Notes on Adaptive Quadrature on the Hemisphere

Technical Report Number 411

Peter Shirley* and Kenneth Chiu†
Department of Computer Science
Indiana University
Bloomington, IN 47405

1 Introduction

This report discusses several issues that arise when integrating field radiance functions on the hemisphere. Rather than representing new work, it is an expression of what issues underly our current research strategy for scenes that contain specular surfaces. It outlines several basic practical concerns, and identifies problems in adaptive sampling strategy that remain unresolved. It also provides some background on how the images in [4] were produced.

Many modern rendering algorithms perform one or many "gathers" at each pixel from some type of world-space global illumination solution. This can be thought of as a quadrature over a hemisphere (for opaque receiving surfaces) that uses an approximate field radiance[1] derived from a zonal (radiosity) solution:

$$L(x,\omega) \approx L_e(x,\omega) + \int_\Omega \rho(x,\omega,\omega')\bar{L}'(x,\omega')\cos\theta d\mu(\omega') \quad (1)$$

where $L(x,\omega)$ is the surface radiance at a point x in direction ω, $L_e(x,\omega)$ is the emitted component of the surface radiance, Ω is the set of incoming directions ω', $\rho(x,\omega,\omega')$ is the BRDF at x, $\bar{L}'(x,\omega')$ is the approximate field radiance incident at x from direction ω', θ is the angle between the surface normal vector at x and ω', and μ is the solid angle measure. The approximate field radiance function $\bar{L}'(x,\omega')$ is typically evaluated by firing rays from x into a zonal (radiosity) solution over a discretized environment. This zonal solution can consist of diffuse or more general BRDF surfaces, and may not have values for specular or near specular surfaces. Our strategies for calculating $\bar{L}'$ were inspired by [12], but the idea of using an approximate field radiance goes back to at least [5]. An example of a high resolution image that shows L and the low resolution radiosity solution used to calculate $\bar{L}'$ are shown in Figure 1.

In these notes we examine the issues that arise when trying to numerically evaluate Equation 1 when $\bar{L}'(x,\omega')$ is potentially complicated (e. g., contains small spatial features with large values such as luminaires). This treatment differs somewhat from previous techniques used in graphics in that we will assume very little a priori knowledge of $\bar{L}'(x,\omega')$, and will assume that $\bar{L}'(x,\omega')$ contains potentially significant contributions from small objects and reflections of small objects. Our rationale for why some scenes imply the desirability of such assumptions is in [4].

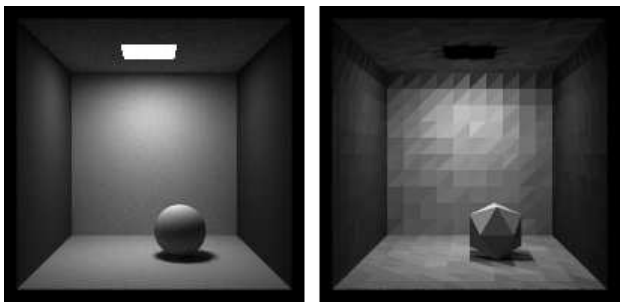

Figure 1: Left: rendered image. Right: low resolution radiosity image used to calculate $\bar{L}'$ at each visible point in the image to the left.

In Section 2 we explain our assumptions about $\bar{L}'$, and why this implies we need a naive adaptive algorithm. In Section 3 we categorize previous adaptive quadrature techniques used in rendering. In Section 4 we discuss how feature characteristics in the integrand influences adaptation strategy. In Section 5 we discuss how to extend the results of Section 3 to integrals with hemispherical domains. In Section 6 we discuss our current attempts to improve our quadrature strategy.

2 Characteristics of Field Radiance Function

Some of the bright parts of the field radiance function can be *virtual luminaires* (reflections of luminaires). These are hard to identify in complex environments because a set of luminaires and mirrors can cause an arbitrarily large number of virtual luminaires. This makes any method that preferentially treats areas covered by luminaires suspect because real and virtual luminaires are treated differently. Working backwards from the luminaires can be effective (e. g., [1, 15, 6]) but this method usually stores some information on the primitives which is problematic if the geometry is procedural. We could also use the *image method*[13] to account for specular transfer, but this is difficult when mirrors are not planar.

*Email: shirley@graphics.cornell.edu. Current address: Program of Computer Graphics, 580 Engineering and Theory Center, Cornell University, Ithaca, NY 14853.

†Email: chiu@cs.indiana.edu

[1] We are using the terminolgy of [2]: the *field radiance* function describes the incident radiance from a direction at a point, and the *surface radiance* function describes the outgoing radiances at a point in a direction.

Virtual luminaires are often not important. However, if we can handle them correctly, then we can painlessly model several basic effects, such as varnish on wood. Also, dew or frost on grass or a pane of glass might also be something difficult to portray accurately without virtual luminaires. Windows can be faked by simply ignoring the glass when tracing shadow rays, but what about architectural simulations of buildings that actually have virtual luminaires in their design? For example, an architect might put a fountain in the middle of a mall with a sky light above it, counting on the virtual luminaires to provide the effect he desires. Or he might use faceted glass to break up the light coming through a sky light. Also most light bulb and reflector combinations cast some kind of pattern into the environment. All of these situations can probably be handled with special case code, but it is attractive to have a general method that is robust against greatly varied scene characteristics.

3 Adaptive Quadrature on the Unit Square

Although Equation 1 has the unit hemisphere as the domain of integration, in this section we will examine integration with the domain $[0, 1]^2$. In Section 5 we will extend this discussion to the hemisphere. Consider the integral I over a square two dimensional domain $\Omega = [0, 1]^2$

$$I = \int_{\Omega} L(x, y) dA$$

where L(x, y) is the integrand value of the point (x, y) on the square. If we think of the function L as the radiance of points on a square pixel, then I would represent the value of a pixel calculated using a width-one box-filter.

3.1 Direct and Indirect Lighting

One way to attack the problem is to break the integrand into two parts, where one part contains the peaks. Suppose that L and Ω are those shown in the upper-lefthand part of Figure 2. Here there are several bright circles on a fairly dim backgound (this might represent lights seen in the distance). The two integrals would then be:

$$I = \int_{\Omega_r} L_r(x, y) dA + \int_{\Omega_e} L_e(x, y) dA$$

Where L_r is the reflected component of the color, and L_e is the emitted component. In practice $\Omega_r = \Omega$ because even luminaires are reflective. Because L_e is zero for the background, the domain Ω_e is simply the union of circle domains. If we have a priori knowledge of what the Ω_e is (i. e., we know where all the bright spots might be), the techniques of Ward[16] or Shirley and Wang[14] can be used.

3.2 Naive Adaptive Quadrature

Unfortunately, when specular reflections or transmissions of luminaire images might be seen (virtual luminaires), then we will lack a priori knowledge of Ω_e. A more naive integration strategy on the square could find these virtual luminaires, however. There are integration strategies that work directly on I without looking at emitted versus reflected properties as shown in Figure 3. Glassner suggests using *adaptive weighted Monte Carlo integration*, where samples are generated adaptively, placing more samples in "interesting" areas[7]. The estimate for I is then taken by using the Voronoi diagram to decide the weight for each sample:

$$I \approx \sum_{i=1}^{N} A_i L(x_i, y_i)$$

where A_i is the area of the Voronoi cell for sample (x_i, y_i). The attrative part of this scheme is that samples can be chosen arbitrarily because a reweighting is carried out as the last step.

Many authors have modified Whitted's *adaptive subdivision* to include randomness to avoid aliasing (e. g., [11]). The chief problem with this method is that if important features are missed the error can be high. Also, great care needs to be taken if bias needs to be avoided[9]. An adaptive scheme has also been used that places sample points using an adaptive triangulation scheme on the hemisphere[3].

Kirk and Arvo[10]suggest *domain partioning* and calculating the sum of integrals of L over each subdomain by Monte Carlo Integration. Domains are chosen so that they will include luminaires, and more samples are taken in these domains. This strategy is shown at the top of Figure 3.

We have used two different criteria for subdividing cells. The first criteria is based on the contrast weighted by the cell size and BRDF. This criteria attempts to capture the likely maximum error contribution of each cell. In other words, if the integrand in this cell is actually much more like one of its neighbors than revealed by its sample value, how much error could this contribute to the final radiance?

Another criteria is simply value weighted by cell size and BRDF. This sends samples in brighter regions because they have a greater contribution to the final radiance.

Which of these criteria should be used depends on the nature of the luminaires. If the luminaires are more or less compact, then we won't expect any important features internal to the projected boundary. In this case, determining the edges of the luminaires accurately will reduce variance. If, however, the luminaires are very irregular in shape, perhaps because they are virtual luminaires, then we cannot expect them to have projected boundaries that are short relative to their projected area. In this case, we might as well give up, and just sample where it is bright.

Using the Delaunay triangulation instead of a quadtree to partition and weight samples is easier to work with, because of the regularity imposed by the triangulation (i. e., each region always has a fixed number of neighbors, etc.), and should also produce better results. We believe that the improvements will only be incremental, however.

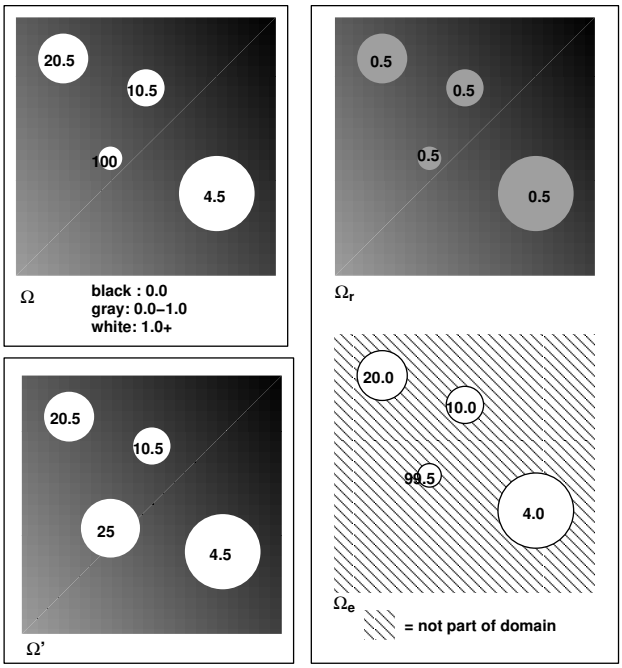

Figure 2: Domains of integration.

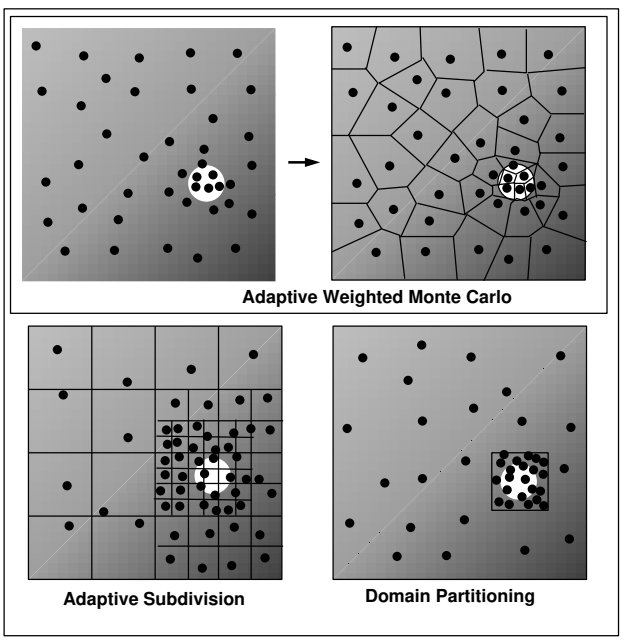

Figure 3: Methods of integration.

4 Dealing with Small Features in the Integrand

The adaptive strategies discussed in the last section can be divided into two categories: those that use a priori knowledge of the integrand to preferentially sample in areas of the integration domain likely to have large integrand values (e.g. shadow rays in Kajiya's path tracing[8]), and those that sample naively on the square. Unfortunately, the former will have trouble accounting for virtual luminaires, and the latter will always have the potential to miss small bright sources (e.g. a halogen filament). In this section we discuss why this problem is difficult and speculate on solutions to it.

Suppose our image is made up of a bright luminaire with value $1.0/A$ and projected area A on the pixel, and a background with value 0.5 and area $(1-A)$. The value of I is:

$$I = (1-A)*0.5 + A(1/A) = 1 + (1-A)/2 = (3-A)/2$$

For very small values of A, I will be very near 1.5. If we approximate I by a method such as adaptive quadrature shown in Figure 3, then our estimate will be quite good if any of the initial samples hit the source, but will be 0.5 if none of them do. Without a priori knowledge that the bright region is in the pixel, it will be hard to design a method that is guaranteed to sample the bright spot without an excessive number of initial samples. This will make the image somewhat dark on average; this is a special case of the bias discussed in [9].

However, we could approximate I by increasing the size of the luminaire to have a projected area kA and value (1/(kA)). The value of this integral I' would then be:

$$I' = (1-kA)*0.5 + kA(1/kA) = 1 + (1-kA)/2 = (3-kA)/2$$

So our error would be:

$$E(I') = I - I' = (3-A)/2 - (3-kA)/2 = (k-1)A/2$$

so as long as kA is small, or the background value is small, this method has potential. This basic idea is shown in the lower-left corner of Figure 3.

How might this be used in practice? Suppose we use 100 samples. If the samples are jittered, then there is a 100 percent chance that disks of area $\pi(1/10^2 + 1/20^2)$ are detected, so each disk is replaced by a bigger one. This introduces systematic error, but for many purposes this is acceptable. Certain kinds of errors are visually disturbing, such as aliasing, but very low frequency errors, and high frequency random errors (noise) are tolerated well by our visual system.

Another strategy would be the use of shrinking luminaires to locate important directions. The initial luminaire size would be determined to satisfy some likelihood of detection criterion. For example, if the luminaires were spheres, and the samples were jittered, we could choose the initial size such that the luminaires were guaranteed to be detected. During successive resampling, the size of the luminaire could be coupled to the sample density such that at least one hit was always guaranteed. In the presence of shadows and curved reflectors, such an assurance would be difficult if not impossible,

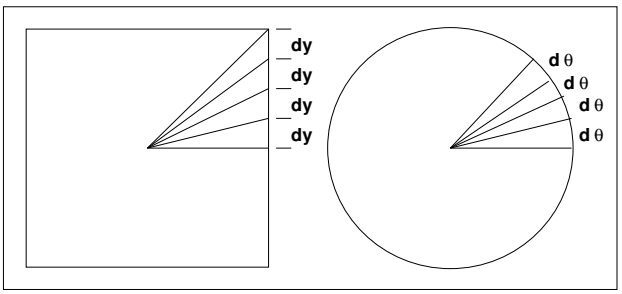

Figure 4: Area preserving transform from square to disk uses equal area strips. A similar transform is used to go from square to sphere.

but probabilistic methods can also be satisfactory. When the luminaires are shrunk to correct size, we have now determined at least one ray per luminaire that hits that luminaire, and we can use that information to perform adaptive quadrature.

5 Extensions to the Hemisphere

To actually evaluate the radiance at a point we need to evaluate Equation 1 which has the hemisphere of directions as the domain of integration. In spherical coordinates this becomes:

$$L(x,\theta,\phi) \approx L_e(x,\theta,\phi) + \int_0^{2\pi}\int_0^{\frac{\pi}{2}} \rho(x,\theta,\phi,\theta',\phi')\bar{L}'(x,\theta',\phi')\cos\theta'(\sin\theta' d\theta' d\phi') \quad (2)$$

In principle, we could transform this to an integral on $[0,1]^2$ by simply substituting θ with $(2/\pi)a$ and ϕ with $b/(2\pi)$. Although this is convenient for non-adaptive quadrature, it causes practical problems for adaptive implementations. The main problem is that points that are nearby in (θ,ϕ) space might be far apart in (a,b) space–for example (θ,ϵ) and $(\theta,2\pi-\epsilon)$ are nearby in (θ,ϕ) space if ϵ is small, but are quite far apart in (a,b) space because a "cut" is introduced at $\phi = 0$. This can play havoc in (a,b) space if the concept of nearness is used to select new sample points. Another problem is that features that have a small aspect ratio in (θ,ϕ) (e.g. the directions subtended by a light bulb) will typically have a larger aspect ratio in (a,b) space (features are streched), and can thus be harder to locate in an initial sampling phase.

To avoid the problems with a simple (θ,ϕ) to (a,b) mapping, we seek a mapping that preserves adjacency, aspect ratio, and fractional area. This is probably an overconstrained transform, but we use one that preserves adjacency, area, and does not change aspect ratio too much. The pinciple behind the basic transform is illustrated in Figure 4 for a square to disk. The square to hemisphere transform is slightly more involved to keep the fractional area property, and the code for this transform is available in the Appendix. The transform from a checkerboard on $[0,1]^2$ to the himisphere is shown in Figure 5. The use of this transform allows us to work in the unit square which simplifies code design a great deal, although some overhead is added.

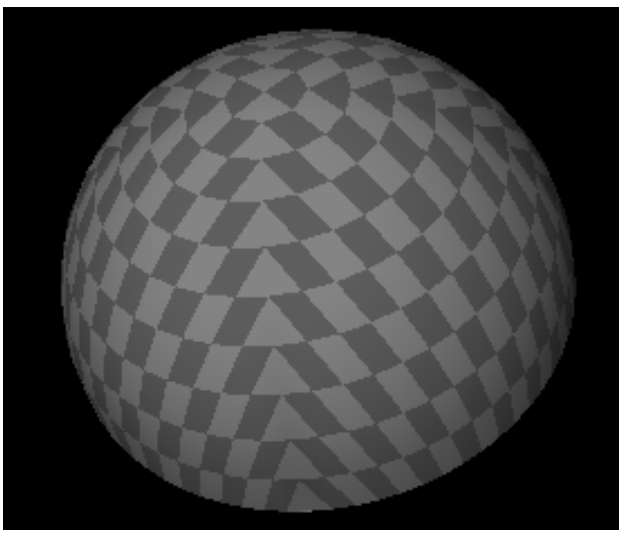

Figure 5: Map of a checkerboard on $[0,1]^2$ to the hemisphere.

6 Discussion

Assuming we wish to do an adaptive quadrature of a complex field radiance function, we either need to change the integrand to make the problem easier to solve, or we need a good predictor for where large values of the integrand are expected. Of course, we can also try using both tactics. Nothing precludes us from sending shadow rays, and also searching for virtual luminaires. Whether these shadow rays are useful will depend on the scene.

Also, we should not overlook the usefulness of human input. The user already controls such parameters as lighting, colors, textures, movements, positions, etc. Giving him additional control of the rendering algorithm should not be considered taboo.

Acknowledgements

Special thanks to Andrew Glassner who got us interested in high-cost quadrature techniques during his visit to Indiana University in 1993. Thanks to Jim Arvo for help on terminology. Thanks to Bruce Shei for keeping software working on weekends and evenings when it was needed most.

This work was supported by Indiana University and by NSF grant *True Virtual Reality Systems* NSF-CCR-92-09457.

Appendix: Code for square to sphere mapping

The following code accomplishes the square to sphere mapping discussed in Section 5. It is available via anonymous ftp at `ftp.moose.cs.indiana.edu` (129.79.254.191) in `pub/chiuk`.

```
/*
 * This function takes a point in the unit square,
 * and maps it to a point on the unit hemisphere.
 *
 * Copyright 1994 Kenneth Chiu
 *
 * This code may be freely distributed and used
 * for any purpose, commercial or non-commercial,
 * as long as attribution is maintained.
 */

void
map(float x, float y, float *x_ret, float *y_ret, float *z_ret) {

    float xx, yy, offset, theta, phi;

    x = 2*x - 1;
    y = 2*y - 1;

    if (y > -x) {  // Above y = -x
        if (y < x) {   // Below y = x
            xx = x;
            if (y > 0) {  // Above x-axis
                /*
                 * Octant 1
                 */
                offset = 0;
                yy = y;
            } else {     // Below and including x-axis
                /*
                 * Octant 8
                 */
                offset = (7*M_PI)/4;
                yy = x + y;
            }
        } else {    // Above and including y = x
            xx = y;
            if (x > 0) {   // Right of y-axis
                /*
                 * Octant 2
                 */
                offset = M_PI/4;
                yy = (y - x);
            } else {     // Left of and including y-axis
                /*
                 * Octant 3
                 */
                offset = (2*M_PI)/4;
                yy = -x;
            }
        }
    } else {   // Below and including y = -x
        if (y > x) {   // Above y = x
            xx = -x;
            if (y > 0) {   // Above x-axis
                /*
                 * Octant 4
                 */
                offset = (3*M_PI)/4;
                yy = -x - y;
            } else {  // Below and including x-axis
                /*
                 * Octant 5
                 */
                offset = (4*M_PI)/4;
                yy = -y;
            }
        } else {     // Below and including y = x
            xx = -y;
            if (x > 0) {   // Right of y-axis
                /*
                 * Octant 7
                 */
                offset = (6*M_PI)/4;
                yy = x;
            } else {  // Left of and including y-axis
                if (y != 0) {
                    /*
                     * Octant 6
                     */
                    offset = (5*M_PI)/4;
                    yy = x - y;
                } else {
                    /*
                     * Origin
                     */
                    *x_ret = 0;
                    *y_ret = 1;
                    *z_ret = 0;
                    return;
                }
            }
        }
    }

    theta = acos(1 - xx*xx);
    phi = offset + (M_PI/4)*(yy/xx);

    *x_ret = sin(theta)*cos(phi);
    *y_ret = cos(theta);
    *z_ret = sin(theta)*sin(phi);
}
```

References

[1] James Arvo. Backward ray tracing. *Developments in Ray Tracing*, pages 259–263, 1985. ACM Siggraph '85 Course Notes.

[2] James Arvo. The irradiance jacobian for partially occluded polyhedral surfaces. *Computer Graphics*, 28(3), July 1994. ACM Siggraph '94 Conference Proceedings.

[3] Markus Beyer and Brigitta Lange. Rayvolution: An evolutionary ray tracing program. In *Proceedings of the Fifth Eurographics Workshop on Rendering*, pages 137–146, June 1994.

[4] Kenneth Chiu and Peter Shirley. Rendering, complexity and perception. In *Proceedings of the Fifth Eurographics Workshop on Rendering*, pages 19–33, June 1994.

[5] Micheal F. Cohen, Donald P. Greenberg, David S. Immel, and Philip J. Brock. An efficient radioisty approach for realistic image synthesis. *IEEE Computer Graphics and Applications*, 6(2):26–35, 1986.

[6] Steven Collins. Adaptive splatting for specular to diffuse light transport. In *Proceedings of the Fifth Eurographics Workshop on Rendering*, pages 119–135, June 1994.

[7] Andrew S. Glassner. Dynamic stratification. In *Proceedings of the Fourth Eurographics Workshop on Rendering*, pages 5–14, 1993.

[8] James T. Kajiya. The rendering equation. *Computer Graphics*, 20(4):143–150, August 1986. ACM Siggraph '86 Conference Proceedings.

[9] David Kirk and James Arvo. Unbiased sampling techniques for image sysnthesis. *Computer Graphics*, 25(4):153–156, July 1991. ACM Siggraph '91 Conference Proceedings.

[10] David Kirk and James Arvo. Unbiased variance reduction for global illumination. In *Proceedings of the Second Eurographics Workshop on Rendering*, 1991.

[11] James Painter and Kenneth Sloan. Antialiased ray tracing by adaptive progressive refinement. *Computer Graphics*, 23(3):281–288, July 1989. ACM Siggraph '89 Conference Proceedings.

[12] Holly Rushmeier, Charles Patterson, and Aravindan Veerasamy. Geometric simplification for indirect illumination calculations. In *Graphics Interface '93*, pages 227–236, May 1993.

[13] Holly E. Rushmeier and Kenneth E. Torrance. Extending the radiosity method to include specularly reflecting and translucent materials. *ACM Transaction on Graphics*, 9(1):1–27, January 1990.

[14] Peter Shirley and Changyaw Wang. Direct lighting by monte carlo integration. In *Proceedings of the Second Eurographics Workshop on Rendering*, 1991.

[15] Brian E. Smits, James R. Arvo, and David H. Salesin. An importance-driven radiosity algorithm. *Computer Graphics*, 26(2):273–282, July 1992. ACM Siggraph '92 Conference Proceedings.

[16] Greg Ward. Adaptive shadow testing for ray tracing. In *Proceedings of the Second Eurographics Workshop on Rendering*, 1991.