Backward Ray Tracing

James Arvo
Apollo Computer, Inc.
Chelmsford, MA

August, 1986*

1 Introduction

Ray tracing has become a very popular method for image synthesis due to its unparalleled flexibility and its ability the generate images of high quality and realism. Simulation of effects such as reflection and refraction have been the hallmarks of ray tracing since its introduction [5]. With the advent of stochastic ray tracing [2], the range of effects expanded to include motion blur, soft shadows, depth of field, and both blurry reflections and translucency. A notable omission from the repertoire of ray tracing, however, is diffuse reflection of indirect light. Though several new algorithms have been introduced which simulate multiple diffuse reflections of light in polygonal environments [1, 4], ray tracing continues to neglect these higher order effects, and consequently falls short of a truly global model of illumination.

These notes describe a simple extension to ray tracing which takes a first step toward alleviating this deficiency. The problem addressed is that of simulating diffuse reflection of specularly reflected and/or refracted light originating from point light sources. The technique involves one or more passes of backward ray tracing (from the light) and the construction of illumination maps as a pre-processing step.

2 Background

In traditional ray tracing, rays are cast from the eye into the environment in order to perform visible surface calculations. Additional rays are spawned at the points of intersection for the purpose of shading. This normally requires at least one ray to

*In *Developments in Ray Tracing*, SIGGRAPH '86 Course Notes, Volume 12, August, 1986. Retyped from original manuscript, January, 1995.

be cast from the point of intersection toward each light source to determine which of them contribute to the illumination of the surface at that point. If the ray encounters an intervening object the point is either declared to be in shadow, (with respect to that light source) or, if the obstruction is translucent, the contribution of the light source is attenuated to simulate absorption.

A problem arises when we attempt to take reflection and refraction into account as rays are traced from the point of intersection back to the light source(s). Were we to reflect rays from or refract rays through obstructing objects as with rays cast from the eye, we would find that virtually none of them stay on course to a light sources and therefore reveal nothing about the amount of light reaching the point in question.

The problem is a difficult one since light deflected by reflection or refraction can come from many unpredictable directions (figure 1). The positions of the light sources offer not the faintest clue where to look except in the case of direct (unobstructed) illumination. One possible approach to locating sources of indirect light would be to carefully sample the entire environment from each point being shaded in an attempt to discover those paths which divert light back toward a light source. This would be enormously expensive and an alternative is clearly necessary.

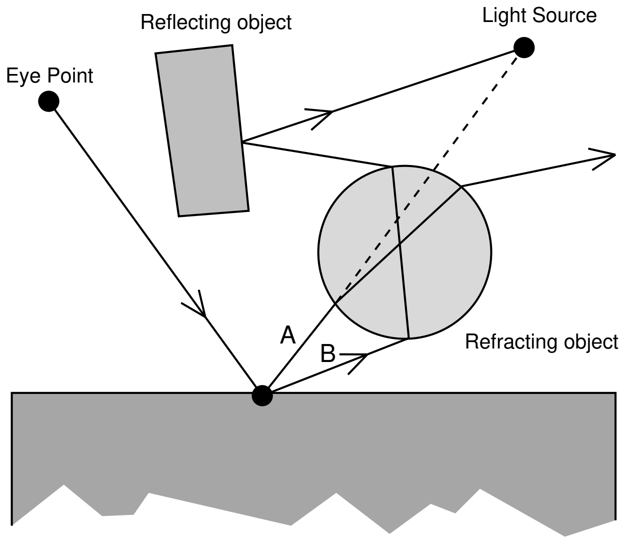

Figure 1: *Ray A directed toward the light source is deflected away. Ray B finds its way to the light source via a complicated path. What proportion of the rays sampling the environment from P make it back to the light?*

Since we need only consider paths which ultimately reach the light source (during the shading calculation), an obvious plan of attack is to reverse the process and cast rays from the light source into the environment. It is this process of ray tracing from the light source which we refer to as *backward* ray tracing since it is counter to the familiar scenario in which all rays begin at the eye. This is the concept at the heart of the algorithm presented here. The role of backward ray tracing is confined to shading calculations, however, and traditional *forward* ray tracing is still employed to determine visible surfaces, simulate mirror reflections, etc. In principle the pre-processing step described in the next section could completely obviate the need for shading rays which sample the light source (for purely diffuse reflectors), but the algorithm to date has been used only as a supplement to a standard shader.

3 Illumination as a Pre-Processing Step

The idea behind the algorithm is quite simple. As a pre-processing step, shower the environment with rays emanating from the light source(s), and assign to each ray some amount of "energy". As each ray recursively wends its way through multiple reflections and refractions, a fraction of its energy is deposited at each surface which possesses a component of diffuse reflection. (We can neglect first-generation rays which hit purely diffuse surfaces.) Finally, render the scene using forward ray tracing and a shader which takes the local energy density into account by adding it into the diffuse component. Generally speaking, areas in which many illumination rays fall are illuminated more brightly by indirect light than areas in which few fall. More precisely, the intensity of the indirect illumination at any point is the energy per unit area, and it is this which we seek to approximate by considering the energy in a small neighborhood surrounding each point being shaded.

The algorithm therefore requires multiple passes of ray tracing; one pass from each light source and a final pass from the eye point. The missing ingredient in what has been discussed thus far is how to communicate the resulting distribution of deposited energy computed during the illumination (backward) phase to the shader used during the rendering (forward) phase; i.e. how the forward and backward rays "meet in the middle."

4 Illumination Maps

The key to recording and retrieving the information generated during the illumination phase lies in the construction of an *illumination map* for each surface in the scene which is illuminated by specularly reflected and/or refracted light. An illumination map is very similar to a texture map except that it carries "brightness" information instead of displacement or normal vector perturbation information.

An illumination map consists of a rectangular array of data points (scalar values for pure white light, or RGB triples for handing colored light) imposed on a 1x1 square, which, in turn, is in one-to-one correspondence with a surface of an object via a parametrization function $T(u, v) \to (x, y, z)$. The inverse of this mapping provides the means of journaling information about where illumination rays hit the surface and how much energy they deposit. Before the illumination phase begins, all data points of all illumination maps are initialized to zero. When a ray hits a surface with an associated illumination map (which need only be true if the surface is a diffuse reflector) during the illumination pass, the u-v coordinates of the point of intersection are computed and a portion of the ray's energy is contributed to the illumination map at that point. This is done by bilinearly partitioning the contribution among the four surrounding data points in the illumination map and incrementing their current values (figure 2). At the completion of the illumination phase, every data point of an illumination map contains a record of nearby hits; close hits weighing more heavily than far away hits. (A direct hit on a data point results in all the energy being contributed to that point alone.)

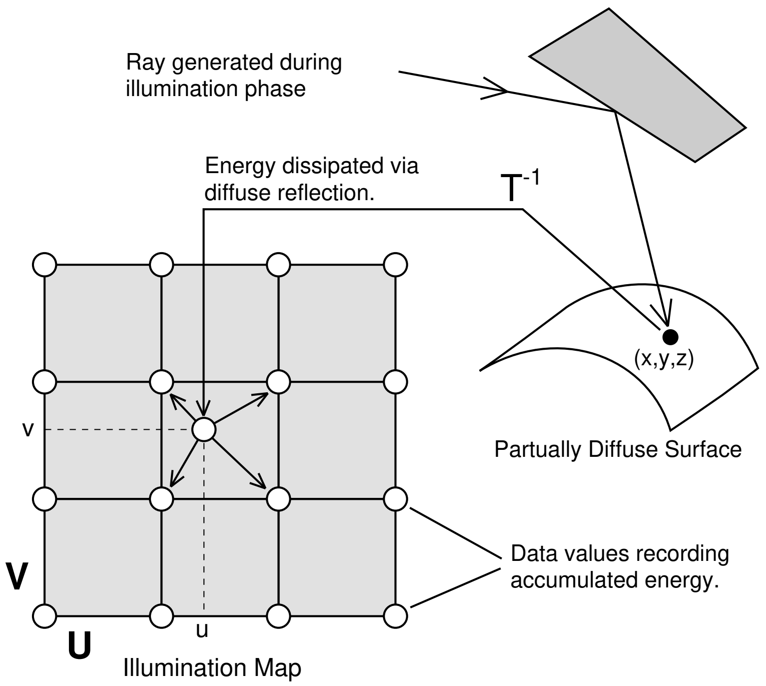

Figure 2: *When an illumination ray hits a partially diffuse surface, some of its energy is split among the four surrounding data points in the illumination map.*

At this stage, the illumination map contains only energy information. To determine intensity we must divide the energy value at each data point by the are it represents in world coordinates. This area can be approximated using the partial derivatives of the surface parametrization function, $T(u, v)$. Hence, the intensity associated with a data point in the illumination map is given by

$$\text{Intensity}(u, v) = \frac{E(u, v)}{\left\| \frac{\partial}{\partial u} T(u, v) \times \frac{\partial}{\partial v} T(u, v) \right\|} \tag{1}$$

where $(u, v) \in [0, 1] \times [0, 1]$ corresponds to one of the data points, and $E(u, v)$ is its accumulated energy after the illumination phase. After dividing each data point by its corresponding area, the illumination map represents intensity and is ready for use by the shader. During the rendering phase, the intensity of indirect light falling on surface points being shaded can be determined by bilinearly interpolating into the associated illumination map. The value obtained in this way provides information about global illumination which is normally unavailable through ray tracing, yet the shader can treat it as though it were just another source of diffuse light.

5 Chromatic Aberration

The position of the focal point of a lens changes slightly with the wave-length of the light passing through it. This causes an effect (usually undesirable for real lenses) called chromatic aberration, which is slight diffusion, or color separation, of the refracted light. A simple method of approximating this phenomenon is to compute red, green, and blue values in the illumination maps in separate illumination phases. In each of the three passes, the refractive index of materials in the environment is altered slightly. The refractive index during the "blue" pass should be the highest, and the "red" pass the lowest.

6 Pitfalls

This method can produce good results if one is careful to sidestep several lurking pitfalls.

- Illumination rays must be many times as dense as the grid points in the illumination map in areas of interest. This produces a more even result statistically.
- Discard low level "noise" in the form of isolated samples in the illumination map. Some of the illumination rays can be scattered haphazardly due to multiple internal reflections, drastic divergence, etc. Since these sparse samples are a

result of spreading the light energy thinly over a large area, the contribution is small and the samples can be safely ignored.

- Since the focusing of a lens can create extremely bright areas, the limited dynamic range of display devices quickly becomes a problem. This can be dealt with quite effectively by gradually desaturating, thereby rendering the brightest points as white.

7 Future Extensions

- Two of the pitfalls described above stem from the fact that this method is based on point sampling in which infinitely thin rays contribute "quanta" of energy. In contrast, Heckbert and Hanrahan [3] have suggested a rather elegant technique for approximating diffuse reflection of specularly reflected and refracted light in polygonal environments which uses "light beam tracing." A similar technique could probably be employed to trace "shafts" of light (polygonal cross-section defined by infinitesimal rays) in the setting of a more general ray tracer. This could produce great improvements in quality and performance over the point sampling method.

- The use of illumination maps in this algorithm is quite similar to the use of form factors in the radiosity algorithm [1]. In fact, illumination maps could provide a means of combining the radiosity algorithm with ray tracing. The resulting hybrid could approach a truly global model of illumination much more closely than either independently.

- Finally, the simulation of chromatic aberration immediately suggests yet another dimension to be sampled stochastically in distributed ray tracing. That is, the illumination rays could be jittered in "frequency", affecting both the RGB energy distribution of the ray and the amount by which to perturb the refractive indices of materials encountered by the ray.

References

[1] Cohen, M. F., and Greenberg, D. P. The hemi-cube: A radiosity solution for complex environments. *Computer Graphics 19*, 3 (July 1985), 75–84.

[2] Cook, R. L. Distributed ray tracing. *Computer Graphics 18*, 3 (1984), 137–145.

[3] Heckbert, P. S., and Hanrahan, P. Beam tracing polygonal objects. *Computer Graphics 18*, 3 (July 1984), 119–127.

[4] Nishita, T., and Nakamae, E. Continuous tone representation of 3-D objects taking account of shadows and interreflection. *Computer Graphics 19*, 3 (July 1985), 23–30.

[5] Whitted, T. An improved illumination model for shaded display. *Communications of the ACM 32*, 6 (June 1980), 343–349.

Addendum, 1995

As the paper appeared in the course notes in 1986, it did not include any sample images generated with the algorithm. A number of examples were shown at the presentation, however, including animations, and the image below:

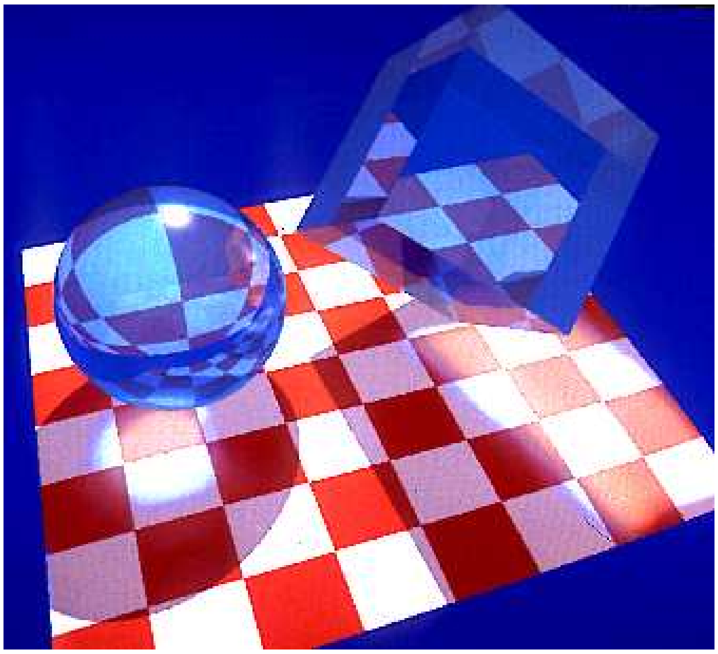

Figure 3: *Two glass objects acting like lenses, focusing light onto the surface beneath them.*

The author regrets introducing the term *backward* to mean “emanating from the light source.” Bad terminology can be worse than no terminology, and this title is an example. The word “backward” has understandably caused a lot of confusion over the years since, in the context of ray tracing, the term is frequently used to mean the opposite of what was intended in this paper! Terms such as *light ray* tracing and *eye ray* tracing are to be preferred, as they are unambiguous.