon a simplified model, which is developed in the appendix.

The resulting interpolation formula for indirect illuminance at a point $\vec{P}$ is a weighted mean:

$$E(\vec{P}) = \frac{\sum_{i \in s} w_i(\vec{P}) E_i}{\sum_{i \in s} w_i(\vec{P})} \tag{4}$$

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 = average distance to adjacent objects at $\vec{P}_i$

S = $\{ i : w_i(\vec{P}) > k \}$

k = user selected constant

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

Both the interpolation error and the average distance between illuminance values will be inversely proportional to k. A new indirect illuminance value is calculated wherever the set S is found to be empty. (The method used to find members of S is also given in the appendix.)

Specular interreflection

Unless a surface is perfectly diffuse (Lambertian), the outgoing luminance will have a directional, or specular, component. This component can be computed using the specular bidirectional reflectance-distribution function (2). In the extreme cases of mirrors and glass surfaces, the diffuse reflectance will be zero and the specular reflectance-distribution will be a Dirac delta function. A mirror reflectance-distribution will have an integratable singularity in the reflected direction, and a glass surface will have singularities in both the reflected and refracted directions. For glossy or translucent surfaces, the specular reflectance-distribution will be finite but still highly peaked.

In ray-tracing, computing specular components is both simple and efficient. For a mirror, a single ray is traced in the reflected direction. For glass, one ray is traced in the reflected direction and one ray is traced in the refracted direction. For glossy and translucent surfaces, rays are scattered about the reflected and refracted directions by interpreting the reflectance-distribution as a probability density function.

Multiple interreflections

Recursion is used to compute multiple reflections between surfaces. Since ray-tracing is defined in terms of itself (ie. luminance values come from luminance values), some criteria must be established to limit the process.

For specular reflections, the cumulative weight of a ray is monitored. This weight is related to the ray's fractional contribution to the final value. It is computed at each level of recursion by multiplying the weight of the parent ray by the transmittance or reflectance of the surface. If a ray's weight is below a preset level, it is not traced. In this way, only the rays most likely to have a significant effect on the final value are followed.

For diffuse reflections, recursion is more complicated. Since the diffuse indirect calculation requires tracing many rays at each surface, limiting diffuse propagation is critical. After a fixed (usually small) number of bounces, a constant value is used for the diffuse component. This value can be the result of a zonal cavity approximation, or zero. The observation that higher bounces contribute less to the final value allows us to make a simple optimization. If the average surface reflectance is less than 50 percent, twice as much error can be tolerated in each successive bounce. Only one quarter as many samples are necessary to obtain this, since the error is inversely proportional to the square root of the number of samples. If interpolation is used, the value for k (4) can be decreased by 30 percent at the same time the number of samples is cut in half. The combined error of interpolation and Monte Carlo integration will then double. In either case, one fourth as much computation will be required for each successive bounce.

Color

The discussion so far has concentrated on monochromatic environments, but spectral effects can be simulated as well. Like most lighting calculation methods, ray-tracing models color by dividing the visible spectrum into bands.

We use a basic treatment of color that breaks the spectrum into three components, the CIE tristimulus values X, Y, and Z. The Y value (green) is luminance, which we have already discussed. The X and Z values (red and blue) are treated in parallel with Y. Any time two luminance values are added together, two X values are added and two Z values are added. Reflectance functions are tripartite, and modify the tristimulus values independently.

For a truly accurate treatment of color (and luminance), the spectrum must be divided into *many* bands. The size and distribution of the bands will determine the accuracy of the calculation. If luminance is the most important result, the narrowest bands should be near the peak of $V(\lambda)$, such that all bands define equal areas under the curve. Spectral power distributions are used for light sources, and spectral reflectance functions are used for surfaces. Even with many spectral bands, the calculation will be dominated by ray intersection testing, and affected only slightly by the treatment of color.

Results

The methods discussed have been incorporated into a family of computer programs running under the UNIX† operating system. They have been used to model a variety of lighting environments. What follows is a collection of synthetic photographs produced on VAX 11/780 and MicroVAX II computers. The resolution, execution times, as well as the number of rays traced is given where available*. Note that the time required is closely tied to the resolution of the image.

Figure 4 shows a roadway scene with and without street lights. Visibility for nighttime driving depends critically on road surface conditions. Reflection from moisture or ice will dramatically change the visual appearance of a roadway, and these changes can be accurately modeled in a ray-tracing calculation. Candlepower distributions were used for the street lights as well as the car headlights. The images were produced at a resolution of 1024 by 1024 and took about three CPU hours each.

Figure 5 shows a desk top with lamp. The specularity of the surface results in a veiling reflection that would impair visibility. This image has a resolution of 3000 by 2000 and took 30 CPU hours to generate. Figure 6 shows a simple office scene under fluorescent lighting. This image has a resolution of 2048 by 2048, and took 20 CPU hours. Figure 7 shows an office illuminated by daylight through venetian blinds. The distribution of light from the window was computed in a separate ray-tracing calculation which took about 40 minutes, and considered both sun and sky components.

Figure 8 shows three views of an ice cream store. All three images have a resolution of 1500 by 1000. Figure 8a models the five incandescent lights as point sources, and required 12 CPU hours to trace 4.5 million rays. Figure 8b models the lights as area sources, and took 27 CPU hours to trace 14.5 million rays. Figure 8c shows indirect cove lighting, and took 38 CPU hours to trace 7.5 million rays.

Discussion

Although the compute times for the images shown may seem prohibitive for practical lighting design, there are several factors to consider. First, the image generation time is roughly proportional to resolution. Therefore, a 256 by 256 image that is perfectly acceptable for lighting evaluation would take only one sixteenth as long to compute as the 1024 by 1024 image produced for publication. In the roadway example, such an evaluation image requires less than 15 minutes to compute, and a single luminance value can be calculated in a hundredth of a second. The second consideration is the growing power and availability of personal computers. The computations that seem impractical today will in all likelihood be commonplace tomorrow. A third consideration is an alternate approach to synthetic imaging -- adaptive refinement.

Adaptive refinement is an interactive form of computer simulation where the user directs the generation of an image[16]. Each time the user specifies a new view, the program quickly produces a rough image from a few rays, then progressively improves the resolution of the image while the user looks on. At any time, the user may select a different view or specify a particular section of the image to improve. This process

†UNIX is a trademark of AT&T

*Execution times are scaled to equivalent CPU hours on an 11/780.

proceeds quickly, permitting a scene to be evaluated for lighting quality from many directions in a matter of minutes. This is particularly important where specular surfaces are involved, since view direction plays a critical role in scene appearance.

Another means of reducing computation time is optimized sampling, the selection of samples based on statistical tests of variance. This has the potential for reducing the number of rays required for a given accuracy by concentrating samples where the luminance varies significantly. This type of optimization is used on the image plane to reduce generation times by a factor of three to ten. Optimized sampling can also be used in the calculation of indirect illuminance by concentrating rays on the brightest or most rapidly changing parts of the hemisphere.

Currently, the time required by the calculation increases linearly with the number of sources, a condition that is unacceptable for scenes with flood lighting, etc. One possible optimization would limit calculations to the area each source affects. For spot lighting, the solid angle of the source would be checked before shadow testing. If a point is not within the illumination angle of the source, no test is necessary. For area lighting, a source at a distance greater than its radius of influence can be ignored.

Our ray-tracing method has been checked against certain solvable cases, such as a Lambertian sphere on an infinite plane, but has not undergone more rigorous empirical tests. Part of the difficulty is getting reliable luminance data from a well-defined environment. There is currently little data on the reflectance-distributions of common surface materials. Recent advances in automated measurement together with improvements in CCD camera technology may soon make it possible to gather the enormous amount of data necessary for a full validation.

Conclusion

We have presented a ray-tracing method for calculating luminance and producing realistic images of complex lighting environments. The methods described can treat spaces containing many objects of complex shape with surfaces that have both specular and diffuse properties. The length of time required to compute various images was found to be strongly influenced by the resolution of the image and the number of light sources and weakly influenced by the number and complexity of the objects in the scene. Compute times generally ranged between minutes and hours on a micro-based workstation, indicating the method's applicability to the newest generation of personal computers. Further improvements in technique together with expected advances in computer technology will greatly expand the capabilities of this type of lighting calculation. Eventually there will be no need to guess what a design will look like; creation and evaluation will be done interactively on a computer workstation.

Acknowledgements

Our thanks go to the reviewers, and to Robert Clear for his valuable contributions. This work was supported by the Assistant Secretary for Conservation Renewable Energy, Office of Building Technologies, Building Equipment Division of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

References

1. DiLaura, David L. 1979. "On a new technique for interreflected component calculations" *Journal of IES* 9(no. 1):53-59

2. Siegel, R., J. R. Howell. 1972. *Thermal Radiation Heat Transfer* New York: McGraw-Hill

3. Selkowitz, Stephen, Jong-Jin Kim, Mojtaba Navvab, Frederick Winkelmann. 1982. "The DOE-2 and Superlite daylighting programs" Lawrence Berkeley Laboratory Report 14569

4. Immel, David S., Donald P. Greenburg, Michael F. Cohen. 1986. "A radiosity method for non-diffuse environments" *Computer Graphics* 20(no. 4):133-142

5. Tregenza, P. R. 1983. "The Monte Carlo method in lighting calculations" *Lighting Research and Technology* 15(no. 4):163-170

6. Stanger, Dan. 1984. "Monte Carlo procedures in lighting design" *Journal of IES* 13(no. 4):368-371

7. Whitted, Turner. 1980. "An Improved Illumination Model for Shaded Display" *Communications of the ACM* 23(no. 6):343-349

8. Nicodemus, F. E. 1970. "Reflectance nomenclature and directional reflectance and emissivity" *Applied Optics* 9(no. 6):1474-1475

9. DiLaura, David L., Denis P. Igoe, Richard G. Mistrick. 1985. "Synthetic photography" *Lighting Design & Application* 15(no. 8):24-27

10. Cook, Robert, Thomas Porter, Loren Carpenter. 1984. "Distributed ray tracing" *Computer Graphics* 18(no. 3):137-147

11. Weghorst, Hank, Gary Hooper, Donald P. Greenburg. 1984. "Improved computational methods for ray tracing" *ACM Transactions on Graphics* 3(no. 1):52-69

12. Glassner, Andrew S. 1984. "Space subdivision for fast ray tracing" *IEEE Computer Graphics and Applications* 4(no. 10):15-22

13. Spencer, Domina Eberle, Eugene A. Gaston. 1978. "Diffuse and specular components of reflectance: why and how" *Journal of IES* 7(no. 4):240-248

14. Kajiya, James T. 1986. "The rendering equation" *Computer Graphics* 20(no. 4):143-150

15. Lee, Mark E., Richard A. Redner, Samuel P. Uselton. 1985. "Statistically optimized sampling for distributed ray tracing" *Computer Graphics* 19(no. 3):61-67

16. Bergman, Larry, Henry Fuchs, Eric Grant. 1986. "Image rendering by adaptive

refinement" *Computer Graphics* 20(no. 4):29-37

Appendix

This appendix describes the method used for interpolating indirect illuminance, and gives its derivation.

Illuminance interpolation

Imagine a surface element in the center of a sphere. Half of the sphere is bright, the other half is dark. The surface normal faces the dividing line between the two halves. This example will serve as a worst case for the purposes of error prediction. For motion along the line perpendicular to the brightness division, the change in illuminance is given by the partial differential with respect to x. For rotation around the axis perpendicular to the normal and parallel to the brightness division, the change in illuminance is equal to the partial with respect to ξ. The first order estimate of error due to changes in position is given by the Taylor expansion for a function of two variables:

$$\varepsilon \leq \left| \frac{\partial E}{\partial x}(x - x_o) + \frac{\partial E}{\partial \xi}(\xi - \xi_o) \right| \tag{5}$$

$$\varepsilon \leq \frac{4}{\pi}\frac{E_o}{R}\left| x - x_o \right| + E_o \left| \xi - \xi_o \right|$$

Generalizing this equation to motion in any direction, x and ξ can be replaced by vector-derived values:

$$\varepsilon(\vec{P}) \leq E_o \left[\frac{4}{\pi}\frac{\| \vec{P} - \vec{P}_o \|}{R_o} + \sqrt{2 - 2\vec{N}(\vec{P})\cdot\vec{N}(\vec{P}_o)} \right] \tag{6}$$

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

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

The *valid domain* of the computed illuminance E_0 is the set of all surface locations $\vec{P}$ where this expression is less than a specified value. The average radius R_0 can be computed from the distances to adjacent surfaces, given by the ray-tracing process during illuminance calculation. A harmonic average is used, since distance appears in the denominator of the error term.

By calculating an adequate number of indirect illuminance values for a surface, we can interpolate between values to completely describe the indirect component over that surface. The formula for interpolation uses the error estimate (6) in a weighted mean, which was given earlier (4). The set S represents those illuminance values which contain the point $\vec{P}$ in their valid domain. If S is empty, we must compute a new E_i at this location. The constant k is selected by the user, and will be related to the interpolation

error. The error estimate for a single value (6) is used to derive the of interpolation for the split sphere example:

$$\varepsilon(\vec{P}) \leq 1.4\frac{\bar{E}}{k} \qquad (7)$$

where: $\bar{E}$ = average E_i

For efficient interpolation, we need a fast method for determining the set S. Here too, an octree is used as an aid to sorting and searching. We store each illuminance value E_i in the cube containing the value's position $\vec{P}_i$ with a size (side length) greater than twice but not more than four times the value's valid domain R_i/k. This guarantees that a stored illuminance value will not be valid in more than eight cubes on its own octree level.

Each cube in the tree will contain a (possibly empty) list of the illuminance values it contains, and a (possibly nil) pointer to its eight children. To search the tree for values whose valid domain may contain the point $\vec{P}$ we proceed as follows:

```
For each illuminance value in this cube
    If w_i(P)>k
        Include value
For each child
    If P is within the child cube's size of its center
        Recurse on child
```

This algorithm will not only pick up the cubes containing $\vec{P}$, but will also search cubes whose centers are within the cube size of $\vec{P}$. In this way, we will look at all lists that might have illuminance values whose valid domain contains $\vec{P}$.

Discussion

Most of the elements in the methodology employed by the authors were considered by the discusser's own work on computer graphics. The ray tracing method is a relatively straightforward technique with guaranteed results. The main issue here is the computing time required to generate a picture. A second drawback of ray tracing is that every time a new view is needed, the process must start almost from the beginning. It is very time consuming in this respect.

Now, a few questions about the paper:

(1) Is the relationship between physical luminance and psychological brightness considered?

(2) What is the symbol structure used by the computer program?

(3) What level of computer knowledge does the user need for the program?

(4) How user friendly is the program? Or, how easy is it for a user to input data?

(5) What are the hardware and software (excluding the program itself) requirements for the program?

(6) The discusser would like the authors' comments on their objectives. Is the work for academic interest, or is it for practical application? If it is for the former, I believe others have done something similar, although not in quite the same way. This by itself is of interest. If it is for the latter, how practical is it to use this program in a design production environment?

Peter Y. Ngai
Peerless Lighting Corporation
Berkeley, CA

I am tremendously impressed by the potential of this new technique for computer calculation of luminances that represent the view of an observer or "camera". The technique calculates the luminance at a point on a surface as seen from the location of the observer; the accuracy of the calculation will be dependent, as the authors point out, on the number of rays to be traced, the correctness of the directional reflectance characteristics, and the correctness of the luminaire candela distribution. It is a challenge therefore to our photometric laboratories to add the ability to determine and classify the directional reflectance properties of commonly used surfaces. It is a challenge to provide photometric data in a form that can be used when the luminaire is large compared to the distance of the object or surface from that luminaire.

We should not be deterred by the long computational time quoted in this paper but should expect that the computational time and accuracy will depend upon our ability to simplify the calculational procedure and relate it to the accuracy level involved. My own experience with reverse ray tracing to predict the photometric output of a proposed luminaire design indicates to me that very high accuracy is possible -- but frequently not needed.

I wish to point out to the authors that in addition to the need to create a rendition of complete scenes to evaluate the suitability of a proposed lighting design that there is also the need to calculate the luminances of a small area of that scene to provide the data needed to enter a visibility model and determine object or target visibilities. In the foveal visibility models that are available today, only a small part, a one degree or less field, needs to be simulated with accuracy. I therefore encourage the authors to pursue the interactive concept described in the paper.

My only question to the authors relates to the availability of part or all of the family of computer programs currently running. Since this work was supported by the U.S. Government, are the programs in the public domain and if so, how can they be obtained in a form so that others can work with and modify or extend the source code?

Merle E. Keck
Lighting Consultant

Vicksburg, MS

Response

The discussants' comments are greatly appreciated. We will address each of Mr. Ngai's questions in order.

The psychological perception of brightness is important not only to computer simulation, but to lighting evaluation in general. Most researchers in human vision agree that the perception of color and brightness is affected dramatically by the visual environment. However, the exact correlation is not known. The programs currently simulate images as they would be produced by a camera, without compensating for or otherwise considering the visual interpretation. The ultimate solution to the problem of psychological perception may be to reproduce the entire scene, photon for photon. Compensating an inferior simulation is much like adjusting the tone control on a pocket radio -- the results will never be satisfactory.

The programs use a hierarchical symbol structure for their input. The exact hierarchy is left to the user. Typically, a room will contain furniture or sets of furniture; the furniture will have associated objects, and the objects may have additional structure. In addition to surfaces, scene descriptions contain material definitions and procedural objects (objects generated by specialized programs).

The computer knowledge required to operate the programs is a basic understanding of files and text editors. Experience writing shell scripts (batch files) is helpful, but not necessary.

The input for the programs can be somewhat difficult to construct, requiring the calculation of surface locations in space. To simplify this process, object generators for boxes, prisms, objects of rotation, and arbitrary curved surfaces are provided. A three-dimensional geometric editor was developed for the Apple MacIntosh, providing visually-oriented creation and modification of objects. The output of the editor serves as input to the ray-tracing software, which does the actual rendering.

The minimum hardware and software required by the programs is a microcomputer workstation with a UNIX-compatible operating system. Some form of color output is desirable; a high-resolution display with 8 or more bit planes is optimal. If other CAD software is available, its output can be adapted and used as input to the programs.

The software system was originally designed as a research tool, and that remains its primary function. Ray-tracing techniques have existed for several years, but the ability to compute both diffuse and specular reflections between thousands of surfaces has not. More research remains to be done, particularly in the area of light source contributions and sample optimization, before the method will be robust enough for general lighting design.

We are grateful to Mr. Keck for his encouragement, and would like to echo his plea for better photometric data. We believe that expensive goniophotometric measurements can be avoided by classifying surfaces and parameterizing their characteristics. Using only a few well-chosen measurements of a given material, it should be possible to approximate the corresponding bidirectional reflectance-distribution function. We intend to demonstrate this approach on a common class of architectural materials, and verify the results with a goniophotometer.

Mr. Keck suggests a very interesting method of lighting simulation, adaptive refinement based on eye position. A sensor tracking eye movement could guide the generation of an image, and the display would respond immediately to a user's changing interest. Taking the idea to its logical limit, a form of head gear could be worn that would sense physical position and viewing direction as well. One can imagine walking around a simulated room, painting hypothetical walls and moving weightless furniture.

Although we have no immediate plans for releasing the software, the programs are currently being used and tested by the Architecture Department at UC Berkeley.

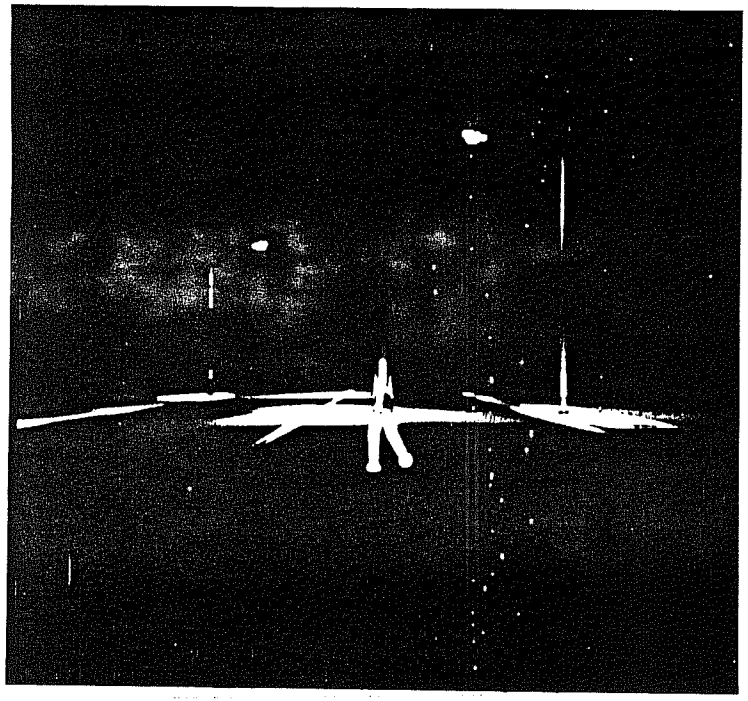

Figure 4a.

Figure 4b.

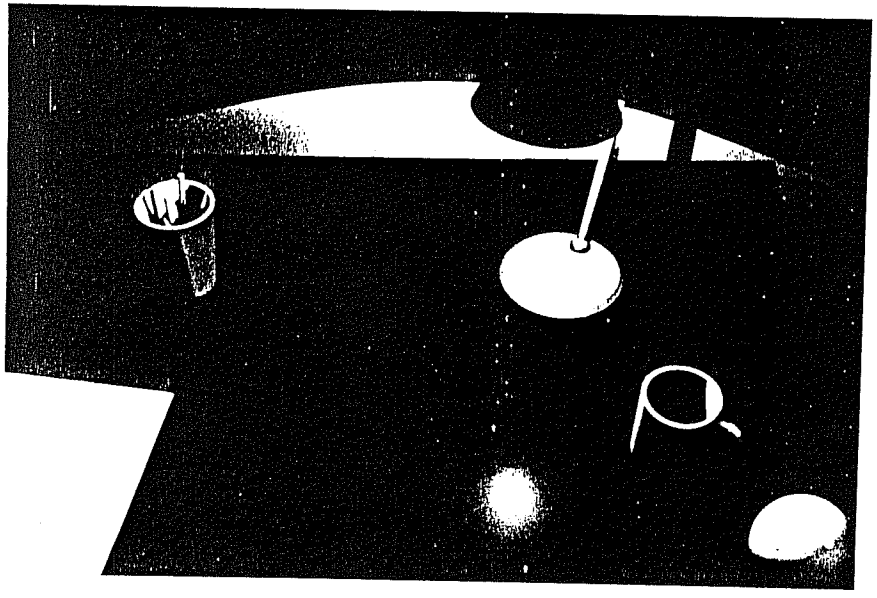

Figure 5.

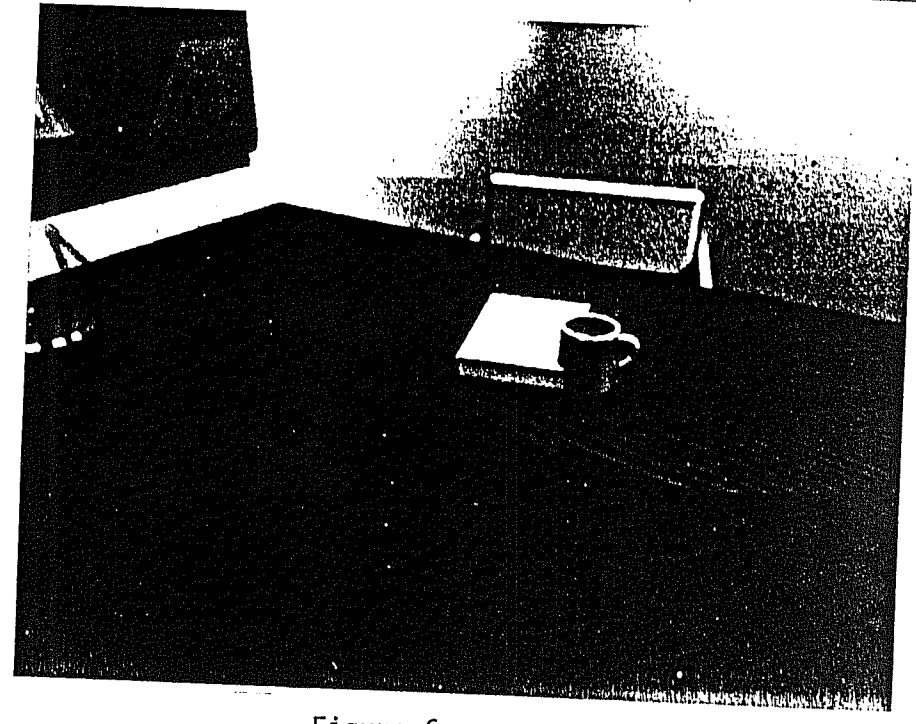

Figure 6.

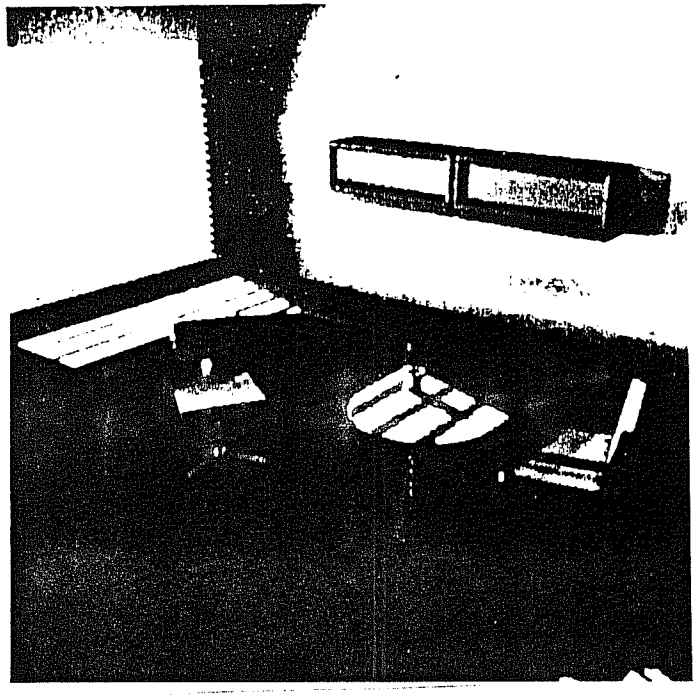

Figure 7.

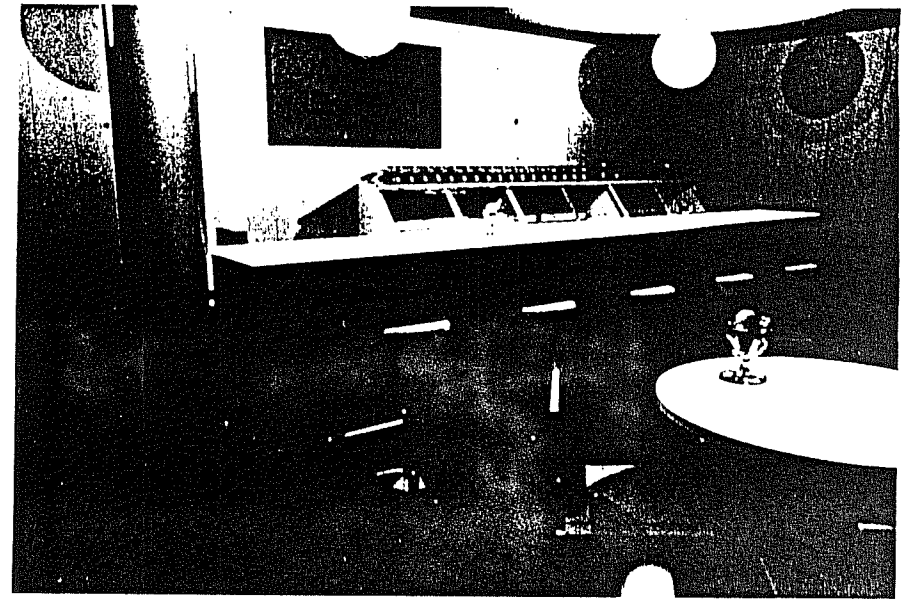

Figure 8a.

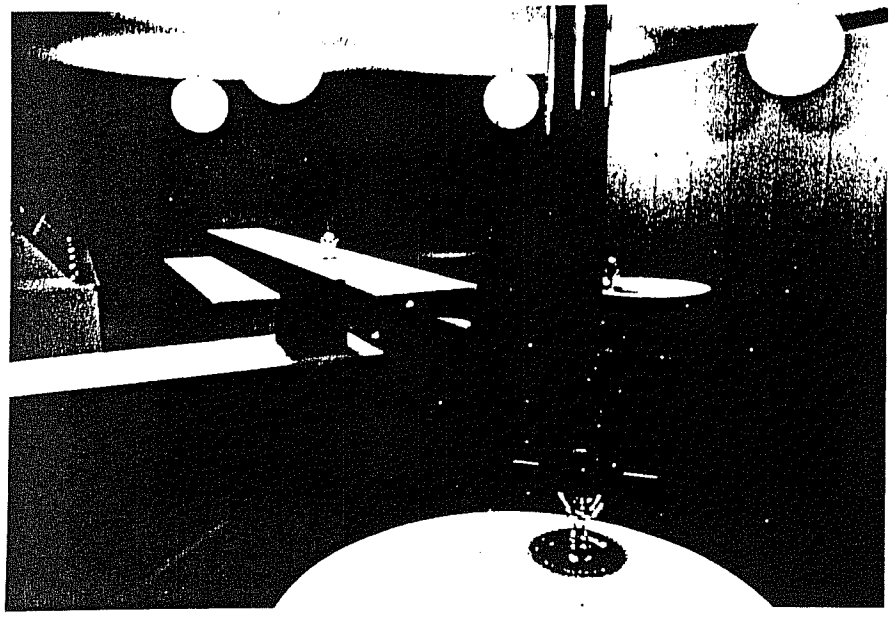

Figure 8b.

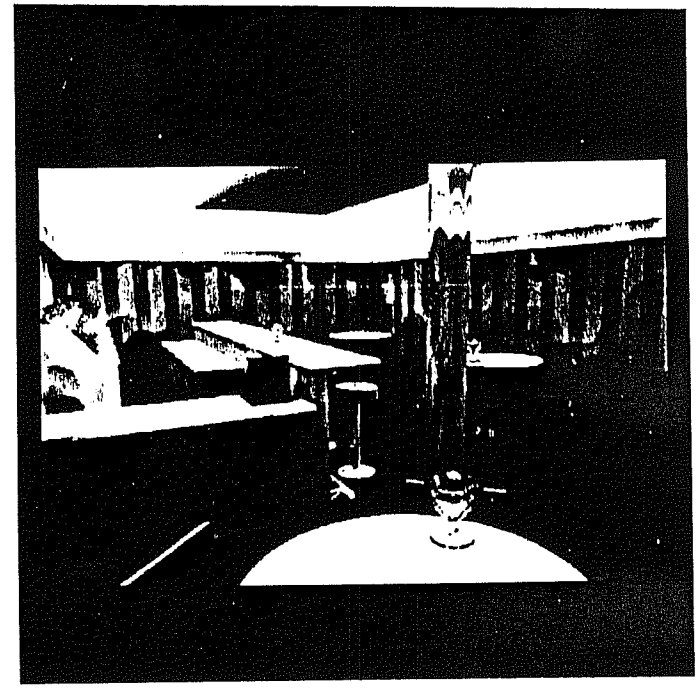

Figure 8c.