Bidirectional Estimators for Light Transport

Eric Veach
Leonidas Guibas

Computer Science Department, Stanford University

Abstract

Most of the research on the global illumination problem in computer graphics has been concentrated on finite-element (radiosity) techniques. Monte Carlo methods are an intriguing alternative which are attractive for their ability to handle very general scene descriptions without the need for meshing. In this paper we study techniques for reducing the sampling noise inherent in pure Monte Carlo approaches to global illumination. Every light energy transport path from a light source to the eye can be generated in a number of different ways, according to how we partition the path into an initial portion traced from a light source, and a final portion traced from the eye. Each partitioning gives us a different unbiased estimator, but some partitionings give estimators with much lower variance than others. We give examples of this phenomenon and describe its significance. We also present work in progress on the problem of combining these multiple estimators to achieve near-optimal variance, with the goal of producing images with less noise for a given number of samples.

1 Introduction

Many techniques have been proposed for solving the problem of global illumination in computer graphics. By far the simplest of these algorithms are the *pure Monte Carlo* (MC) methods. These methods have several other advantages: they guarantee that the *expected* value of the solution at each image pixel is correct (compared with the true mathematical solution); they require almost no storage beyond the scene model itself; and they can be applied to arbitrary surface geometries and reflectance functions in a clean, uniform way. The interface to the scene model is particularly nice—all operations access the scene as an object-oriented *black box*, allowing truly procedural geometric and reflection models. Pure MC methods do not suffer from many of the artifacts and limitations that must be addressed by radiosity techniques[8, 9, 10] ("blocky" appearance, Mach bands, missing shadows, limited reflectance models), making them an excellent choice for the validation of other methods.

However, Monte Carlo methods have one well-known drawback: *noise*. The focus of this research is to determine to what extent this noise is an inherent limitation. That is, how far can MC methods be taken in terms of

variance reduction, without adding bias to the solution? Many techniques for variance reduction have been described in the Monte Carlo literature[1, 2] (e.g. importance sampling, stratified sampling) and have long been used by the computer graphics community[3, 6, 7, 5]. Yet even with these techniques, there are many reasonable scenes for which current MC algorithms are not practical. Our goal is the development of new variance reduction methods that exploit the special properties of global illumination.

In this paper, we restrict ourselves to *pure* Monte Carlo methods for global illumination. These are methods which

- give an unbiased estimate at every pixel,
- have no correlation between the errors at different pixels,
- work for general surface geometries and reflectance functions, and
- do not require any data structures in object space (such as a subdivision of surfaces into patches).

For example, MC methods which express the solution as a linear combination of basis functions are not pure, since this introduces correlations between the errors at different pixels. Pure methods are attractive because the only image artifact is noise; thus if an image we compute does *not* appear to be noisy, we have strong reason to believe that it is correct. Pure methods include distribution ray tracing[3] and path tracing[6]. Many variants on these techniques are possible[4, 6, 5].

All *pure* MC techniques described in the literature have one feature in common: rays are traced only from the eye, not from the light sources. Techniques such as *light ray tracing*[15, 13], *bidirectional ray tracing*[16, 12], and *Monte Carlo radiosity*[14, 11] all use the light rays to deposit energy on surface patches. Since this requires a mesh in object space, these methods are not "pure" for the purpose of this paper. Also, these techniques do not extend well to environments with many small patches[17] or to non-diffuse surfaces[18, 19, 20].

Lafortune and Willems[21] have independently developed a "bidirectional path tracing" technique which uses some of the ideas presented in this paper. However their framework does not recognize explicitly the multiple estimators for each path length, or the problem of optimally combining them.

Let's consider a specific cause of noise in MC images: highly non-uniform incoming illumination. The problem is that the outgoing illumination L_o is essentially the product of the incoming illumination L_i with a reflection term; generally we can obtain accurate local information about the reflection term but not about L_i. For this reason, existing methods sample where the reflection term is large (importance sampling). However if L_i is highly non-uniform (for example 99% of the light comes from only 1% of the hemisphere of solid angles), this strategy is a poor predictor of the important sampling directions, leading to high variance.

In this paper we investigate pure MC methods which *balance* between sampling where the reflection function is large and where the incoming illumination is large. These methods build transport paths in two parts, one starting from a light source and the other from the eye. We show that there are k ways to evaluate the light flowing on transport paths of length k, according to where we break the path between the eye and light portions. We are also experimenting with techniques for *partitioning* the transport paths between the k methods to reduce the variance of our estimates.

Our results generalize the *direct lighting calculation*[6, 5], a common optimization for MC methods. Rather than following paths all the way back to the light sources, this optimization handles the last path segment specially. Our partitioning technique gives a rule for when the direct lighting calculation should be applied; there are some situations where it is not beneficial. More generally, our techniques address the problem of noise due to highly non-uniform *indirect* lighting. We demonstrate that noise from bright indirect light in typical MC images is due to following transport paths only from the eye.

This paper is organized as follows. Section 2 gives an outline of our rendering algorithm, along with several examples which demon-

strate how it works. Section 3 describes a reformulation of the rendering equation as an integral over rays. We have found this useful in describing and analyzing the algorithm. Section 4 discusses the problem of partitioning transport paths among the rendering methods to reduce variance, and describes several ideas we are currently experimenting with to solve this problem. Finally, the Appendix describes a recursive formulation of the bidirectional sampling, and gives some additional mathematical details.

2 Outline of the Algorithm

The desired value at a pixel P can be expressed as an integral

$$\int_{\Omega} f(x) d\mu(x)$$

over the space Ω of all transport paths x, where the weight $f(x)$ is proportional to the contribution made to P by the light flowing along x (see Sect. 3,4). The largest contributions typically come from short paths, so we can either ignore paths whose length exceeds some threshold, or use *Russian roulette*[4] to terminate long paths without adding bias. This lets us partition the estimate at P into a finite sum; we estimate separately the contribution due to each path length k.

To estimate the contribution for a particular k, we use MC integration (Sect. 4). This involves randomly generating a path x of length k which potentially contributes to P, and scoring the contribution $f(x)/p(x)$ where $p(x)$ is the differential probability with which we generated x. To reduce the variance, we repeat the whole process M times and take the average (see Fig. 1).

How should we go about generating paths of length k? In typical MC algorithms, paths are generated by following random bounces backward starting from the eye. The key feature of our algorithms is that they construct transport paths starting from *both* the light sources and the eye. The transport paths have

```
1  ESTIMATE-PIXEL(P)
2     S ← 0
3     for n ← 1 to M
4        for k ← 1 to Max-length
5           x ← CHOOSE-PATH(P, k)
6           S ← S + f(x)/p(x)
7     return S/M
```

Fig. 1. Simplified pseudocode for estimating the value at a pixel P. CHOOSE-PATH(P, k) generates a path x of length k which potentially contributes to P. $f(x)$ is the differential contribution to P of light flowing along x, and $p(x)$ is the probability density with which CHOOSE-PATH generates x. In practice, the estimates for each k are not independent; we can incrementally add a segment to partial paths from the previous step(s), and save the effort of generating an entire path each time.

the form

$$\mathbf{y}_0 \to \mathbf{y}_1 \to \cdots \to \mathbf{y}_n \\ \rightsquigarrow \mathbf{x}_m \to \mathbf{x}_{m-1} \to \cdots \to \mathbf{x}_0$$

consisting of a *light portion* $\mathbf{y}_0, \ldots, \mathbf{y}_n$ starting at a point $\mathbf{y}_0$ on a light source, followed by an *eye portion* $\mathbf{x}_m, \ldots, \mathbf{x}_0$ ending at a point $\mathbf{x}_0$ on the lens aperture. All $\mathbf{x}_i, \mathbf{y}_i$ lie on surfaces of the scene S (see Fig. 2).

The eye portion of a transport path is built by following a chain of m random bounces starting from the eye; this is "backward" relative to the direction light travels. Similarly the light portion is built by following n random bounces forward from the light source. The connecting segment $\mathbf{y}_n \rightsquigarrow \mathbf{x}_m$ is *not* chosen randomly; it is completely determined by the choice of $\mathbf{y}_n$ and $\mathbf{x}_m$. Of course it is possible that segment $\mathbf{y}_n \to \mathbf{x}_m$ is occluded, in which case no light flows along this path.

By controlling the number of steps taken in each direction, there are k different methods for path generation; each segment is a possible breakpoint between the eye and light portions. That is, by taking m steps from the eye and n steps starting from a light source, we can generate a path of length $m+n+1 = k$. The choice of m and n can have a large effect on

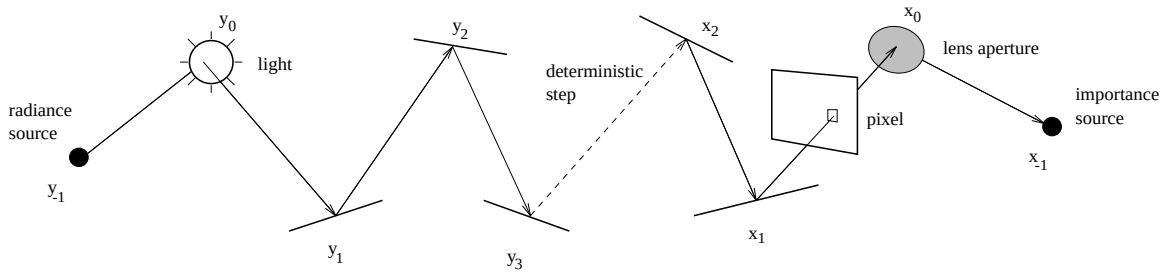


Fig. 2. A complete transport path. For symmetry with the first two light steps, we draw the lens aperture and film plane in the opposite order from a real camera.

the probability distribution of randomly generated paths; for example if we take all steps from the eye, the path distribution does not depend on the light source locations. This fact is crucial in obtaining good MC estimates, since the more closely the path distribution $p(x)$ matches the contributions $f(x)$ made by these paths, the lower the final variance will be (see Sect. 4).

In effect, each of the k partitioning choices leads to a different rendering algorithm for the light flowing on paths of length k. We define a notation for these algorithms: an (m,n)-*method* is one that generates transport paths by taking exactly m eye steps and n light steps (where $m+n+1=k$). Examples are given below.

2.1 Area Lights and Lens Apertures

Note that Fig. 2 refers to two additional points $\mathbf{x}_{\text{-}1}$ and $\mathbf{y}_{\text{-}1}$. These points do not belong to the scene S; they are *artificial* points that allow our algorithms to extend naturally to area light sources and finite-area lens apertures.

To model area light sources, we consider $\mathbf{y}_{\text{-}1}$ to be a point *radiance source* which distributes light energy to the emitting surfaces of the scene. We can think of $\mathbf{y}_{\text{-}1}$ as having a directional distribution on the rays $\mathbf{y}_{\text{-}1} \to \mathbf{y}$ for each point $\mathbf{y}$ of the scene. Since the point $\mathbf{y}_{\text{-}1}$ is entirely artificial, we can *define* the behavior of the light transport kernel K on these rays. In particular we define K so that after one "bounce"[1] the energy emitted along rays $\mathbf{y}_{\text{-}1} \to \mathbf{y}$ is scattered into exactly the desired emitted distribution L_e. This is like applying light transport in reverse; given an arbitrary distribution L_e, we define an artificial kernel that produces L_e after one bounce from a single point light source.

Similarly, we can model the effects of an arbitrary lens system as a scattering function from the external lens surfaces to an artificial *importance source* $\mathbf{x}_{\text{-}1}$. The directional distribution at $\mathbf{x}_{\text{-}1}$ assigns importance to the light arriving at each point on the lens surface.

In effect, this modification gives us two extra places to break the transport paths, since choosing a point on the area light source/lens aperture is considered a "step". It lets us handle problems involving arbitrary emitted light distributions and filter functions with the same methods that we use for a single point light source and a single pinhole lens. It is purely a formalism, in the sense that an implementation must still handle these cases specially. We can include multiple cameras and motion-blur effects with the same technique.

A *complete* transport path is now a sequence

$$\begin{aligned}&\mathbf{y}_{\text{-}1} \to \mathbf{y}_0 \to \cdots \to \mathbf{y}_{n-1} \\ &\rightsquigarrow \mathbf{x}_{m-1} \to \cdots \to \mathbf{x}_0 \to \mathbf{x}_{\text{-}1}\end{aligned}$$

consisting of $k = m+n+1$ segments, where

[1] We define a *bounce* as a single application of the light transport operator determined by K (section 3).

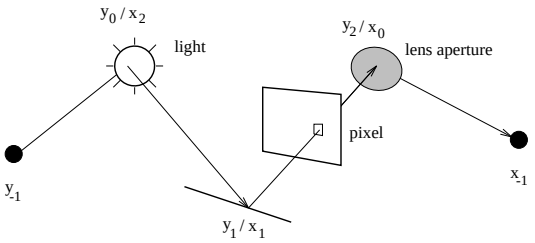


Fig. 3. This transport path leaves the light source, bounces off a surface, and passes through the lens aperture onto the film plane. We can generate paths of this kind in four ways, by taking differing numbers of eye and light steps.

we maintain the convention that an (m, n)-method takes m eye steps and n light steps. The first light step $\mathbf{y}_{-1} \to \mathbf{y}_0$ chooses a random point on a light-emitting surface; the second chooses a direction in which light is emitted. Similarly the first two eye steps (from $\mathbf{x}_{-1}$) choose a point on the lens aperture, and a sampling direction that contributes to the current pixel P.

2.2 Algorithms for One Bounce

Let's qualitatively examine the rendering algorithms we obtain for paths with $k=4$. Such paths account only for light which bounces exactly once on the way from a light source to the eye (the bounces at $\mathbf{x}_0$ and $\mathbf{y}_0$ are artificial). Since there are four possible segments where we could break between eye and light portions, there are four possible rendering algorithms.

A. The (0,3)-method (see Fig. 3). The light steps are: choose a point $\mathbf{y}_0$ on a light source, choose a direction to get an emitted ray $\mathbf{y}_0 \to \mathbf{y}_1$, and follow it through one random bounce to get $\mathbf{y}_1 \to \mathbf{y}_2$. A non-zero contribution occurs only if this ray happens to pass through the lens aperture and strike the film plane near the pixel P. This is exactly what happens with a real camera.

B. The (1,2)-method. The two light steps choose an emitted ray $\mathbf{y}_0 \to \mathbf{y}_1$. The eye step chooses a point $\mathbf{x}_0$ on the lens aperture. To contribute, y_1 must be visible to x_0 in the small range of directions corresponding to pixel P (and of course $y_1 \rightsquigarrow \mathbf{x}_0$ must be unobstructed).

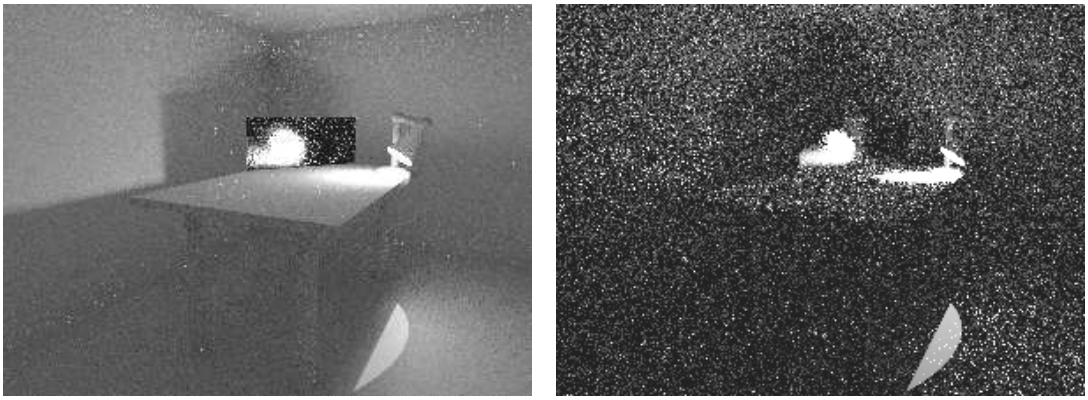
C. The (2,1)-method. The light step chooses a point $\mathbf{y}_0$ on a light source. The eye steps choose an aperture point and direction. To contribute, $\mathbf{x}_1$ and $\mathbf{y}_0$ must be mutually visible. This technique is the one normally associated with MC ray tracing, which follows paths backward from the eye, but computes the direct lighting separately.

D. The (3,0)-method. The eye steps choose a sample ray and follow it through one bounce. To contribute, the path must land on a light source. This technique is naive MC ray tracing with no direct lighting component.

Note that to get a sample contribution with any of these methods, not only must the connecting segment be unobstructed, but also the BRDF's at both ends must reflect some light along it.

Methods A and B seem impractical. However if we allow point light sources and perfect mirrors, it is easy to construct examples where these are the *only* methods (of the four) capable of producing a reasonable result. To see this, note that two or more eye steps result in a sample ray that is guaranteed to miss any point light sources. This deficiency is often seen in Monte Carlo or ray-traced images, where the effects of a point light source are visible but the light itself is not. (Depending on the filter function used over the image plane, a point source should be blurred over several pixels.)

A more practical example comes from the direct lighting calculation, i.e. the difference between methods C and D. It is well-known that if we view an area light source through a perfect mirror, the direct lighting calculation fails. Only a single point on the light source contributes to a transport path ending on the mirror; the probability of randomly choosing this point is zero. It is much better to follow the transport path backward through an additional bounce. More generally, if the surface is *almost* a mirror, the direct lighting "optimization" will give much noisier estimates than the naive method (although both have the correct expected value).



(a) (b)

Fig. 4. (a) An image computed by building transport paths starting from the lights (in this case, a single point source inside the lamp shade). **(b)** The same scene using standard MC path tracing from the eye. See Plates 1 and 2 in the color section for more detailed images.

We emphasize that we are not claiming that these techniques (when $k=4$) are practical in most scenes; they are simply examples of the methods which come naturally from our formulation of bidirectional sampling. For larger k, we *do* claim that the new methods can be superior in practice; the following section presents some evidence of this.

2.3 Two or More Bounces

As an example we have chosen a scene that is very challenging for pure Monte Carlo methods, to emphasize the differences between various techniques. The best images produced by our preliminary implementation are still quite noisy. All images were computed with 50 samples per pixel. The images in the text itself were computed at a resolution 300 by 225; the color section contains two images computed at 900 by 675.

The scene in Fig. 4(a) consists of a table, a desk lamp, and a shiny slab of metal in a closed room. (See Plate 1 in the color section.) All surfaces are diffuse, except for the metal slab which is Phong-specular. All light in the scene comes from a single point light source located in the lamp shade (the "bulb filament"). Because of this, almost all lighting is indirect. Most light is reflected one or more times within the lamp shade, and then it strikes the table top before illuminating the rest of the scene. This image was made with the new techniques described in this paper; it is the union of all $(2,k)$-methods (all steps taken from the lights, except for the choice of initial viewing ray).

A standard MC image made with the same number of samples is substantially more noisy and darker (Fig. 4(b), Plate 2 in the color section). The reason is clear when we examine the surfaces that are lit directly (Fig. 5(a)). Even with the direct lighting optimization, a transport path must randomly strike one of these directly lit regions to make any contribution, and most of the light energy after one bounce is concentrated on the interior of the lamp shade and a small area of the table. In terms of the (m,n) notation defined above, this image is the union all $(k,1)$-methods.

Note that Fig. 4(a) and (b) should have the same average brightness, since both give the correct expected value at each pixel. The reason for the discrepancy is that most white pixels in Fig. 4(b) are actually much brighter than could be displayed, and have been truncated

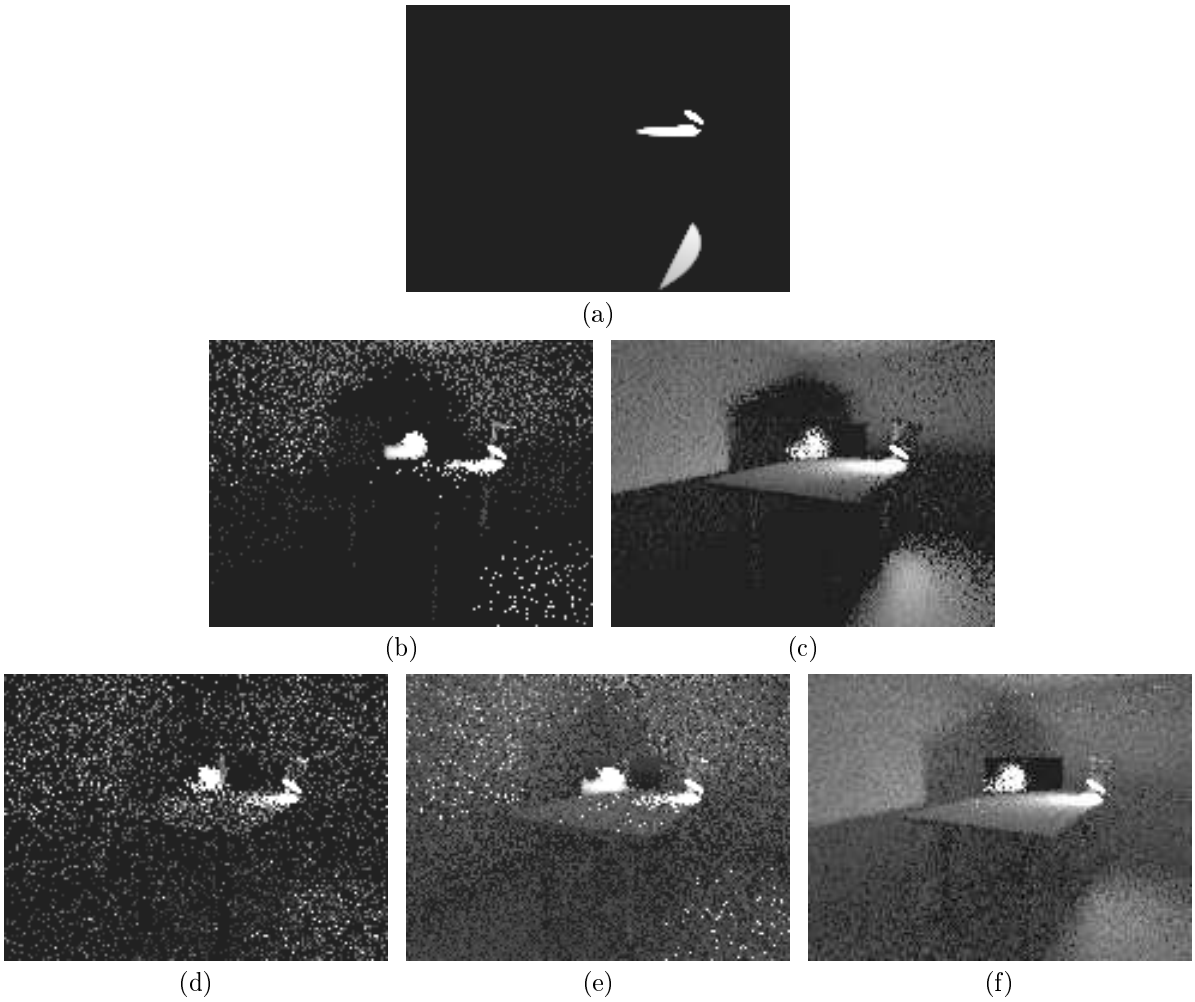


Fig. 5. (a) The direct lighting component (light which bounces exactly once on its way from the light source to the eye). It is rendered with the $(2,1)$-method: two eye steps (to choose a viewing ray) and one light step (choosing the point on the light source). **(b,c)** The two-bounce component. The left image shows the $(3,1)$-method, the right image shows the $(2,2)$-method (i.e. the right image takes an additional light step). **(d,e,f)** Three bounces. From left to right, we have the $(4,1)$, $(3,2)$, and $(2,3)$-methods.

at a maximum value.

Fig. 5 shows various possibilities for rendering the light due to one, two, and three physical bounces (the first 3 components of the steady-state solution). The $(k,1)$-methods correspond to standard Monte Carlo with the direct lighting optimization. Our implementation does not yet support the $(0,k)$- and $(1,k)$-methods, although this should be easy to do. The images have been computed at low resolution (160 by 120) so that individual pixels can be seen. It was necessary to brighten these

images relative to Fig. 4, since each image accounts for only a fraction of the light in the scene.

3 Light Transport in Ray Form

To analyze our algorithms, we have found it useful to reformulate the rendering equation[6] as an integral over *rays*. This leads to a very simple expression for the kernel; the geometric terms are hidden in the *measure function* we use for the inner product. We find that the *ray measure* simplifies the description of bidirectional sampling, and is very useful when dealing with general filter functions and light distributions.

3.1 Local Form

Pat Hanrahan has written an excellent development of the following material which can be found in [22]. Please consult this reference for an explanation of terms not defined here.

Light transport is described by an integral equation of the form:

$$L_{\mathrm{o}}(\mathbf{x},\vec{\omega}_{\mathrm{o}}) = L_{\mathrm{e}}(\mathbf{x},\vec{\omega}_{\mathrm{o}}) + \int_{\Omega_{4\pi}} K_{\mathrm{L}}(\mathbf{x},\vec{\omega}_{\mathrm{o}},\vec{\omega}_{\mathrm{i}}) L_{\mathrm{i}}(\mathbf{x},\vec{\omega}_{\mathrm{i}})\, d\omega_{\mathrm{i}} \quad . \tag{1}$$

We call this the *local* form of light transport equation, because all quantities are expressed in terms of $\mathbf{x}$. It expresses the relation between incoming and outgoing light at a particular point $\mathbf{x}$ on a surface of the scene S.

The function K_{L} is called the *kernel* of the integral equation, and describes how light is scattered. For reflectance functions from surfaces, K_{L} has the form

$$K_{\mathrm{L}}(\mathbf{x},\vec{\omega}_{\mathrm{o}},\vec{\omega}_{\mathrm{i}}) = f_{\mathrm{r}}(\mathbf{x},\vec{\omega}_{\mathrm{o}},\vec{\omega}_{\mathrm{i}}) \cos(\theta_{\mathrm{i}})$$

where f_{r} is the bidirectional reflectance distribution function (BRDF) and θ_{i} measures the angle between $\vec{\omega}_{\mathrm{i}}$ and the surface normal at $\mathbf{x}$. Physically accurate surface reflection models lead to a symmetric BRDF, i.e. $f_{\mathrm{r}}(\mathbf{x},\vec{\omega}_{\mathrm{o}},\vec{\omega}_{\mathrm{i}}) = f_{\mathrm{r}}(\mathbf{x},\vec{\omega}_{\mathrm{i}},\vec{\omega}_{\mathrm{o}})$ due to a physical principle known as *Helmholtz reciprocity*[22].

3.2 Three-Point Form

We can express L_{i} in terms of L_{o} via a change of variables to get the *three-point* form of the rendering equation (note that $\mathbf{x}$ is now $\mathbf{x}'$, and L_{o} is now L):

$$L(\mathbf{x}'\to\mathbf{x}'') = L_{\mathrm{e}}(\mathbf{x}'\to\mathbf{x}'') + \int_S K_3(\mathbf{x}\to\mathbf{x}'\to\mathbf{x}'') L(\mathbf{x}\to\mathbf{x}')\, d\mathbf{x} \quad . \tag{2}$$

The integration is now over the scene S, and the kernel is given by

$$K_3(\mathbf{x}\to\mathbf{x}'\to\mathbf{x}'') = f_{\mathrm{r}}(\mathbf{x}\leftrightarrow\mathbf{x}'\leftrightarrow\mathbf{x}'')\, V(\mathbf{x}\leftrightarrow\mathbf{x}') \cdot \frac{\cos(\theta)\cos(\theta')}{\|\mathbf{x}-\mathbf{x}'\|^2} \quad .$$

3.3 Ray Form

Note that the domain of L is a 4-dimensional space, the space of all *rays*. However, the integration in (2) is taken only over a 2-dimensional subset (those rays $\mathbf{x}\to\mathbf{x}'$ where $\mathbf{x}'$ is fixed). It seems natural to integrate instead over the domain of all rays, which leads to the *ray* form of the rendering equation:

$$L(\vec{\mathbf{x}}) = L_{\mathrm{e}}(\vec{\mathbf{x}}) + \int_R K_{\mathrm{R}}(\vec{\mathbf{x}},\vec{\mathbf{y}}) L(\vec{\mathbf{y}})\, d\mu(\vec{\mathbf{y}}) \tag{3}$$

where $\vec{\mathbf{x}} = \mathbf{x}\to\mathbf{x}'$ and $\vec{\mathbf{y}} = \mathbf{y}\to\mathbf{y}'$ are rays, and R contains all rays with $\mathbf{y},\mathbf{y}' \in S$. Not all rays contribute equally; this is controlled by the measure function

$$d\mu(\mathbf{y}\to\mathbf{y}') = V(\mathbf{y}\leftrightarrow\mathbf{y}') \cdot \frac{\cos(\theta)\cos(\theta')}{\|\mathbf{y}-\mathbf{y}'\|^2}\, d\mathbf{y}\, d\mathbf{y}'$$

where $V(\mathbf{y}\leftrightarrow\mathbf{y}')$ is 1 if $\mathbf{y}$ and $\mathbf{y}'$ are mutually visible and 0 otherwise. The quantity $d\mu(\mathbf{y}\to\mathbf{y}')$ is known as the *throughput* of a differential beam[22]. Finally, the kernel K_{R} describes the fraction of light travelling along $\vec{\mathbf{y}}$ which is scattered along $\vec{\mathbf{x}}$. For scattering to take place, we need a delta function which says that one ray terminates where the next begins:

$$K_{\mathrm{R}}(\mathbf{x}\to\mathbf{x}',\mathbf{y}\to\mathbf{y}') = f_{\mathrm{r}}(\mathbf{y}\leftrightarrow\mathbf{x}\leftrightarrow\mathbf{x}')\delta(\mathbf{y}'-\mathbf{x}) \quad .$$

It is useful to think of the integration as an inner product:

$$\langle f, g \rangle \equiv \int_R f(\vec{\mathbf{x}}) g(\vec{\mathbf{x}})\, d\mu(\vec{\mathbf{x}}) \qquad (4)$$

and to think of K_{R} as defining a *light transport operator*

$$(\mathcal{T}L)(\vec{\mathbf{x}}) = \langle K_{\mathrm{R}}(\vec{\mathbf{x}}, \cdot), L \rangle \ . \qquad (5)$$

$\mathcal{T}$ has an intuitive meaning: it describes the way light bounces (for the given scene S). If L is any light distribution, then $\mathcal{T}L$ is the distribution after exactly once bounce. Using this notation, (3) has a very simple form: $L = L_{\mathrm{e}} + \mathcal{T}L$.

This is the essence of the ray form: a simple form for the kernel K_{R}, and a symmetric, intuitively meaningful inner product over the space of all rays. The framework is more general than the three-point form, since operators described in this way are closed under composition. (For example, the three-point form cannot represent the transport operator $\mathcal{T}^2$.) This lets us think about general linear operators, where the output on a given ray depends linearly on the entire input distribution.

3.4 Filter Functions on Rays

In computer graphics, the goal is to compute intensity values at a discrete set of *pixels*. The value at pixel P is computed by integrating the solution L with a *weighting* or *filter* function W_P. Normally the filter for a given pixel is equivalent to point-sampling a convolution over the image plane.

The inner product over rays provides a simple way to manipulate more general filter functions. Rather than specifying a filter over the image plane, we supply a weighting coefficient $W_P(\mathbf{x} \to \mathbf{x}')$ for each ray. The integration to obtain a pixel value is written as $\langle W_P, L \rangle$, using the inner product over all rays. Note that W_P also models the effects of the imaging system. For example, we can model a simple finite-aperture "lens" by taking a standard pinhole camera and making the hole a little larger. The rays that contribute to W_P all pass through the aperture A and meet the film plane near P (see Fig. 6). For a pinhole aperture, W_P has a component which is a two-dimensional δ-function, reducing the inner product to a 2D integration over the film plane.

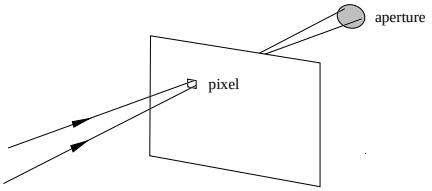


Fig. 6. Rays that contribute to the filter function for a given pixel P. For symmetry with area light sources, we have drawn the lens aperture behind the film plane (they obviously occur in the other order in a real camera!)

3.5 Importance Transport

Adjoint methods for the solution of integral equations have long been used in other fields, such as neutron transport[23, 24, 2]. A continuous adjoint formulation for radiance transport was first proposed in the computer graphics literature by [25], based on earlier work in [26] and [27]. We review this material here for two reasons: first, we believe that the inner product on rays helps to clarify the relationship between the rendering equation and its adjoint. Second, the idea of building paths from the light sources can be viewed as a direct solution method for the adjoint rendering equation. This idea has been applied in neutron transport problems[24], where in fact "direct" and "adjoint" methods have the opposite meaning they are given in computer graphics. Appendix A develops this relationship further. To our knowledge the bidirectional Monte Carlo techniques proposed by this paper have not been explored elsewhere.

Two linear operators $\mathcal{O}$ and $\mathcal{O}^*$ are *adjoint* if $\langle f, \mathcal{O}g \rangle = \langle \mathcal{O}^* f, g \rangle$ for all f and g, where $\langle f, g \rangle$ in an inner product (we use the inner product defined by (4)). The adjoint is not a complex notion; the corresponding idea for matrices of real numbers is the transpose operator.

If $\mathcal{T}$ is light transport operator defined by $(\mathcal{T}L)(\vec{\mathbf{x}}) = \langle K_{\mathrm{R}}(\vec{\mathbf{x}}, \cdot), L \rangle$, then it is easy to verify that its adjoint is defined by

$$(\mathcal{T}^* W)(\vec{\mathbf{x}}) = \langle K_{\mathrm{R}}(\cdot, \vec{\mathbf{x}}), W \rangle \tag{6}$$

where the only difference between $\mathcal{T}$ and $\mathcal{T}^*$ is the order of the arguments to K_{R}. What is the meaning of $\mathcal{T}^*$? Just as $\mathcal{T}$ describes one bounce of a light distribution L, $\mathcal{T}^*$ describes a way to bounce the *filter function* W such that $\langle W, \mathcal{T}L \rangle = \langle \mathcal{T}^* W, L \rangle$. We speak of W as an *importance* distribution when it is propagated by $\mathcal{T}^*$ in this way.

We give a simple proof here of a result from [25], that except for a change in ray orientation, radiance and importance are propagated in the same way. We have

$$\begin{aligned} K_{\mathrm{R}}(\mathbf{x} \to \mathbf{x}', \mathbf{y} \to \mathbf{y}') &= f_{\mathrm{r}}(\mathbf{y} \leftrightarrow \mathbf{x} \leftrightarrow \mathbf{x}') \delta(\mathbf{y}' - \mathbf{x}) \\ &= f_{\mathrm{r}}(\mathbf{x}' \leftrightarrow \mathbf{y}' \leftrightarrow \mathbf{y}) \delta(\mathbf{x} - \mathbf{y}') \\ &= K_{\mathrm{R}}(\mathbf{y}' \to \mathbf{y}, \mathbf{x}' \to \mathbf{x}) \end{aligned}$$

from which we see that (5) and (6) are the same except for the orientation of the rays. This shows that the importance on a given ray is propagated just as light flowing in the opposite direction. This is not to be confused with the notion of a self-adjoint linear operator $\mathcal{T} = \mathcal{T}^*$, since this requires a *symmetric* kernel K_{R}.

4 Partitioning for Variance Reduction

As in Sect. 2, we focus on the problem of estimating the contribution to a pixel P from light that travels on paths of length k. As outlined there, we have k methods for generating the transport paths, which lead to k algorithms for estimating the contribution to P. In the terminology of statistics, we have k different *estimators* for the same quantity. It is natural to ask under what conditions each of these k estimators has the lowest variance, or more generally how to combine them to get the best features of each one. In this section we describe several ideas we are experimenting with.

4.1 What Makes a Good Estimator?

First, let's examine how the estimators are constructed and why they should have different variances. We will need a basic principle of MC integration, namely that if X is a random variable with probability distribution $p(x)$ (with respect to a measure μ), then

$$\begin{aligned} \int_\Omega f(x)\, d\mu(x) &= \int_\Omega \frac{f(x)}{p(x)} p(x)\, d\mu(x) \\ &= E\left[\frac{f(X)}{p(X)}\right] \end{aligned}$$

provided that $f(x)/p(x) < \infty$ for all x. Essentially this says that to estimate an integral $\int f$, we sample a point x chosen from an arbitrary probability distribution $p(x)$, and take $f(x)/p(x)$ as our estimate.

We need to relate this to the estimators for paths of length k. In our case, x is a transport path of length k, and Ω is the space of all such paths. The integral $\int f(x)\, d\mu(x)$ is just a reformulation of the inner product $\langle W_P, \mathcal{T}^k L_{\mathrm{e}} \rangle$ (see Sect. 3.4) as an integral over these paths:

$$\langle W_P, \mathcal{T}^k L_{\mathrm{e}} \rangle = \int_\Omega f(x) d\mu(x) \tag{7}$$

where $f(x)$ is proportional to the light flowing along x, (we call this the *transport coefficient* of the path), and $d\mu_k(x)$ measures the throughput of the path (we omit the details in this discussion). Finally, $p(x)$ is the probability density with which we generate x, which is different for each of the k estimators. Each estimator works by generating a path x, computing $f(x)$ and $p(x)$ for this path, and scoring a contribution $f(x)/p(x)$ (we average several samples from estimator to reduce the variance).

Let's examine why the methods generate transport paths with different probabilities. Each path is built by taking a number of steps, where at each step we randomly choose a local direction $\vec{\omega}$ in which to extend the path. Let $\mathbf{y}$ be the current path endpoint, and let $p(\vec{\omega})$ be the probability distribution we use to extend

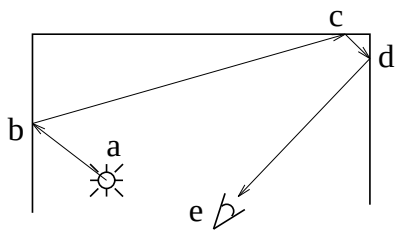


Fig. 7. The choice of breakpoint between light and eye steps has a large effect on the probability of generating this path.

the path. The probability density of extending the path to a point $\mathbf{y}'$ is

$$p(\vec{\omega})\, d\omega = \frac{p(\vec{\omega}) \cos(\theta')}{\|\mathbf{y} - \mathbf{y}'\|^2}\, d\mathbf{y}' \quad . \tag{8}$$

So we see the probability of generating a given path segment depends on the geometry (i.e. the length of the segment and the surface normals at its endpoints), as well as the choice of $p(\vec{\omega})$. Note that the probabilities of generating $\mathbf{y} \to \mathbf{y}'$ and $\mathbf{y}' \to \mathbf{y}$ could be very different, and this is exactly the distinction between using a light step or an eye step to generate this segment. Most important, for each of the k methods there is a path segment that does *not* need to be generated randomly (the connecting segment $\mathbf{y}_n \rightsquigarrow \mathbf{x}_m$).

For example, consider the path of length 4 in Fig. 7. What is the probability of generating this path if we take two light steps and one eye step, vs. one light step and two eye steps? In the first case we must generate **bc** but not **cd**, and in the second we must generate **dc** but not **cb**. Since **dc** is much shorter than **bc** in this example, by (8) the second method is more likely to generate the path **abcde** (other things being equal). It is this effect that causes the noise visible in Plate 1 where two walls meet; there are important transport paths which are generated with very low probability.

Finally, we need to understand the relationship between a probability distribution on paths and the variance of the corresponding estimator. Let F be one of the *original estimators* $f(X)/p(X)$ described above. One way to write the variance of F is

$$\mathrm{Var}[F] = E[F^2] - E[F]^2$$

and since $E[F]$ is the fixed quantity we are trying to estimate, we want to minimize the second-order moment $E[F^2]$. If $E[F^2]$ is large, our image will be noisy. Examining $F = f(X)/p(X)$, it is clearly undesirable to have $p(x)$ small where $f(x)$ is large, since this makes a large contribution to $E[F^2]$. This effect is responsible for the large amounts of noise observed in Plate 2.

4.2 A Discrete Analogy

Here is a simple analogy which demonstrates one idea that we are experimenting with. Suppose that only four transport paths contribute to the pixel we are evaluating (rather than an infinite number), and we have three methods A,B,C of path generation (i.e. these are paths of length three). We show the probability distribution of each method as a bar graph (Fig. 8). The paths are shown as bars with different shadings; each path appears in all three graphs (since each method can generate all the paths). However the bars have different shapes: the *width* of a bar is the probability $p(x)$ of generating path x; the *height* is the sample value $f(x)/p(x)$ (recall $f(x)$ is the transport coefficient for the path). Note that the bar corresponding to a path x has the same area $f(x)$ in all three graphs. This is necessary for the methods to be unbiased.

We want to combine these estimators in a way that remains unbiased (each path x occupies the proper area $f(x)$), but also has a lower variance. First we need to decide what sort of estimator combinations we will allow. A natural way to combine the estimators is a *partitioning*, where three new estimators A', B', C' each estimate the integral over a subset of the paths, and the final estimate has the form $S = A' + B' + C'$. The estimator A' uses the same method for path generation as A; however A' has the flexibility to discard samples when this is desirable, as long as the discarded paths are accounted for by one of B' or C'. We would like to find a way to minimize the variance of S over all such partitionings.

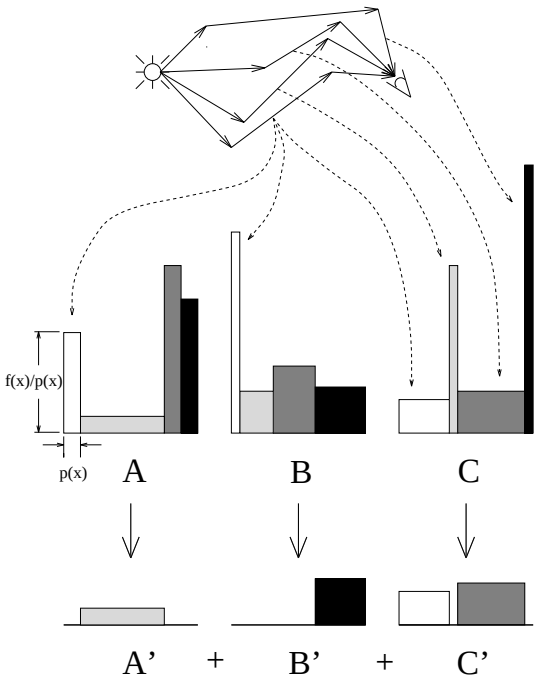


Fig. 8. The upper bar graphs show a discrete probability distributions on four paths for three estimators A, B, C. The lower bar graphs show a partitioning of the paths among new estimators A', B', C' which minimizes the sum of their second-order moments.

4.3 A Partitioning Heuristic

We noted above that minimizing $\mathrm{Var}[S]$ is equivalent to minimizing $E[S^2]$ since $E[S]$ is fixed. This is the second-order moment of a sum $A'+B'+C'$. It turns out to be much easier to minimize the sum of the second-order moments, $E[(A')^2]+E[(B')^2]+E[(C')^2]$. This is not the same as minimizing the variance, but it is similar. To understand what this does, examine Fig. 8. Geometrically, the expected value $E[A']$ is the sum over each path of its rectangle area. Similarly the second-order moment $E[(A')^2]$ is the sum of each rectangle's area *times its height*. Intuitively, minimizing the second-order moment is good because it penalizes tall, thin rectangles; these "spikes" can make a large contribution to the variance.

How do we partition the paths to minimize this sum? Since each path x must be assigned to one of A', B', C', and the area of x is the same in all cases, we want to place x where it has the *greatest width*, i.e. the highest probability of being generated. We call this the *maximum heuristic*, which assigns each path x to the estimator that generates it with highest probability.

```
1  MAXIMUM-HEURISTIC(P)
2    S ← 0
3    for k ← 1 to Max-length
4      for i ← 1 to k
5        x ← CHOOSE-PATH(P, k, i)
6        if p_i(x) ≥ p_j(x)  ∀j ≠ i
7          then  S ← S + f(x)/p_i(x)
8          else  (Discard x and score 0)
9    return S
```

Fig. 9. The *maximum heuristic* for combining estimators. CHOOSE-PATH(P, k, i) chooses a path contributing to pixel P of length k, using segment i as the breakpoint between the eye and light portions. Note that for efficiency, path generation for length k will reuse the eye and light portions from smaller path lengths.

Pixel estimation using the maximum heuristic is outlined in Fig. 9. The basic idea (for $k = 3$) is to take one sample from each of A', B', C' and sum the results. Sampling from an estimator A' is easy: we simply take a sample x from A, then we compute the probabilities $p_A(x), p_B(x), p_C(x)$. If $p_A(x)$ is not the largest of these, we reject x and return zero. Note that once a path x has been chosen, it is easy to compute the probability with which any of the other methods generates it.

We are experimenting with several other heuristics that combine the original estimators in more general ways. For example, we can try to minimize the variance over all *weighted partitionings* of the paths. In our discrete example, this corresponds to splitting the rectangle area $f(x)$ among the new estimators, for each path x. We have some preliminary theoretical results about these heuristics, but have not

yet verified their effectiveness in practice. The goal is to automatically combine the best features of all the original estimators.

5 Acknowledgements

Thanks to Marc Levoy and Brian Curless for reviewing early drafts of this paper. Thanks also to Jorge Stolfi and Stephen Harrison for many stimulating discussions. This work was supported by the National Science Foundation (CCR-9215219), the Digital Systems Research Center, and the Digital External Research Program.

References

1. J. Hammersley, D. Handscomb, *Monte Carlo Methods*, Chapman and Hall, 1964.
2. M. Kalos, P. Whitlock, *Monte Carlo Methods, Volume I: Basics*. J. Wiley, New York, 1986.
3. R. Cook, T. Porter, L. Carpenter, Distributed ray tracing. *Computer Graphics (SIGGRAPH '84)*, **18**, 137–146 (1984).
4. J. Arvo, D. Kirk, Particle transport and image synthesis, *Computer Graphics (SIGGRAPH '90)*, **24**, 63–66 (1990).
5. P. Shirley, C. Wang, Luminaire sampling in distribution ray tracing. Technical Report 343, CS Dept., Indiana University, Jan 1992. Also appears in: *SIGGRAPH '93* Global Illumination Course Notes.
6. J. Kajiya, The rendering equation. *Computer Graphics (SIGGRAPH '86)*, **20**, 143–150 (1986).
7. D. Kirk, J. Arvo, Unbiased sampling techniques for image synthesis, *Computer Graphics (SIGGRAPH '91)*, **25**, 153–156 (1991).
8. F. Sillion, J. Arvo, S. Westin, D. Greenberg, A global illumination solution for general reflectance distributions. *Computer Graphics (SIGGRAPH '91)*, **25**, 187–196 (1991).
9. D. Baum, S. Mann, K. Smith, J. Winget, Making radiosity usable: automatic preprocessing and meshing techniques for the generation of accurate radiosity solutions. *Computer Graphics (SIGGRAPH '91)*, **25**, 51–60 (1991).
10. D. Lischinski, F. Tampieri, D. Greenberg, Combining hierarchical radiosity and discontinuity meshing. *Computer Graphics (SIGGRAPH '93)*, **27**, 199–208 (1993).
11. S. Pattanaik, S. Mudur, Efficient potential equation solutions for global illumination computation. *Computers and Graphics*, **17** (4), 387–396 (1993).
12. S. Chen, H. Rushmeier, G. Miller, D. Turner, A progressive multi-pass method for global illumination. *Computer Graphics (SIGGRAPH '91)*, **25**, 165–174 (1991).
13. M. Watt, Light-water interaction using backward beam tracing. *Computer Graphics (SIGGRAPH '90)*, **24**, 377–385 (1990).
14. P. Shirley, A ray tracing method for illumination calculation in diffuse-specular scenes. *Graphics Interface '90*, 205–212 (1990).
15. J. Arvo, Backward ray tracing. *SIGGRAPH '86* "Developments in Ray Tracing" course notes (1986).
16. P. Heckbert, Adaptive radiosity textures for bidirectional ray tracing. *Computer Graphics (SIGGRAPH '90)*, **24**, 145–154 (1990).
17. P. Shirley, Time complexity of Monte-Carlo radiosity. *Eurographics '91 Proceedings*, 459–465 (1991).
18. P. Shirley, K. Sung, W. Brown, A ray tracing framework for global illumination. *Graphics Interface '91*, 117–128 (1991).
19. B. Le Saec, C. Schlick, A progressive ray-tracing based radiosity with general reflectance functions. *Eurographics Workshop on Photosimulation, Realism, and Physics in Computer Graphics*, 1990.
20. G. Ward, F. Rubinstein, R. Clear, A ray tracing solution for diffuse interreflection. *Computer Graphics (SIGGRAPH '88)*, **22**, 85–92 (1988).
21. E. Lafortune, Y. Willems, Bidirectional path tracing. *CompuGraphics Proceedings* (Alvor, Portugal), 145–153 (Dec. 1993).
22. M. Cohen, J. Wallace, *Radiosity and Realistic Image Synthesis*. Academic Press, 1993.
23. Lewins, Jeffery, *Importance, The Adjoint Function: The Physical Basis of Variational and Perturbation Theory in Transport and Diffusion Problems*. Pergamon Press, New York, 1965.
24. J. Spanier, E. Gelbard, *Monte Carlo Principles and Neutron Transport Problems*, Addison-Wesley, 1969.
25. P. Christensen, D. Salesin, T. DeRose, A continuous adjoint formulation for radiance

transport. *Fourth Eurographics Workshop on Rendering*, 1993.
26. S. Pattanaik, S. Mudur, The Potential Equation and Importance in Illumination Computations. *Computer Graphics Forum*, **12** (2), 131–136 (1993).
27. B. Smits, J. Arvo, D. Salesin, An importance-driven radiosity algorithm. *Computer Graphics (SIGGRAPH '92)*, **26**, 273–282 (1992).

A Recursive Formulation

In this appendix we develop further the relation between bidirectional sampling and light/importance transport. The key is to show how taking a light or eye step reduces the estimation of $\langle W_P, L\rangle$ to a problem of the same form.

Recall that $\mathcal{T}$ denotes the light transport operator (5). As long as $\|\mathcal{T}\| < 1$ (satisfied by all physically valid models), the formal solution to $L = L_\mathrm{e} + \mathcal{T}L$ is given by the *Neumann series*[2],

$$L = L_\mathrm{e} + \mathcal{T}L_\mathrm{e} + \mathcal{T}^2 L_\mathrm{e} + \cdots \quad .$$

The term $\mathcal{T}^i L_\mathrm{e}$ in the expansion represents the contribution of light which bounces exactly i times. Note that $\|\mathcal{T}^i L_\mathrm{e}\|$ necessarily decreases as i grows. For convenience we define the *solution operator*

$$\mathcal{S} \equiv (\mathcal{I} - \mathcal{T})^{-1} = \mathcal{I} + \mathcal{T} + \mathcal{T}^2 + \cdots \qquad (9)$$

(where $\mathcal{I}$ is the identity operator), so that the solution to the light transport equation is now just $L = \mathcal{S}L_\mathrm{e}$, and our goal is to estimate $\langle W_P, \mathcal{S}L_\mathrm{e}\rangle$.

A.1 Notation for Eye and Light Steps

We assume that W_P represents a pinhole lens with the point aperture located at $\mathbf{x}_0$, and that the emitted light L_e is due to a single point light source at $\mathbf{y}_0$. It will be convenient to define W_P and L_e as "slices" of the kernel K_R, i.e. as functions $K_\mathrm{R}(\vec{\mathbf{x}}, \vec{\mathbf{y}})$ where one of $\vec{\mathbf{x}}$ or $\vec{\mathbf{y}}$ is held fixed. In particular, we define $K_\mathrm{R}(\mathbf{x}_0 \to \mathbf{x}_{\text{-}1}, \mathbf{x} \to \mathbf{x}') = W_P(\mathbf{x} \to \mathbf{x}')$ and $K_\mathrm{R}(\mathbf{y} \to \mathbf{y}', \mathbf{y}_{\text{-}1} \to \mathbf{y}_0) = L_\mathrm{e}(\mathbf{y} \to \mathbf{y}')$. We can think of the emitted light distribution L_e as the result of scattering energy flowing along a single (artificial) ray $\mathbf{y}_{\text{-}1} \to \mathbf{y}_0$, and similarly for W_P. We can still handle arbitrary light sources and filter functions by using the artificial point sources $\mathbf{x}_{\text{-}1}, \mathbf{y}_{\text{-}1}$ as described earlier. However to describe these emitted distributions as slices of the kernel, we must take this one step further: the initial light is concentrated on a single ray $\mathbf{y}_{\text{-}2} \to \mathbf{y}_{\text{-}1}$ which is scattered into the distribution at $\mathbf{y}_{\text{-}1}$ by the first bounce.

For each eye step our algorithm will define an associated weighting function W_i, where $W_0 = W_P$. Similarly each light step introduces a new emitted radiance function L_j where $L_0 = L_\mathrm{e}$. The algorithm works by estimating a sequence of inner products $\langle W_i, \mathcal{S}L_j\rangle$, starting with the original problem $\langle W_P, \mathcal{S}L_\mathrm{e}\rangle$. The sequence of functions W_i and L_i will always have the following form ($i \geq 0$):

$$\begin{aligned} W_i(\vec{\mathbf{x}}) &= K_\mathrm{R}(\vec{\mathbf{x}}_i, \vec{\mathbf{x}}) \\ L_i(\vec{\mathbf{y}}) &= K_\mathrm{R}(\vec{\mathbf{y}}, \vec{\mathbf{y}}_i) \end{aligned} \qquad (10)$$

where $\vec{\mathbf{y}}_i \equiv \mathbf{y}_{i-1} \to \mathbf{y}_i$ and $\vec{\mathbf{x}}_i \equiv \mathbf{x}_i \to \mathbf{x}_{i-1}$ (Fig. 2). Thus each W_i is a "slice" of the kernel K_R that weights incoming rays $\mathbf{x} \to \mathbf{x}_i$ according to how much light they reflect along the outgoing ray $\mathbf{x}_i \to \mathbf{x}_{i-1}$, and similarly for L_i. Property (10) is an important invariant maintained by the following algorithm.

A.2 The Eye Step

To estimate $\langle W_i, \mathcal{S}L_j\rangle$, we first apply the identity $\mathcal{S} = \mathcal{I} + \mathcal{T}\mathcal{S}$ (see (9)):

$$\langle W_i, \mathcal{S}L_j\rangle = \langle W_i, L_j\rangle + \langle W_i, \mathcal{T}\mathcal{S}L_j\rangle \quad .$$

We now need to estimate each of the two terms. By the assumption of (10), $\langle W_i, L_j\rangle$ can be evaluated exactly. In fact W_i and L_j can interact along only a single ray (Fig. 10):

$$\begin{aligned} &\langle W_i, L_j\rangle \\ &= \int_R K_\mathrm{R}(\vec{\mathbf{x}}_i, \vec{\mathbf{x}}) K_\mathrm{R}(\vec{\mathbf{x}}, \vec{\mathbf{y}}_j)\, d\mu(\vec{\mathbf{x}}) \\ &= f_\mathrm{r}(\mathbf{y}_j \leftrightarrow \mathbf{x}_i \leftrightarrow \mathbf{x}_{i-1}) f_\mathrm{r}(\mathbf{y}_{j-1} \leftrightarrow \mathbf{y}_j \leftrightarrow \mathbf{x}_i) \\ &\quad \cdot V(\mathbf{y}_j \leftrightarrow \mathbf{x}_i) \frac{\cos(\theta)\cos(\theta')}{\|\mathbf{y}_j - \mathbf{x}_i\|^2} \end{aligned} \qquad (11)$$

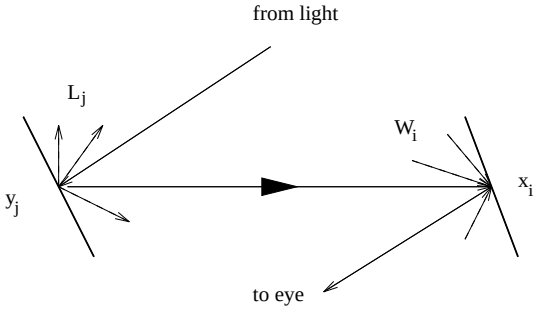


Fig. 10. The inner product $\langle W_i, L_j \rangle$

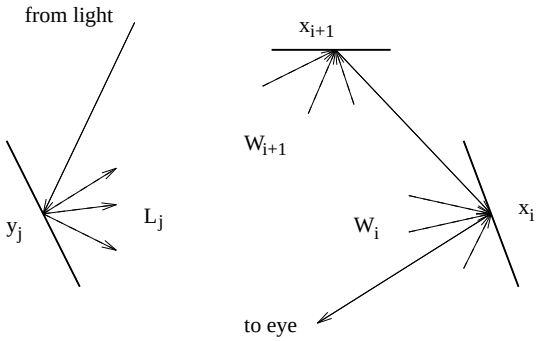


Fig. 11. The eye step replaces W_i by a new emitted importance function W_{i+1}

where θ and θ' are the angles between the ray $\mathbf{y}_j \to \mathbf{x}_i$ and the surface normals at $\mathbf{y}_j$ and $\mathbf{x}_i$ respectively. All terms in (11) can easily be evaluated.

To evaluate the second term $\langle W_i, \mathcal{T}\mathcal{S}L_j \rangle$, we use Monte Carlo sampling. Since W_i is zero except on rays of the form $\mathbf{x} \to \mathbf{x}_i$, we choose a probability distribution p_i which gives positive weight to rays of this form. (In practice, p_i is a distribution on the set of directions out of $\mathbf{x}_i$, i.e. the possible directions to extend our transport path). As long as $W_i(\vec{\mathbf{x}})/p(\vec{\mathbf{x}}) < \infty$ for all $\vec{\mathbf{x}}$, we have

$$\langle W_i, \mathcal{T}\mathcal{S}L_j \rangle = E\left[\frac{W_i(\vec{\mathbf{x}}_{i+1})(\mathcal{T}\mathcal{S}L_j)(\vec{\mathbf{x}}_{i+1})}{p_i(\vec{\mathbf{x}}_{i+1})}\right] \tag{12}$$

where $\vec{\mathbf{x}}_{i+1}$ is randomly distributed according to p_i. The new ray is $\mathbf{x}_{i+1} \to \mathbf{x}_i$ where $\mathbf{x}_{i+1}$ is first surface point intersected in the randomly chosen direction (which we are following backwards relative to the direction that light travels).

All that remains is to evaluate the parenthesized expression in (12), which requires that we estimate $(\mathcal{T}\mathcal{S}L_j)(\vec{\mathbf{x}}_{i+1})$. This is simply the problem we started with, in disguise:

$$\begin{aligned}(\mathcal{T}\mathcal{S}L_j)(\vec{\mathbf{x}}_{i+1}) &= \langle K_{\mathrm{R}}(\vec{\mathbf{x}}_{i+1}, \cdot), \mathcal{S}L_j \rangle \\ &\equiv \langle W_{i+1}, \mathcal{S}L_j \rangle \ .\end{aligned}$$

We have a new weighting function W_{i+1}, whose form is the same "slice" of K_{R} (10) that we assumed for W_i. The eye step is illustrated in Fig. 11.

We can apply this operation as many times as we like, building a transport path extending backward from the eye. We are actually building a family of transport paths, since each prefix of the path contributes to the sample value. At each step, the term $\langle W_i, L_j \rangle$ *connects* the current prefix and suffix to build a complete transport path from the light to the eye.

A.3 The Light Step

To take a light step, we use the adjoint transport operator (see Sect. 3.5). Rather than estimating $\langle W_P, \mathcal{S}L_{\mathrm{e}} \rangle$, we estimate $\langle \mathcal{S}^* W_P, L_{\mathrm{e}} \rangle$, where $\mathcal{S}^* = [(\mathcal{I} - \mathcal{T})^{-1}]^* = (\mathcal{I} - \mathcal{T}^*)^{-1}$. To understand $\mathcal{S}^*$, consider the importance transport equation $W = W_P + \mathcal{T}^* W$. Its solution is the steady-state importance distribution $W \equiv \mathcal{S}^* W_P$. The value $W(\vec{\mathbf{x}})$ is proportional to the contribution that radiance emitted along $\vec{\mathbf{x}}$ eventually makes to the final solution $\langle W_P, \mathcal{S}L_{\mathrm{e}} \rangle$. Thus we have two choices: we can either solve for the steady-state radiance L and compute $\langle W_P, L \rangle$, or solve for the steady-state importance W and compute $\langle W, L_{\mathrm{e}} \rangle$. In fact we have more choices, since we can choose whether to propagate light or importance at each step of building a transport path. It is this observation that leads to k estimators for paths of length k.

We are now ready to describe a *light step*. Again we want to estimate $\langle W_i, \mathcal{S}L_j \rangle$, but this

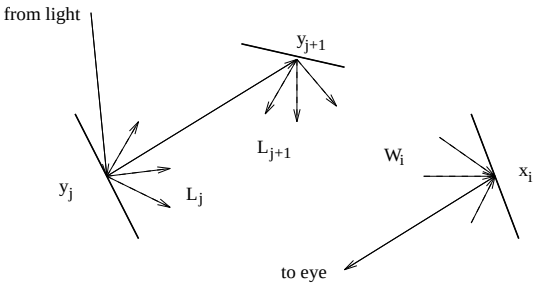


Fig. 12. The light step replaces L_j by a new emitted radiance function L_{j+1}

time we apply the identity $\mathcal{S}^* = \mathcal{I} + \mathcal{T}^*\mathcal{S}^*$:

$$\langle \mathcal{S}^* W_i, L_j \rangle = \langle W_i, L_j \rangle + \langle \mathcal{T}^*\mathcal{S}^* W_i, L_j \rangle \ .$$

The term $\langle W_i, L_j \rangle$ is evaluated exactly as before. For the other term, we have

$$\langle \mathcal{T}^*\mathcal{S}^* W_i, L_j \rangle = E\left[\frac{(\mathcal{T}^*\mathcal{S}^* W_i)(\vec{\mathbf{y}}_{j+1}) L_j(\vec{\mathbf{y}}_{j+1})}{q_j(\vec{\mathbf{y}}_{j+1})}\right] \tag{13}$$

where $\vec{\mathbf{y}}_{j+1}$ is a ray chosen randomly according to q_j (a sampling distribution for the rays contributing to L_j). Finally we need to estimate

$$\begin{aligned}(\mathcal{T}^*\mathcal{S}^* W_i)(\vec{\mathbf{y}}_{j+1}) &= \langle \mathcal{S}^* W_i, K_{\mathrm{R}}(\cdot, \vec{\mathbf{y}}_{j+1}) \rangle \\ &\equiv \langle \mathcal{S}^* W_i, L_{j+1} \rangle\end{aligned}$$

which has the same form we started with, except for the new emitted radiance distribution L_{j+1}. The light step is illustrated in Fig. 12.

Since both eye and light steps leave us with an estimation problem of the same form, we can apply some of each. A complete transport path consists of a prefix $\mathbf{y}_0 \to \cdots \to \mathbf{y}_n$ built by taking light steps, a suffix $\mathbf{x}_m \to \cdots \to \mathbf{x}_0$ built by taking eye steps, and a segment $\mathbf{y}_n \to \mathbf{x}_m$ that deterministically connects them.