

# The Light Volume: an aid to rendering complex environments

Ken Chiu (a) Kurt Zimmerman (a) Peter Shirley (a,b)

(a) Indiana University, (b) Cornell University

## 1 Introduction

The appearance of an object depends on both its shape and how it interacts with light. Alter either of these and its appearance will change. Neglect either of these and realism will be compromised. Computer graphics has generated images ranging from nightmarish worlds of plastic, steel, and glass to gently-lit, perfect interiors that have obviously never been inhabited. The reassuring realism lacking in these extremes requires the simulation of both complex geometry and complex light transport.

By complex geometry, we mean at least millions of primitives, possibly procedural. By complex light transport we mean the full range from ideal specular to ideal diffuse.

In this paper we discuss how several algorithms attempt to approximate the light incident at a point (field-radiance) and to what degree they deal with this dual complexity. We also discuss an algorithm designed specifically for this dual complexity by decoupling the interaction between geometric complexity and transport complexity by modeling the light transport in space, rather than on surfaces. We do this through the use of a view independent statistical 5-dimensional approximation based on a particle tracing preprocess that we call the *light volume*.

## 2 Background

The fundamental obstacle to rendering complex scenes is determining what light hits a given point. This is difficult because an arbitrarily small region can generate an arbitrarily complex transport path that contributes an arbitrarily large amount of light. Only by analyzing all possible photon paths can we provide any kind of error tolerance whatsoever. Clearly this is impossible, and any viable algorithm must make approximations.

Although these approximations may take many forms, ultimately they express themselves as approximations to the field radiance. It is instructive to

investigate these approximations in the context of the rendering equation.

$$L = \int \rho L_f \tag{1}$$

where  $\rho$  is the BRDF and  $L_f$  is the field radiance. This form uses the terminology and cosine-weighted solid angle measure advocated by Arvo et al. [1]. We sometimes forget how simple our task is to state. We know  $\rho$ , so all we need to do is a quadrature for an unknown  $L_f$ . Obviously, in practice, this is not easy, but if we had an approximation  $\hat{L}$  to  $L_f$  our job would become easy.

An approximation  $\hat{L}$  to  $L_f$  can be used in a number of different ways. Since  $\hat{L}$  is an estimate of  $L_f$  it can be substituted for  $L_f$  to generate an approximate solution to the rendering equation. Or it could be used for importance sampling as is done by Lafortune [10], where a density function proportional to  $\rho L_f$  is used for choosing scattered directions.

A somewhat surprising fact is that if  $\hat{L}$  is used directly, it does not have to be very accurate unless  $\rho$  has sharp peaks. This is because  $L$  is really just weighted average of  $L_f$  scaled by energy absorption. For Lambertian surfaces,  $L$  is an unweighted average with respect to the cosine-weighted solid angle measure. This is the fact behind many simplifications in rendering algorithms.

### 3 Non-diffuse rendering

A number of authors have developed algorithms for rendering scenes with general reflectance properties (glossy/specular/diffuse). The general families of methods are reviewed in this section.

Kajiya [7] introduced path tracing to solve the rendering equation. This is a complete solution, but can have high variance for “reasonable” runtimes.

Kok [9] and Rushmeier et al. [11] simplified the geometry of illuminating surfaces and does a radiosity preprocess to make path tracing simpler. This truncates the ray tree and eliminates high frequencies in the approximate field-radiance. It is not clear how specular transport is handled in this context.

Lafortune [10] attempts to improve path tracing by storing the radiances as they are computed. Previously computed radiances are then used to guide the current path tracing in an ongoing process. His method works well provided there are no small bright sources that have important transport involving mirrors.

Ward [18] has a very interesting optimization in his Radiance software. When  $\rho$  is sufficiently diffuse, he approximates  $L_f$  with a Lambertian world. This is a good approximation unless “virtual” luminaires are present, in which case Radiance uses a special mechanism that assumes the specular surfaces involved are planar. In addition, he reuses computed values of  $\hat{L}$ . Ward caches previously computed irradiance values in a spatial data structure. He then uses these cached values to terminate path tracing early if a cached value is “close enough”. Essentially, he is approximating the field radiance with previously computed values. This method is well-suited to scenes of medium complexity, but breaks down to

path-tracing with high-complexity scenes because there is not enough surface-normal correlation between adjacent samples.

Jensen [6] generates a photon map during a particle trace preprocess. This photon map can directly give  $\hat{L}$  by using a statistical combination of nearby photon hits. This is essentially doing density-estimation on statistics related to incident lighting. The potential weakness of this technique is that enough surface-particle interactions must be collected to ensure sufficient statistics, which could require a large amount of storage.

A number of finite element methods have been developed to handle non-diffuse scenes, beginning with the work of Immel et al. [5]. This work has been extended using progressive-refinement Monte Carlo [12, 13], and progressive-refinement with spherical harmonic basis functions [15]. With deterministic methods clustering has been used to reduce the number of interactions between surfaces [14], and this has been extended to the non-diffuse case [3]. The basic problem with these methods is that they take too much storage to represent a solution with sufficient spatial and directional accuracy. This problem has been addressed by making a coarse solution and then gathering from it [16, 3].

Fournier and his collaborators [4] have promoted a divide-and-conquer strategy. They have observed that if you know all the light that crosses the surface of an imaginary volume enclosing a part of the scene, then the volume can be closed off as long as no new light enters the volume. If new light does enter the volume, then the new light can be treated independently of the current solution. The simplification here occurs in the representation of the light flux at the volume surfaces.

## 4 Light volume

Looking over the methods in the last section, we find several compelling features. Jensen’s method uses a statistical preprocess to approximate the field-radiance. Lafortune maintains an adaptive piece-wise constant 5D approximation to field-radiance. Fournier et al. have approximations to radiance at arbitrary virtual surfaces embedded in space. All of these could be used to avoid explicitly and recursively querying complex geometry. In addition, they all decouple the structure used to store and use the radiance from the actual scene geometry.

What we would like is something that combines the explicit representation of Fournier et al. and Lafortune, with the statistical particle tracing generality of Jensen. These considerations lead us to offer an alternative abstraction we call the *light volume*. The light volume  $L_V(\mathbf{x}, \hat{\omega})$  represents light as it flows through space, rather than only on surfaces. It is defined for all points and all directions. At any point,  $L_V$  approximates the field radiance.

By approximating the light through space, the work at a surface is automatically extended to all objects near which the light passes. Also, the information is stored as it is used, that is, as a kind of average. In each volume, we store the average value of the radiance in that volume, rather than the flux at surfaces, as Fournier does.

We do not believe that our method is capable of handling all the requirements we think should be asked of a rendering algorithm. Rather, we present it as an example of an algorithm that uses Monte-Carlo, and that decouples the radiance from the scene geometry.

We chose a piece-wise constant function as our light volume. Within each piece, the radiance is approximated by its average value. Although other approximations could be chosen, we believe that the simplicity of a constant function makes it appropriate to test the feasibility of the light-volume concept.

To store light volume, we use a 5-dimensional data structure. We first divide Euclidean space into regular cells. Within each cell, we then extend the data structure into two more dimensions to account for the dependence on direction. The effect is a 3-dimensional grid in Euclidean space of 2-dimensional grids in directional space.

Each directional cell covers a set of directions. To simplify the grid representation we map the set of all directions to two squares, one for each hemisphere, with a measure-preserving function [13].

#### 4.1 Computing the light volume

The light volume is computed through a particle trace. Particles carrying power are sent from all initial luminaires. Particles are scattered from surfaces according to the BRDF of the surface.

As the particles travel through light volumes, their contribution is tallied to compute the light volume. The average radiance in a Cartesian product of a convex volume  $V$  and a convex set of solid angles  $\Omega$  is

$$L_{ave} = \frac{1}{V\Omega} \int_{\Omega} \int_V L dv d\omega \tag{2}$$

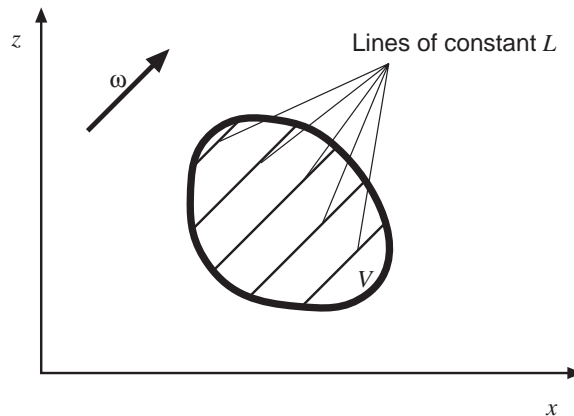


Fig. 1. Definition of our light volume.

Because radiance is constant along lines in the volume if it is empty, we can eliminate one of the dimensions of integration, and integrate over projected areas and directions:

$$L_{\text{ave}} = \frac{1}{V\Omega} \int_{\Omega} \int_{A(\omega)} L\ell(x, \omega) dx d\omega \quad (3)$$

where  $A(\omega)$  is the projected area of  $V$  normal to the direction  $\omega$ , and  $\ell$  is the length of the line segment contained in  $V$  that passes through  $x$  and is in the direction  $\omega$ . Because the distribution of  $L$  is exactly that of the impacting particles, we can normalize  $L$  to get

$$L_{\text{ave}} \approx \frac{1}{NV\Omega} \int_{\Omega} \int_{A(\omega)} L dx d\omega \sum_{i=1}^N \ell(\mathbf{x}_i), \quad (4)$$

where  $\ell(\mathbf{x}_i)$  is the length of the path taken by particle  $\mathbf{x}_i$  through the volume, and  $N$  is the total number of particles. Since the integral above is estimated by the number  $N$  of particle hits times the power  $\Phi$  carried by each particle, we finally have

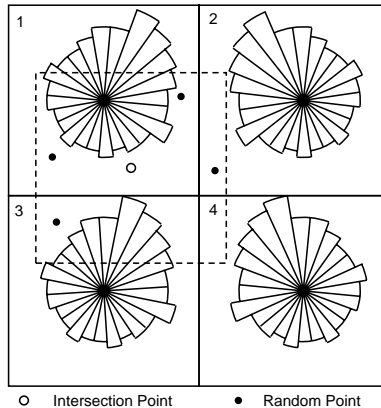
$$L_{\text{ave}} \approx \frac{\Phi}{V\Omega} \sum_{i=1}^N \ell(\mathbf{x}_i) \quad (5)$$

To find  $L_V(\mathbf{x}, \hat{\omega})$  we first locate the Euclidean cell containing  $\mathbf{x}$ , then within that Euclidean cell we locate the directional cell containing  $\hat{\omega}$ . The value stored there is then returned.

To improve the accuracy of the approximation it is important to have a good estimate how much of the cell is actually illuminated. To obtain this estimate and reduce the number of cells, binary subcells are maintained within each Euclidean cell. As a particle pass through a cell, these subcells are flipped on. If a particle hits a surface within a subcell, rays are sent from uniformly distributed random locations within the subcell back toward the particle's origin. The percentage of rays that are unblocked along with the percentage of subcells that are flipped on is an estimate of how much of that subcell is illuminated. This forms the approximation for  $V$ . We believe that an adaptive technique will ultimately provide even better results.

## 4.2 Selecting Light Volumes

Merely choosing the cell containing the illuminated point would result in discontinuities at the cell boundaries. These can be avoided by using a weighted average of nearby cells. An effective way to implement this to generate one or more uniformly distributed random points within a cell-sized box centered on the illuminated point. The cell containing these random points are then integrated, and the resulting radiances are averaged. Figure 2 is a 2-D representation of four cells with light volume cross sections.



**Fig. 2.** Selecting light volumes for averaging

## 5 Example Applications

We have successfully used the light volume in couple of different ways.<sup>1</sup> In Figure ??(a) the light volume is used directly for all radiance values. That is both emitted and reflected particles are averaged and stored in the light volumes during the particle trace. The lack of artifacts in the image is due to the fact that the scene totally diffuse and the table is aligned with the grid of light volumes. Admittedly this is not too interesting. In Figures ??(b - d) only reflected particles were averaged and stored during the particle trace. Thus shadow rays are necessary to compute the direct component. In Figure ??(c) and ??(d) particles which were transmitted through the table top were considered reflected and were thus stored in the light volumes. During the ray tracing stage, shadow rays shot toward the light source from below the glass table top are blocked but the values stored in the light volume give a reasonable approximation to the transmitted light.

Figures 4(a - f) were also rendered using direct lighting and light volumes in which only indirect radiance was stored. However we also added some modifications inspired by Rushmeier et al. [11]. For a diffuse or mostly diffuse reflection a threshold was introduced which determined whether or not the path of the ray is terminated. This makes certain that details like the crack between the refrigerator and the wall do not get washed out. If a reflected ray intersects an object within the threshold, path tracing is continued, otherwise the path is terminated and the light volume is used. The size of the threshold was determined somewhat arbitrarily. In our case, we use the diagonal of a grid cell as the threshold. A Specular or nearly specular reflection was allowed to continue until intersecting a diffuse surface or reaching a maximum ray depth. It should be noted that the only lambertian surfaces Figure 4 are the walls and ceiling.

<sup>1</sup> Color images can be found online at:  
<http://www.cs.indiana.edu/hyplan/kuzimmer/images/egrw96/egrw.html>

The difference between the left column and the right column in Figure 4 is in the tiles. The tiles on the left are smooth and each are represented with a few thousand triangles. The tiles on the right are rippled and each is represented by 500,000 triangles. The point of these images is that sometimes it is necessary to have millions of polygons in order to obtain subtle surface effects.

The images in the left column of Figure 4 were rendered using a 5,000,000 particle prepass and a 49 sample per pixel ray tracing pass. The particle trace took approximately 30 minutes and the rendering took approximately 2.75 hours using a Silicon Graphics Power Challenge with ten R8000 processors running at 75Mh. The images in the right took approximately 40% longer for both the particle trace and the rendering. We used a regular grid with 9600 cells to store the light volumes which, in turn, each stored 242 directions. To keep the implementation simple we chose the grid size to be the same as the grid used to store the geometry. The number of directions is a parameter. Because we only store light volumes in grid cells where geometry exists, the light volume grid for the images in Figure 4 required only about 2Mb of storage which is nominal when compared to the 16Mb of data used for the images in the right column.<sup>2</sup> A traditional Monte Carlo Path Tracing solution using 100 samples per pixel is shown in Figure 3. This picture took approximately 12 hours using same data set as the right column of Figure 4.



**Fig. 3.** Monte Carlo Path Tracing using 100 samples per pixel.

---

<sup>2</sup> Yes, only one tile was actually stored. The rest were instances.

## 6 Where do we go from here

Though effective for some difficult scenes, as presented the light volume has serious problems: a priori meshing, view independence (prevents the use of large databases), static directional resolution (no caustics) and volume interference to name a few.

However, we think we can draw some broad conclusions from our experiences with it. We believe that making full use of the information gained from some kind of particle tracing preprocess is a good approach. Only some kind of stochastic method can hope to avoid accessing every one of the millions of primitives full modelling will require. The speed of modern computers means that many more particles can be traced than can fit in main memory, however, so some kind of information condensation must be used.

We also think some kind of importance or bidirectional approach is needed. Veach [17] has made it clear that sometimes working from the light source is good, and sometimes working from the view point is good. Fournier's [4] insight into how geometry and radiance can be interchanged might provide a way to utilize Rushmeier's geometric simplification ideas.

We have also found that comparing methods is difficult without a standard set of scenes. We would like to see the graphics community agree upon a standard database ranging from simple static scenes to complex procedural ones.

The interaction between complex geometry and complex transport poses one of the greatest challenges to computer graphics. Although it is clear that sophisticated texture techniques [8] coupled with blending techniques [2] can provide convincing realism without many millions of primitives, we believe that ultimately fully modelling most objects will prove to be the most flexible. The advantages are especially desirable if the objects are dynamically changing such as waves acting on sand, or wind on grass. Without addressing these issues, we can never hope to realistically render such common scenes as a shoe underneath a glass table, except as a carefully crafted special case.

## Acknowledgments

This work was supported by Indiana University Faculty Start-up Funds, and the National Science Foundation under grant NSF-CCR-92-09457 and NSF-CCR-94-01961, and used equipment purchased with funds from CISE RI grant 92-23008 and NSF II grant CDA 93-03189. In addition, this work was supported by the NSF/ARPA Science and Technology Center for Computer Graphics and Scientific Visualization (ASC-8920219) and by NSF CCR-9401961 and performed on workstations generously provided by the Hewlett-Packard Corporation.

## References

1. James Arvo, Kenneth Torrance, and Brian Smits. A framework for the analysis of error in global illumination algorithms. *Computer Graphics*, 28(3), July 1994. ACM Siggraph '94 Conference Proceedings.

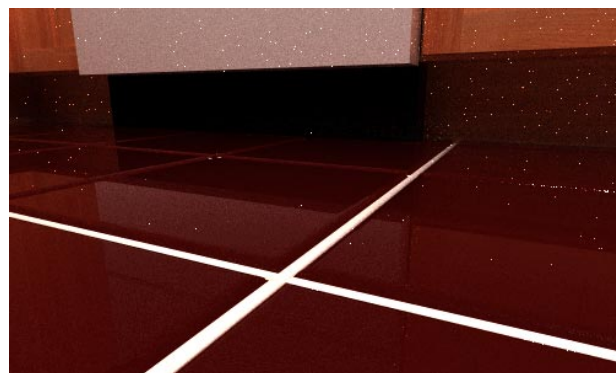
2. Barry G. Becker and Nelson L. Max. Smooth transitions between bump rendering algorithms. *Computer Graphics*, pages 183–190, August 1993. ACM Siggraph '93 Conference Proceedings.
3. Per H. Christensen, Dani Lischinski, Eric Stollnitz, and David Salesin. Clustering for glossy global illumination. Technical Report 95-01-07, Department of Computer Science and Engineering, University of Washington, Seattle, Washington, January 1995.
4. Alain Fournier. From local to global illumination and back. In *Eurographics Rendering Workshop 1995*. Eurographics, June 1995.
5. David S. Immel, Michael F. Cohen, and Donald P. Greenberg. A radiosity method for non-diffuse environments. *Computer Graphics*, 20(4):133–142, August 1986. ACM Siggraph '86 Conference Proceedings.
6. Henrik Wann Jensen. Importance driven path tracing using the photon map. In *Rendering Techniques '95*. Springer-Verlag/Wien, 1995.
7. James T. Kajiya. The rendering equation. *Computer Graphics*, 20(4):143–150, August 1986. ACM Siggraph '86 Conference Proceedings.
8. James T. Kajiya. Rendering fur with three dimensional textures. *Computer Graphics*, 23(3):271–280, July 1989. ACM Siggraph '89 Conference Proceedings.
9. Arjan F. Kok. Grouping of patches in progressive radiosity. In *Proceedings of the Fourth Eurographics Workshop on Rendering*, pages 221–231, 1993.
10. Eric Lafortune and Yves D. Willems. A 5d tree to reduce the variance of monte carlo ray tracing. In *Rendering Techniques '95*. Springer-Verlag/Wien, 1995.
11. Holly Rushmeier, Charles Patterson, and Aravindan Veerasamy. Geometric simplification for indirect illumination calculations. In *Graphics Interface '93*, pages 227–236, May 1993.
12. Bertrand Le Saec and Christophe Schlick. A progressive ray-tracing-based radiosity with general reflectance functions. In *Proceedings of the Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics*, pages 103–116, June 1990.
13. Peter Shirley. Discrepancy as a quality measure for sampling distributions. In *Eurographics '91*, pages 183–193, September 1991.
14. François X. Sillion. Clustering and volume scattering for hierarchical radiosity calculations. In *Proceedings of the Fifth Eurographics Workshop on Rendering*, pages 105–118, June 1994.
15. François X. Sillion, James Arvo, Stephen Westin, and Donald Greenberg. A global illumination algorithm for general reflection distributions. *Computer Graphics*, 25(4):187–196, July 1991. ACM Siggraph '91 Conference Proceedings.
16. Brian E. Smits, James R. Arvo, and David H. Salesin. A clustering algorithm for radiosity in complex environments. *Computer Graphics*, 28(3):435–442, July 1994. ACM Siggraph '94 Conference Proceedings.
17. Eric Veach and Leonidas J. Guibas. Optimally combining sampling techniques for monte carlo rendering. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 419–428. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995.
18. Gregory J. Ward. The radiance lighting simulation and rendering system. *Computer Graphics*, 28(2), July 1994. ACM Siggraph '94 Conference Proceedings.



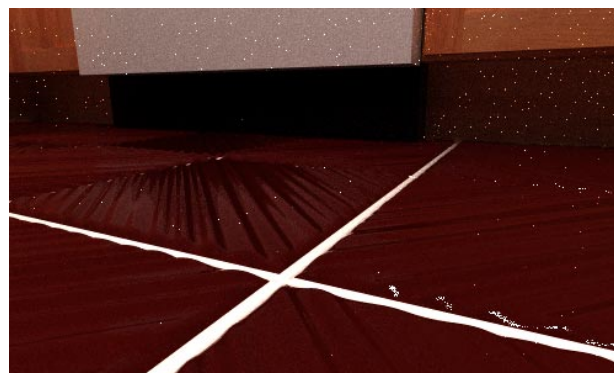
(a)



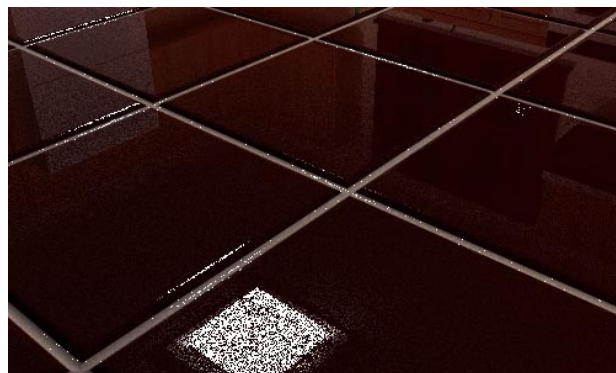
(b)



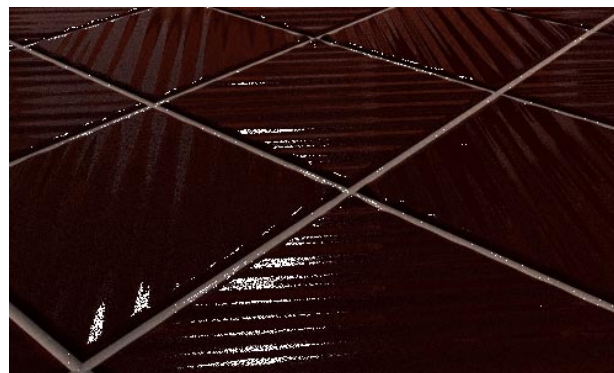
(c)



(d)



(e)



(f)

**Fig. 4.** Author's kitchen: (c - d) Tile details, Each image uses the light volume for indirect illumination. Rendering is done with 49 samples per pixel. The tiles in the left hand column contain 12,800 tiles per tile. Tiles on the right contain 568,178 triangles. Please view the color images on the web at <http://www.cs.indiana.edu/hyplan/kuzimmer/images/egrw96/egrw.html>