Computing Illumination from Area Light Sources by Approximate Contour Integration

Christophe Vedel
LIENS, CNRS URA 1327, Ecole Normale Superieure,
45, rue d'Ulm, 75230 Paris Cedex 05, France
e-mail address: vedel@dmi.ens.fr

ABSTRACT

A new method using approximate contour integration to accurately compute direct illumination from diffuse area sources in presence of curved obstacles is presented. All visibility tests are done using ray tracing so the method can be applied to a large class of objects.

Computation of illumination on a pixel by pixel basis is necessary to accurately capture sharp shadows. However in soft penumbra zones many shadow rays are needed to quantize the penumbra finely enough and avoid banding artifacts. Furthermore, these zones usually cover lots of pixels. We make use of the fact that silhouettes of objects in a scene are smooth for the most part to replace them by polygonal lines in source space. The method allows the estimation of intensity gradients. Penumbras with no aliasing are obtained with fewer rays than with usual adaptive sampling techniques.

KEYWORDS: Rendering, area sources, contour integration, illumination gradients.

1 INTRODUCTION

Most global illumination algorithms build a representation of the radiosity function in object space. In the process of producing shaded images of a scene, this description serves first as the current state of light distribution during the solution of the rendering equation and is then used to reconstruct the shading on surfaces for particular images.

Because of shadows, the radiosity function exhibits many discontinuities. The ones of lower degrees can not be ignored if accurate images of a scene are to be produced [7]. The first attempts at a solution to this problem are adaptive subdivision techniques [4, 1]. Hierarchical structures such as quadtrees allow the representation of light to be refined when high gradients are detected. Images produced with this method look better but sharp shadows can never be perfectly resolved resulting in well known artifacts such as shadow or light leaks and jagged shadow borders.

Recent algorithms have focused on giving better balanced solutions to the global light equilibrium [6, 14]. They use the same hierarchical structure as adaptive subdivision to distribute computations so that the time spent in a light transport operation is, if possible, proportional to the amount of energy involved. As a result, computations are optimized for a given precision. Images obtained with these algorithms suggest that the level of precision that would be needed to produce alias free images exceeds what is reasonable because the eye is much more sensitive to contrast than to the absolute value of the solution. Specific methods not related to the optimization of global physical accuracy have been developed to address this point.

One is to resolve the discontinuities in radiosity by introducing them explicitly in the mesh. Currently this method has only been applied to polyhedral scenes where the most important boundaries are line segments and can be computed [2, 7, 3, 10]. However the complexity of shadows cast by free form objects on free form objects is much greater. Even once the mesh obtained, reconstructing radiosity and prescribing continuity along arbitrary curves would still be a difficult problem.

Another solution is to recompute direct illumination on a pixel by pixel basis during image rendering. If the light sources for which this process is applied are carefully chosen, this technique guarantees that all important shadows and penumbras will be properly accounted for [9]. The drawback of this technique is that lots of shadow rays need to be cast to properly quantize the radiosity in the penumbra of area sources. Techniques to reduce this number using adaptive sampling of the source or Monte Carlo integration have been proposed [13, 8].

This paper introduces approximate contour integration as a new method to solve this problem. In the next section, contour integration is compared to surface sampling techniques and the use of approximate contours in source space is motivated. Section 3 presents the illumination algorithm in detail while section 4 shows how gradients can be computed with little additional cost. Finally results are presented in section 5.

2 SURFACE SAMPLING vs CONTOUR SAMPLING

The form factor between a differential surface element dA_1 and a surface A_2 is defined by the following integral

$$F_{dA_1,A_2} = dA_1 \int_{A_2} vis(dA_1, dA_2) \frac{\cos\theta_1 \cos\theta_2}{\pi r^2} dA_2, \quad (1)$$

where $\vec{r}$ is the vector joining dA_1 and dA_2 and θ_1 (resp. θ_2) is the angle between $\vec{r}$ and the surface normal at dA_1 (resp. dA_2). $vis(dA_1, dA_2)$ is the visibility function of the two differential surface elements. $vis(dA_1, dA_2) = 1$ if the two elements see each other and 0 otherwise. Except for simple configurations, equation (1) can not be solved analytically as is and numerical integration methods are used in practice [16]. These methods approximate the integral by summing the contributions of a finite number of small surface elements on the source. The visibility is tested for each element by tracing rays and the value of each elementary form factor is approximated by the form factor between a differential area and a disc.

With $\vec{n_i}$ the normal vector at dA_i, integral (1) can be rewritten as the flux of a vector field through surface A_2

$$F_{dA_1,A_2} = dA_1 \int_{A_2} \left(vis(dA_1, dA_2) \frac{\vec{n_1} \cdot \vec{r}}{\pi r^4} \vec{r} \right) \cdot \vec{n_2}\, dA_2. \quad (2)$$

Using Stokes' theorem, integral (2) can be transformed into the following contour integral

$$F_{dA_1,A_2} = dA_1 \int_{\partial_{vis} A_2} \frac{\vec{n_1} \times \vec{r}}{2\pi r^2} d\vec{l_2}, \quad (3)$$

where $\partial_{vis} A_2$ is the contour of the visible part of A_2 and $d\vec{l_2}$ is a differential vector element along $\partial_{vis} A_2$.

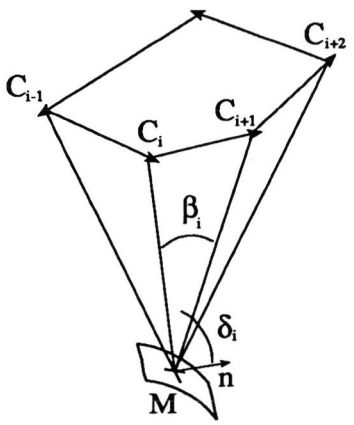

Figure 1: Geometry for contour integration

If the curve describing the contour of the visible part of the source is a polygonal line defined by points $C_0, \ldots, C_n$ as in Figure 1, integral (3) can be put into the following closed form

$$F_{dA_1,A_2} = dA_1 \frac{1}{2\pi} \sum_{i=0}^{n-1} \beta_i \cos(\delta_i), \quad (4)$$

where β_i is the angle between $\overrightarrow{MC_i}$ and $\overrightarrow{MC_{i+1}}$, and δ_i is the angle between the plane defined by M, C_i and C_{i+1}, and the normal $\vec{n}$ at M. Even then, using this formula to compute form factors is costly because for each point where illumination is evaluated, visibility must be solved exactly in object space to get $C_0, \ldots, C_n$. It has however been applied to compute direct illumination [12] and is simpler for discontinuity meshed polyhedral scenes where the visibility problem is partly solved by preprocessing [3, 10]. In the presence of curved objects, the contour is harder to compute and integral (3) cannot be put into closed form in general.

Curved objects have to be tesselated in order to have a polygonal silhouette for contour integration. If the obstacle is near the receiver, it should be tesselated finely so that its sharp shadow does not look polygonal. When it is closer to the source and casts a softer shadow, the tesselation can be coarser. This suggests that the projected silhouette of the obstacle on the source should be polygonized instead of the 3D obstacle itself.

The approach presented here performs the equivalent of tesselation for the part of the 3D silhouette that projects inside the area light source. This part depends on the relative position of the obstacle between the source and the receiver and is, for instance, smaller when the obstacle is near the receiver. Because the algorithm approximates its projection on the source with a given number of line segments, the resolution is tuned automatically.

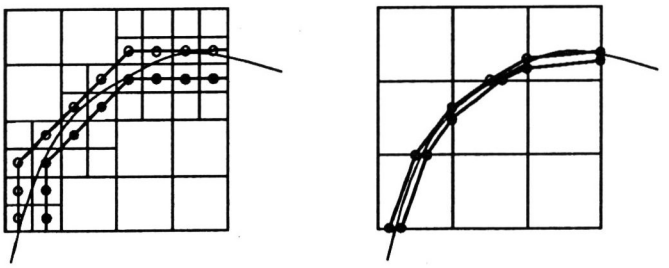

Figure 2: Adaptive surface sampling (left) and contour sampling (right). Using dichotomy with fewer rays, a tighter approximation of the contour is possible.

Figure 2 shows the advantage of contour sampling when the silhouette of objects partially hiding a light source is smooth. When adaptive surface sampling refines the source one level deeper, the form factor is approximated with elements four times smaller. This is optimal for an arbitrary contour but for mostly smooth contours, locating more precisely a few points on the contour and approximating it in between with line segments will give a more accurate form factor for a given number of rays. The difference is similar to the one between the rectangle and the trapezoid rules for numerical

integration.

3 APPROXIMATE CONTOUR INTEGRATION

In this section, a description of the algorithm used to compute the form factor between a differential area dA around point M on the receiving surface and a rectangular light source $\mathcal{S}$ is presented. The normal at M is $\vec{n}$. vis_M is the visibility function from point M, that is $vis_M(P)$ is 1 if P is visible from M and 0 otherwise.

3.1 General Procedure

To build an approximate polygonal contour of the visible part of area light source $\mathcal{S}$, we use a two-dimensional analog of the marching cubes algorithm [11].

The light source is diced into a regular grid of rectangles with sample points $S_{i,j}, i = 0, \ldots, p, j = 0, \ldots, q$, in the corners. $vis_M(S_{i,j})$ is computed by casting a ray (first level ray) from the receiver to each sample point. The squares $(S_{i,j}, S_{i+1,j}, S_{i+1,j+1}, S_{i,j+1})$ are then traversed to identify the pieces of the contour interior to the light source. The visible sections of the border of the light source are also part of the integration domain but their contribution is computed afterwards for the sake of simplicity. Because each of the four corners is either visible or not, there are a priori sixteen possible configurations. Various symmetries reduce this number to the following four distinct cases (shown in Figure 3):

1. All four samples are visible or not at the same time and no contour is detected.

2. One sample differs from the three others. The contour crosses two adjacent sides of the square.

3. Two adjacent samples are visible and the two others are not. The contour crosses two opposite sides of the square.

4. Two opposite samples are visible and the two others are not. The contour crosses the square twice. To decide between case a and b an additional ray is cast to the center of the square.

We first describe the procedure applied to each square to build the approximate contour. The next subsection will present the test used to check the validity of the approximation and the refinement algorithm.

In cases 2, 3, and 4, one or two pieces of contour cross the sides of the square. Following the assumptions made at the end of section 2, the algorithm then locates with more precision the points where the contour crosses the sides of the square instead of refining the square itself. Suppose for instance that $vis_M(S_{i,j}) = 1$ and $vis_M(S_{i+1,j}) = 0$. The segment crossed by the silhouette of the obstacle is narrowed by dichotomy starting with $[S_{i,j} S_{i+1,j}]$. A new ray (second level ray) is cast to the middle point of the segment.

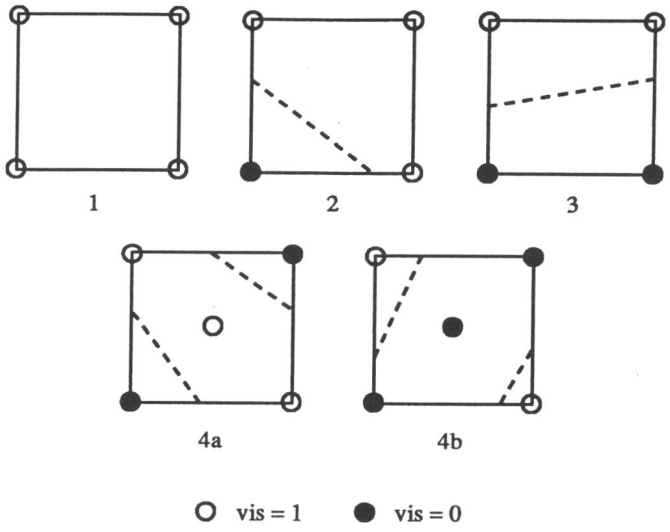

Figure 3: Contour crossing cases

Depending on the value of vis_M at this point, the procedure is repeated recursively on the first or the second half of the segment. The middle point of the final segment becomes an approximate contour vertex. If the true silhouette of the object is simple, this procedure will find one point nearly on it. However if it is more complicated or if there are several small objects, the dichotomy will still return one point, giving no indication of the complexity. These cases will be detected by the procedure described in the next section.

For each successive pair of such vertices, formula 4 gives a contribution to the total form factor. Care must be taken to order the vertices of each pair in a consistent manner for all squares of the source. This can be achieved by always keeping the hidden part of the source to the left.

3.2 Corners

The algorithm described above yields good results when the silhouettes of objects in a scene are smooth curves. However objects have corners and the contour of two different objects can also cross at an angle. If the apex of a corner, for instance, is projected inside one of the squares, it is not detected and part of it is cut from the contour. When crossing the penumbra, severe aliasing occurs because the amount of cut contour beats as the apex slides from square to square on the source (see Figure 6). The algorithm must detect these conditions and apply a special procedure to adjust the form factor.

To check the validity of the approximation in a square, one point is added to the contour. The point chosen depends on the configuration of the square because a new point can only be obtained by dichotomy between a visible sample and a non visible one (see Figure 4):

Case 2 One point is computed on the diagonal of the square. A dichotomy is started with the pair of points (α, δ).

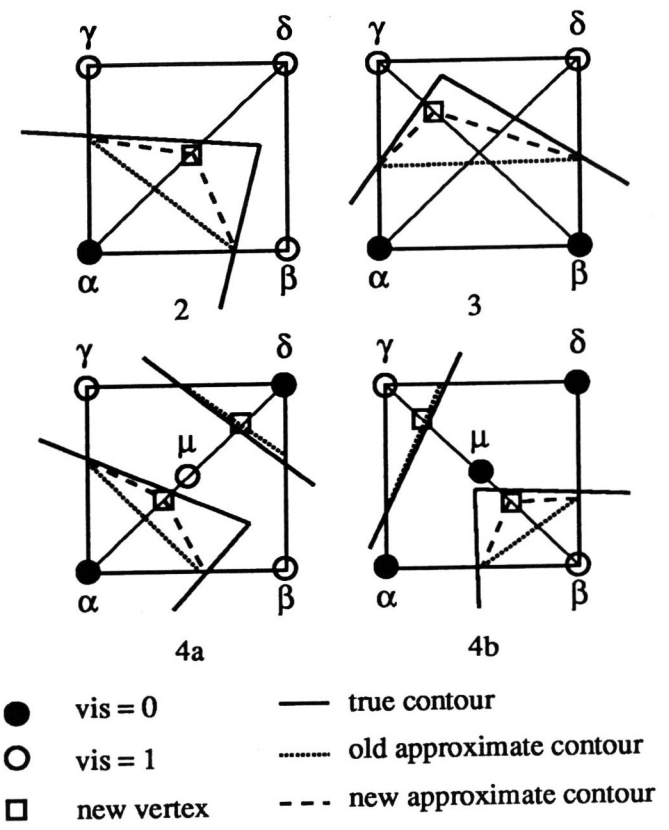

Figure 4: Vertices added to the contour to check for validity.

Case 3 One point is computed on one of the two diagonals of the square. A dichotomy is started with either the pair (α, δ) or (β, γ) chosen at random.

Case 4 One point on each piece of contour is computed by dichotomy on the two semi-diagonals. The center sample μ is already known. Dichotomies are started with the pairs (α, μ) and (μ, δ) or (β, μ) and (μ, γ) depending on the subcase.

The difference of integration between the two contours is the form factor δF for the small part of the source that is missed or added when using the simple version of the contour. Let ϵ be the length of the smallest intervals in the dichotomy process. Because the endpoints of the segment are only known with precision ϵ, the uncertainty on the result of contour integration is the form factor of a strip of width ϵ around the segment. This is also the expected error if the true contour is a line segment and it can be used to test whether δF is an acceptable error or not. Thus the threshold is not directly set by the user but depends naturally on the level of approximation. Its value can be approximated using the formula for the elementary form factor in [16].

If the threshold is exceeded, the square is subdivided like a quadtree and each subsquare is then treated in turn. The number of dichotomy steps is decreased by one so ϵ stays the same and the threshold is divided by two because the length of contour inside the subsquare is smaller. This process can be applied recursively if necessary. When dividing the square, samples are added at the middle point of each side. If the endpoints of the side have the same visibility, the added sample is new and it can happen that its visibility is different. This leads to an inconsistency for the neighboring square and forces its subdivision. If the neighbor had already been treated, its contribution must be subtracted and reevaluated. Furthermore, inconsistencies can propagate from square to square depending on the underlying complexity of the true contour. Because these cases are expected to be rare, we have decided that each time an inconsistency that would require backtracking is detected, the whole source is divided one level deeper and the computation of the form factor begins anew.

3.3 Implementation Note

Given the initial number of samples on the source and the number of dichotomy steps used, an array holding all possible samples is created. All rays traced for dichotomy purposes are traced to points in this array (this is also the case for dichotomies along the diagonals). This way, visibility values computed on the sides of a square are reused for the neighboring squares. When an inconsistency is detected and the process is restarted, the result of previously traced rays is reused.

4 GRADIENTS

Light intensity gradients have been shown to be useful to obtain a better reconstruction of shading across penumbras[17, 15]. They can also be used to reduce the number of shadow rays when computing direct illumination as will be suggested in section 6. In this section, a simple method to compute the gradients from the results of the form factor computation algorithm is presented.

Following [17], we could break down the gradient into a rotational term $\vec{\nabla_r}$ and a translational term $\vec{\nabla_t}$. $\vec{\nabla_r}$ (resp. $\vec{\nabla_t}$) describes the change in form factor when the normal at M is tilted (resp. when M is moved in the tangent plane). Here we have decided to eliminate the influence of the normal on the receiver from the gradient computations. First, formula (4) is expressed as

$$F_{dA_1,A_2} = \left(dA_1 \frac{1}{2\pi} \sum_{i=0}^{n-1} \beta_i \vec{n_i} \right) \cdot \vec{n} = \vec{G}_{dA_1,A_2} \cdot \vec{n} \quad (5)$$

where $\vec{n_i}$ is the normal to the plane defined by M, C_i, and C_{i+1}.

Then $\vec{G}$ is used instead of F to compute partial derivatives when M is moved. To extrapolate the value of F at a point P from the value at M, $\vec{G}$ is extrapolated and F is computed as the dot product of $\vec{G}$ and the true normal at P. This approach has the advantage of uncoupling geometry and shading. Assuming no self shadowing, no more samples

are required to represent lighting on a rippled or bumped surface than on a plane. This technique also simplifies the subdivision criteria for an adaptive radiosity algorithm because the change in normal is not important as long as the source remains above the horizon of the receiver.

The algorithm described in section 3 computes the form factor by accumulating the contributions of a collection of line segments to the contour integral. Because the partial derivation operators are linear, they can be exchanged with this finite summation and applied to each segment of the contour. This elementary derivatives are estimated and accumulated in the total gradient. Formulas giving the partial derivative of $\overrightarrow{G}$ are given in the appendix.

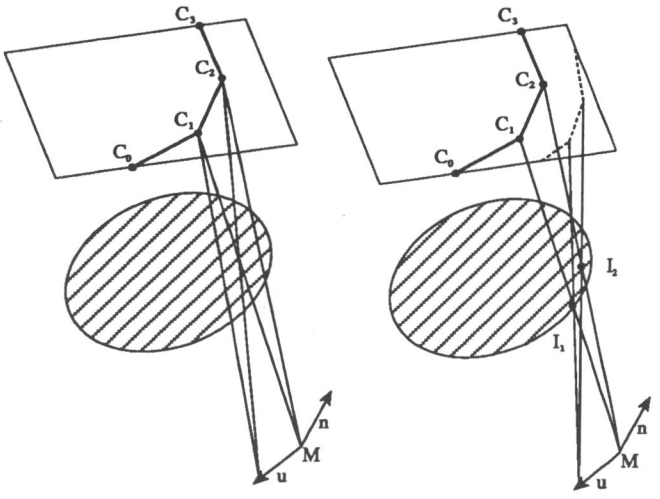

Figure 5: The 3D contour $I_0, \ldots, I_n$ must be used instead of $C_0, \ldots, C_n$ in order to get true derivatives.

For contour integration, the point of the contour on the source or on the obstacle can be used. It should be noticed that for derivation, the results will be different and that points on the obstacles should be used as Figure 5 suggests.

5 RESULTS

The algorithm used to compute images starts by casting all viewing rays for a scanline. Then the line is cut into spans of adjacent pixels covering the same object. A maximum length for each span is also set. A bounding box is then computed for each span and a candidate list of obstacles is built using the shaft culling technique [5]. If this list is empty, the span is shaded by simple integration of the contour of the source. To shade a span with non empty candidate list, the illumination for the first pixel is computed using approximate contour integration and the pattern of first level rays is recorded. For subsequent pixels, only the first level rays are sent until the pattern changes. At this point, illumination is computed and the values for the pixels in between are interpolated. This process is repeated until the end of the span.

The number of first level rays is adapted to the complexity of the obstacles. When doing contour integration, some cells can be recursively split because of insufficient precision (see Section 3.2). If the depth of recursive subdivision is two or greater or if an inconsistency is detected, the density of first level rays is increased and a finer starting grid is used. On the other hand, if no subdivision is needed and the density had been increased previously, it is decreased. If the depth of subdivision is one, the density is kept the same.

Picture	Shadow rays	Shadow rays per Viewing Ray	Av. Dif.
7	603,456,646	1024.0	-
8	5,973,391	11.4	0.23
9	6,084,145	11.6	1.31
10	18,647,345	35.6	0.64

Table 1: Statistics for the pictures. Viewing rays are those for which shadow rays were actually cast.

Experience has shown that derivatives are not precise enough at the span level to ensure a good interpolation everywhere for in between samples. They are best suited for reconstruction techniques described in [15]. Here a quadratic interpolation scheme using a third sample in the middle was applied instead. For curved surfaces, the vector $\overrightarrow{G}$ is interpolated and then projected on the true normal at each pixel.

A Monte Carlo adaptive sampling algorithm ([8] without shadow pattern coherence) was implemented for comparison. Because form factor computations are done after shaft culling, shadow rays are sent for the same pixels for both algorithms.

Images of a simple scene with a few test objects illuminated by an area light source were computed to compare the two algorithms. All images were computed with 1 viewing ray per pixel (no antialiasing) at a resolution of 800×800. To emphasize the characteristics of each image, only a detail is shown in the pictures. The reference picture was computed using traditional ray traced form factors with 1024 shadow rays per viewing ray (Figure 7). It shows very little visible aliasing in the penumbras. A picture was made using the new method with 9 initial samples and 5 dichotomy steps for each segment (Figure 8). It presents no more visible artifacts than the picture of Figure 7. Two images were made using Monte Carlo adaptive sampling. The variance threshold was adjusted so that the first image (Figure 9) has roughly the same number of rays than adaptive contour integration while the second image (Figure 10) has the same visual appearance. The average difference between each image and the reference image was also computed using the formula in [8]. Statistics are summarized in table 1.

The new algorithm is better if objects between the receiver and the source are simple. This is a property that can hold true locally even for complex scenes. However when

the complexity of the contour of the visible part of the source grows, the Monte Carlo method becomes better than approximate contour integration. In these situations, the increase of first level rays mentioned earlier is triggered. An hybrid algorithm could use this information to choose locally the best of the two methods.

6 CONCLUSION AND FUTURE DIRECTIONS

An algorithm for computing direct illumination from area sources using approximate contour integration has been presented. It exploits the fact that the silhouette of objects is smooth for the most part to greatly reduce the number of shadow rays needed to realize a given quality of shading.

The method can be applied to a large class of objects because all visibility tests are done by tracing rays. It was designed to precisely compute direct illumination in the second pass of a radiosity algorithm but can also be used to include area sources in classical ray traced scene, thus increasing realism at a lower cost than was previously possible.

While, in the current implementation, only rectangular sources are allowed, the use of any plane source is not precluded. More complicated shape can be approximated by polygonal lines and the interior of the source diced as is the case for rectangles. An extension of the method to handle triangles instead of squares is likely to be needed to make the whole process simpler.

We believe that further reductions in the number of shadow rays can be obtained by tracing the main discontinuities in the image plane. Interpolation along spans could then be replaced by 2D-interpolation. Techniques to efficiently find the discontinuities involving an edge of the source and curved obstacles are being developed.

ACKNOWLEDGMENTS

Thanks to the reviewers for their comments, to Frederic Asensio for proofreading the paper, and special thanks to Claude Puech, my thesis advisor.

APPENDIX

We define

$$\overrightarrow{G_i} = \beta_i \overrightarrow{n_i}, \tag{6}$$

and formula (4) is then

$$F_{dA_1,A_2} = dA_1 \frac{1}{2\pi} \left(\sum_{i=0}^{n-1} \overrightarrow{G_i} \right) \cdot \overrightarrow{n} \tag{7}$$

If M has coordinates (x, y, z) in the system $(\overrightarrow{I}, \overrightarrow{J}, \overrightarrow{K})$, the partial derivative of $\overrightarrow{G_i}$ with respect to x is

$$\partial_x \overrightarrow{G_i} = \partial_x \beta_i \, \overrightarrow{n_i} + \beta_i \, \partial_x \overrightarrow{n_i}, \tag{8}$$

with

$$\partial_x \beta_i = \frac{\overrightarrow{c_i} + \overrightarrow{c_{i+1}} - \left(\overrightarrow{c_i} \cdot \overrightarrow{c_{i+1}}\right) \left(\frac{\overrightarrow{c_i}}{\overrightarrow{c_i}^2} + \frac{\overrightarrow{c_{i+1}}}{\overrightarrow{c_{i+1}}^2} \right)}{\sqrt{\sqrt{\overrightarrow{c_i}^2}\sqrt{\overrightarrow{c_{i+1}}^2} - \overrightarrow{c_i} \cdot \overrightarrow{c_{i+1}}}} \cdot \overrightarrow{I}, \tag{9}$$

and

$$\partial_x \overrightarrow{n_i} = \frac{\partial_x \overrightarrow{w_i} - \left(\overrightarrow{n_i} \cdot \partial_x \overrightarrow{w_i}\right) \overrightarrow{n_i}}{\|\overrightarrow{w_i}\|}, \tag{10}$$

where $\overrightarrow{c_i} = \overrightarrow{MC_i}$, $\overrightarrow{w_i} = \overrightarrow{c_i} \times \overrightarrow{c_{i+1}}$ and $\partial_x \overrightarrow{c_i} = \overrightarrow{C_i C_{i+1}} \times \overrightarrow{I}$.

REFERENCES

[1] Baum, D. R., Mann, S., Smith, K. P., and Winget, J. M. Making radiosity usable: Automatic preprocessing and meshing techniques for the generation of accurate radiosity solutions. *Computer Graphics (SIGGRAPH '91 Proceedings) 25*, 4 (July 1991), 51--60.

[2] Campbell, III, A. T. *Modeling Global Diffuse Illumination for Image Synthesis*. PhD thesis, Dept. of Computer Sciences, Univ. of Texas at Austin, December 1991.

[3] Chin, N., and Feiner, S. Fast object-precision shadow generation for area light sources using bsp trees. In *Computer Graphics (1992 Symposium on Interactive 3D Graphics)* (March 1992), pp. 21--30.

[4] Cohen, M., Greenberg, D. P., Immel, D. S., and Brock, P. J. An efficient radiosity approach for realistic image synthesis. *IEEE Computer Graphics and Applications 6*, 3 (March 1986), 26--35.

[5] Haines, E. A., and Wallace, J. R. Shaft culling for efficient ray-traced radiosity. *Second Eurographics Workshop on Rendering* (May 1991).

[6] Hanrahan, P., Salzman, D., and Aupperle, L. A rapid hierarchical radiosity algorithm. *Computer Graphics (SIGGRAPH '91 Proceedings) 25*, 4 (July 1991), 197--206.

[7] Heckbert, P. *Simulating Global Illumination Using Adaptive Meshing*. PhD thesis, CS Division (EECS), Univ. of California, Berkeley, June 1991.

[8] Kok, A., and Jansen, F. Adaptive sampling of area light sources in ray tracing including diffuse interreflection. *Computer Graphics Forum (Eurographics '92 Proceedings) 2*, 3 (Sept. 1992), 289--298.

[9] Kok, A. J., and Jansen, F. Source selection for the direct lighting computation in global illumination. In *Second Eurographics Workshop on Rendering* (Barcelona, Spain, May 1991).

[10] Lichinsky, D., Tampieri, F., and Greenberg, D. P. Discontinuity meshing for accurate radiosity. *IEEE Computer Graphics and Applications 12*, 6 (November 1992), 25--39.

[11] Lorensen, W., and Cline, H. Marching cubes: a high resolution 3d surface reconstruction algorithm. *Computer Graphics (SIGGRAPH '87 Proceedings) 21*, 4 (Jul. 1987), 163--169.

[12] Nishita, T., and Nakamae, E. Continuous tone representation of three-dimensional objects taking account of shadows and interreflection. *Computer Graphics (SIGGRAPH '85 Proceedings) 19*, 3 (July 1985), 23--30.

[13] Shirley, P., and Wang, C. Direct lighting calculation by monte carlo integration. In *Second Eurographics Workshop on Rendering* (Barcelona, Spain, May 1991).

[14] Smits, B. E., Arvo, J. R., and Salesin, D. H. An importance-driven radiosity algorithm. *Computer Graphics (SIGGRAPH '92 Proceedings) 26*, 4 (July 1992), 273--282.

[15] Vedel, C. Improved storage and reconstruction of light intensities on surfaces. In *Third Eurographics Workshop on Rendering* (Bristol, England, May 1992).

[16] Wallace, J. R., Elmquist, K. A., and Haines, E. A. A ray tracing algorithm for progressive radiosity. *Computer Graphics (SIGGRAPH '89 Proceedings) 23*, 3 (July 1989), 315--24.

[17] Ward, G., and Heckbert, P. Irradiance gradients. In *Third Eurographics Workshop on Rendering* (Bristol, England, May 1992).

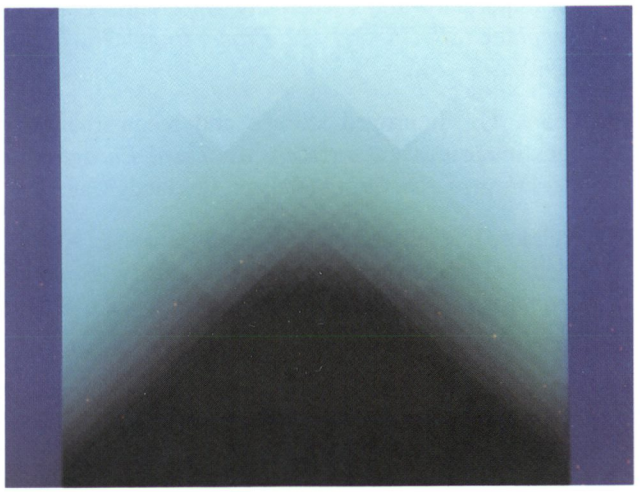

Figure 6: Closeup on the checkerboard like aliasing when the contour is not refined.

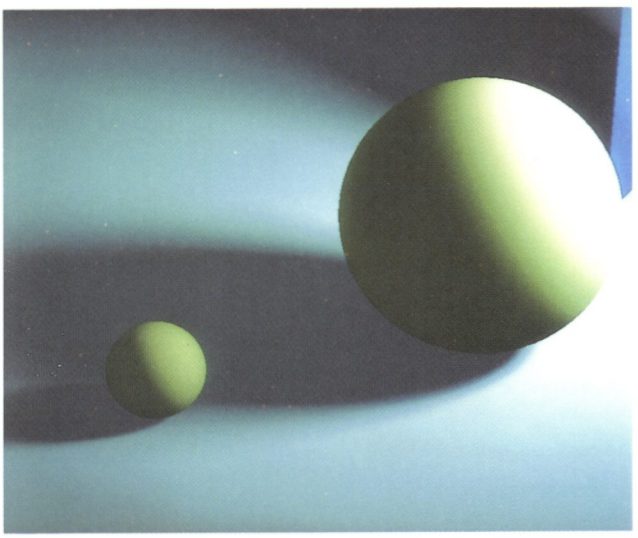

Figure 7: The reference picture computed with 1024 shadow rays per viewing ray.

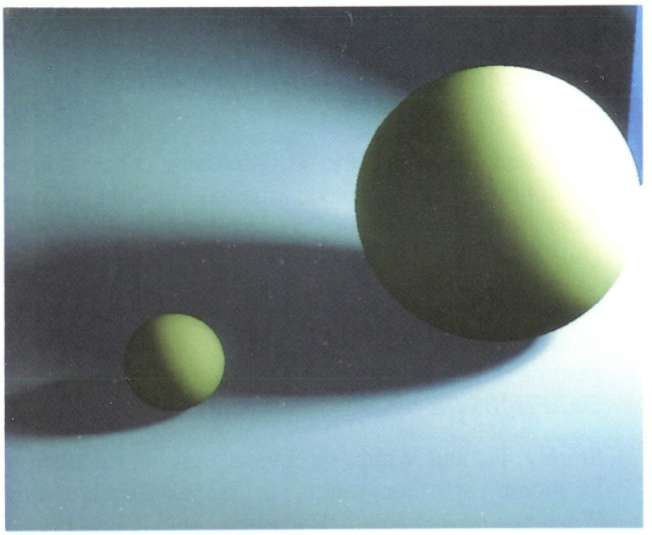

Figure 8: The picture computed with approximate contour integration at 11.4 shadow rays per viewing ray.

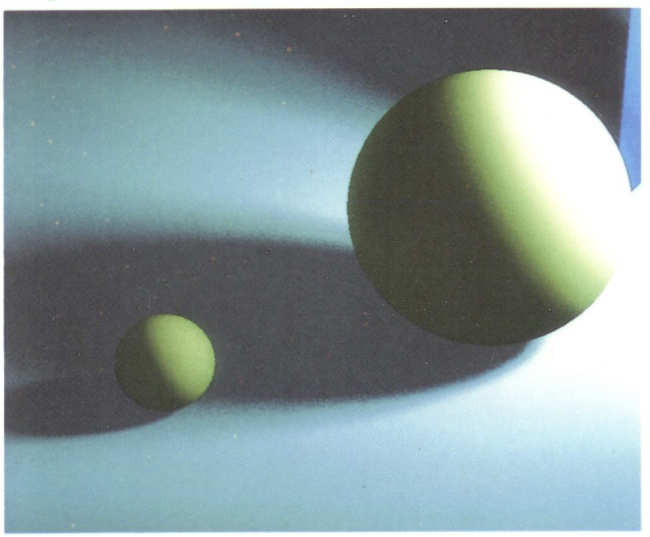

Figure 9: The picture computed with Monte Carlo adaptive sampling at 11.6 shadow rays per viewing ray.

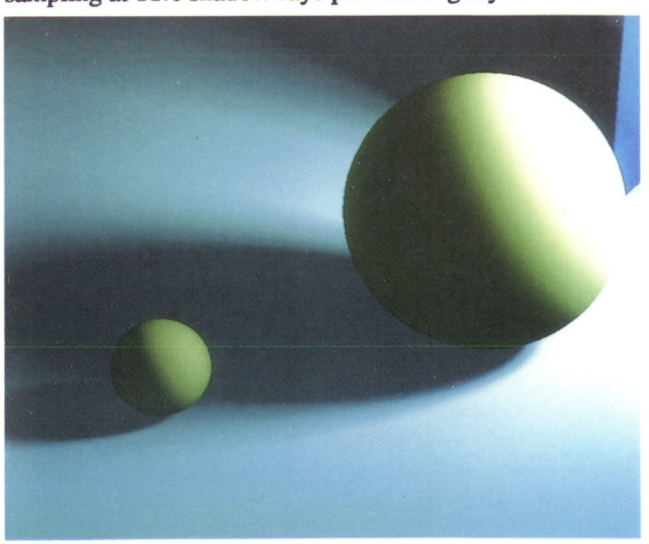

Figure 10: The picture computed with Monte Carlo adaptive sampling at 37.5 shadow rays per viewing ray.