# Accurate Direct Illumination Using Iterative Adaptive Sampling

Michael Donikian, Bruce Walter, Kavita Bala, *Member*, *IEEE*,
Sebastian Fernandez, and Donald P. Greenberg, *Member*, *IEEE*

**Abstract**—This paper introduces a new multipass algorithm for efficiently computing direct illumination in scenes with many lights and complex occlusion. Images are first divided into $8 \times 8$ pixel blocks and for each point to be shaded within a block, a probability density function (PDF) is constructed over the lights and sampled to estimate illumination using a small number of shadow rays. Information from these samples is then aggregated at both the pixel and block level and used to optimize the PDFs for the next pass. Over multiple passes the PDFs and pixel estimates are updated until convergence. Using aggregation and feedback progressively improves the sampling and automatically exploits both visibility and spatial coherence. We also use novel extensions for efficient antialiasing. Our adaptive multipass approach computes accurate direct illumination eight times faster than prior approaches in tests on several complex scenes.

**Index Terms**—Raytracing, Monte Carlo, shadowing.

## 1 Introduction

Scenes in the real-world often contain many area lights. In computer graphics though, rendering shadows from multiple area lights can be very expensive. This cost is dominated by determining visibility which is expensive to compute and difficult to predict. In this paper, we present a new direct illumination algorithm based on the following observation. Even if nearly all lights in a scene make contributions somewhere in the image, typically, only a small subset of the lights contribute significantly to the illumination at any particular point. We use an iterative multipass algorithm for rendering direct illumination in complex scenes that exhibit these properties. We support area lights, point lights, and high dynamic range environment maps as light sources.

We use probability density functions (PDFs) to randomly choose lights to evaluate using shadow rays. These are divided into a uniform PDF and two adaptive PDFs based on feedback data at both the $8 \times 8$ image block level and the pixel level. The adaptive PDFs are modified between each pass, using all sample information from previous passes to refine our sampling. A weighted combination of these PDFs uses mostly block information to guide sampling in early passes. As more samples are collected, the pixel PDFs become increasingly reliable and used for adaptive sampling.

Creating the image block-by-block gives our algorithm a compact memory footprint and allows easy parallel processing. Another important aspect for efficiency is the use of spatially clustered lights to aggregate visibility information over groups of lights. This grouping also aids in constructing and sampling accurate PDFs.

Our system uses coherence in image space to reduce the rendering time for scenes with many area lights and complex occlusion. We achieve speedups of roughly 8x for scenes with 131 thousand to 1.5 million polygons and 72-832 lights and illumination from environment maps.

Section 2 summarizes previous work in many light direct illumination. Section 3 is an overview of our iterative adaptive sampling. Section 4 describes the algorithm in detail and extensions for antialiasing. Section 5 presents results and we conclude in Section 6.

## 2 Previous Work

Direct illumination can often be expensive to compute, particularly when multiple lights or area lights and complex occlusion are involved. There have been a variety of techniques proposed to reduce this cost. Covering all this literature is beyond the scope of this paper. Instead, we will focus on the subset of direct illumination research that solves the problem through sampling.

Ward [1] dealt with the problem of many lights by sorting them by their potential contribution at each point. Visibility to the lights was evaluated in descending order until the total potential contribution of the remainder fell below some threshold. This avoids some visibility checks, but does not work well when the brightest lights are occluded or deal explicitly with area lights.

Shirley et al. [2] divided the scene into cells and, for each cell, classified the lights into important and unimportant lists based on some threshold level of illumination. The

- *M. Donikian is with the 53d Test Support Squadron, 1279 Florida Ave., Suite 221, Tyndall Air Force Base, FL 32403. E-mail: mike@graphics.cornell.edu.*
- *B. Walter is with the Cornell Program of Computer Graphics, Cornell University, 588 Rhodes Hall, Ithaca, NY 14853. E-mail: bjw@graphics.cornell.edu.*
- *K. Bala is with the Computer Science Department and the Program of Computer Graphics, Cornell University, 5142 Upson Hall, Ithaca, NY 14853. E-mail: kb@cs.cornell.edu.*
- *S. Fernandez is with Sportvision Inc., 1240 La Avenida, Mountain View, CA 94043. E-mail: sebastian.fernandez@gmail.com.*
- *D.P. Greenberg is with the Program of Computer Graphics, Cornell University, 580 Rhodes Hall, Ithaca, NY 14853. E-mail: dpg@graphics.cornell.edu.*

*Manuscript received 15 Dec. 2004; revised 31 Oct. 2005; accepted 1 Nov. 2005; published online 10 Mar. 2006.*
*For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-0255-1204.*

important lights were sampled densely while the unimportant lights were sampled sparsely. Subsampling the unimportant lights may still cause visible noise as discussed in [3]. Fernandez et al. [4] extended this approach and generated a list of occluding geometry at each geometric cell. This avoids traversing an acceleration structure for each shadow ray. Such view-independent approaches can be reused for multiple viewpoints, but use more storage and have more problems with nondiffuse surfaces than view-dependent techniques.

Paquette et al. [5] presented a light hierarchy approach to rendering scenes with many lights. Light sources were clustered into a light hierarchy and different levels of the hierarchy were used depending on the point being rendered. This allowed them to render scenes with thousands of lights. However since they do not include shadowing, it has limited applicability.

Hart et al. [6] used an image-plane-based flood-fill to propagate blocker information from pixel to pixel. With a list of the blockers affecting each light at each pixel, and asssuming polygonal geometry, they analytically computed soft shadows. Although the soft shadow quality is very good, the approach can be very expensive when there are many highly tessellated blockers involved. A similar blocker propagation approach has also been extended to handle presampled environment maps [7].

Zaninetti et al. [8] adaptively subdivided area lights, analytically computing unoccluded regions and using point sampling of visibility for partially visible regions. The illumination was sparsely sampled and interpolated for each visible light. However, interpolation may introduce error by missing small features.

Kok and Jansen [9] and Scheel et al. [10], [11] showed how to accelerate the gathering phase of a radiosity algorithm by detecting which lights had to be explicitly sampled and which could be interpolated. This can substantially reduce the computational cost of direct lighting in a radiosity setting. However, these algorithms do not fully address the problem of how to efficiently sample those lights whose illumination cannot be interpolated.

Wald et al. [12] constructed a probability density function (PDF) of the lights sources for the current image using small number of random paths. This PDF is used to sample the lights over the entire image. This approach assumes that only small number of lights will contribute to the illumination over an entire image.

Both Kollig and Keller [13] and Agarwal et al. [14] demonstrated techniques for efficiently sampling environment maps to model distant lighting. However, they do not deal with how to efficiently handle such illumination when there may be significant occlusion (e.g., Fig. 8).

General Monte Carlo optimizations can also be used. Stratified or quasirandom sequences (e.g., [15]) improve the theoretical convergence rate, though in practice, are less effective for complex integrands [16]. Our approach is closely related to the VEGAS Monte Carlo method [17], [18] which iteratively adapts a generic PDF using feedback. However, we have added many domain specific improvements including light clustering, block PDFs, partial visibility, and antialiasing optimizations.

Monte Carlo noise can be reduced by applying post-rendering smoothing operators (e.g., [19], [20], [21]) at the cost of introducing bias (e.g., potentially blurring real features). The only bias in our method is possible termination bias [22] (too few samples taken in some cases). Standard fixes can eliminate termination bias with some extra cost. We did not implement them because our tests did not reveal any significant termination bias effects.

The concurrently developed lightcuts method [23] also builds a light cluster tree, but uses deterministic adaptive sampling based on a perceptual metric and analytic upper bounds with conservative visibility assumptions. Consequently, its cost increases with the degree of light source occlusion. Our Monte Carlo method with iteratively adaptive PDFs detects and exploits such occlusion to outperform lightcuts for scenes with high light source occlusion, such as our Ponderosa example.

## 3 Overview

Computing the direct illumination at a point involves integrating the contributions from all the lights:

$$L(\vec{\mathbf{x}}) = \int_S V(\vec{\mathbf{y}})\, f_r(\vec{\mathbf{y}})\, L_e(\vec{\mathbf{y}})\, \frac{\cos\alpha\, \cos\beta}{|\vec{\mathbf{x}} - \vec{\mathbf{y}}|^2}\, d\vec{\mathbf{y}}, \qquad (1)$$

where $L(\vec{\mathbf{x}})$ is the exitant radiance reflected from a point $\vec{\mathbf{x}}$ towards the eye due to direct illumination and $S$ is the set of light sources. We denote a point on a light source as $\vec{\mathbf{y}}$ with $V()$ being the visibility of $\vec{\mathbf{y}}$ from $\vec{\mathbf{x}}$, $f_r()$ is the BRDF (Bidirectional Reflectance Distribution Function) at point $\vec{\mathbf{x}}$ evaluated for the viewing and light direction, and $L_e()$ is the emitted radiance from $\vec{\mathbf{y}}$ in the direction of $\vec{\mathbf{x}}$. The last term is purely geometric and depends on the distance between $\vec{\mathbf{x}}$ and $\vec{\mathbf{y}}$ and the angles with the surface normals at $\vec{\mathbf{x}}$ and $\vec{\mathbf{y}}$.

In simple cases, (1) can be solved exactly. However, as the complexity of the scene, lights, and materials increases, the cost of an exact solution rapidly becomes prohibitive. The standard solution is to use Monte Carlo estimation of the direct illumination, by generating $N$ samples $\{\vec{\mathbf{y}}_1, \vec{\mathbf{y}}_2, \ldots \vec{\mathbf{y}}_N\}$ on the lights according to a probability density function (PDF) $p()$ to get:

$$L(\vec{\mathbf{x}}) \approx \frac{1}{N} \sum_{i=1}^{N} \frac{V(\vec{\mathbf{y}}_i)\, f_r(\vec{\mathbf{y}}_i)\, L_e(\vec{\mathbf{y}}_i)\, \frac{\cos\alpha_i\, \cos\beta_i}{|\vec{\mathbf{x}} - \vec{\mathbf{y}}_i|^2}}{p(\vec{\mathbf{y}}_i)}. \qquad (2)$$

The noise in this estimator and, hence, the number of samples needed for a sufficiently good estimate is strongly dependent on the probability function $p()$.

### 3.1 Choosing Probability Density Functions

The ideal probability function would be zero on nonvisible samples to lights $\vec{\mathbf{y}}_i$ and otherwise exactly proportional to the numerator in (2). In this case, the terms inside the sum reduce to a constant, and the estimate is exact even if only one sample is taken. Unfortunately, computing the ideal probability function is only achievable and cost effective in the simplest cases. In practice, an approximation to the ideal probability is used. The farther the actual probability is from the ideal probability, the more variance or noise will

be in the estimator, and the longer it will take for the results to converge.

Of all the terms, the visibility term is the most difficult to predict exactly. A typical technique is to sample light sources according to an estimate of their maximum possible contribution (assuming full visibility). But, this oversamples light sources that are occluded. Moreover, even when excluding visibility, exact bounds can be difficult to compute a priori if the BRDF, geometry, or light's directional distribution are complex. Assuming reasonable sampling [2], visibility will typically be the major contributor of variance within any single light source and, therefore, we will take partial visibility conditions into account when constructing our PDFs.

Our goal is to start with a simple approximation to the ideal probability function, and iteratively optimize it using feedback from the lighting samples computed so far. This process will adapt the probability based on the local lighting configuration without requiring extensive precomputation or detailed knowledge about the scene. Conditions like occluded lights are automatically detected statistically and progressively exploited as their reliability increases. During early phases, feedback data is aggregated over larger image regions to generate statistically meaningful information. As more data becomes available, the adaptation shifts toward smaller regions.

To make our task a little simpler, we will only alter the probability that a particular light (or cluster) is chosen. We assume that some standard sampling technique is used to pick the point within a light once it is chosen, such as area sampling or the techniques of [2]. For simplicity we have only discussed area lights, but our system also supports omnidirectional point lights and high dynamic range environment maps as lights.

### 3.2 Iterative Adaptive Sampling

An overview of our algorithm is shown in Fig. 1. First, the image is divided into $8 \times 8$ blocks for processing, where each block is computed independently. This enables the use of aggregate information at the block level while keeping data structures small.

Each block is converted to a set of points in world space where direct illumination must be computed. In the simplest case, this involves shooting one viewing ray per pixel. If an antialiased image is desired then multiple points are generated per pixel as discussed in Section 4.4. A block contains multiple pixels and each pixel contains one or more points. A block is then computed using a variable number of passes.

For each point a probability density function is computed over the set of lights and this PDF is sampled a predetermined number of times. Section 4.3 describes how lights are clustered to increase the efficiency of this sampling. The first pass uses simple probability functions that do not use any feedback information. Subsequent passes use probability functions that blend these simple probability functions with probability functions constructed based on the results of prior samples averaged over the block and pixel (see Section 4.1).

Next, an estimate for the shading value and variance of each pixel is computed by combining the results from all the points associated with the pixel. These results are combined with the results of any prior passes as described in Section 4.2. If the combined variance for the pixel is less than a user-specified target noise threshold, then no further processing of the pixel is required. Otherwise, more samples are computed in the next pass for the points associated with this pixel.

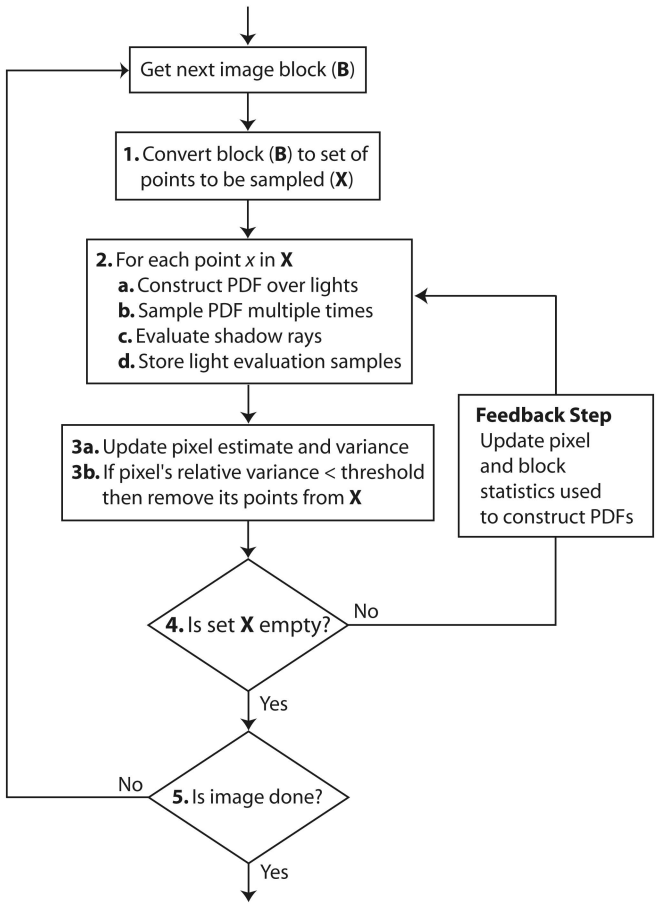



Fig. 1. Overview of iterative adaptive sampling algorithm.

The sampling results from this pass are used to update the pixel and block statistics used to compute our probability functions. This improves the sampling probabilities in subsequent passes. Once all the pixels in a block have finished, all the pixel and block data structures are cleared and we start processing the next image block until the image is finished.

## 4 Algorithm Implementation

The ideal block size is scene, image, and resolution dependent. We chose $8 \times 8$ blocks as big enough to converge faster than the pixel PDFs while small enough to remain reasonably locally adaptive.

We begin the processing of each block by finding one or more intersection points for each pixel. These points are computed by tracing rays from the camera through a pixel and finding which surface they intersect in the scene. This set of points, $X_p$, is cached for each pixel and reused for each pass. Section 4.4 discusses a novel approach toward pixel antialiasing that is particularly well-suited to this

algorithm. For each point in $X_p$ some standard geometric tests are also performed to immediately remove some lights. These include lights on the opposite side of the surface normal or oriented lights that face away from the intersection point.

### 4.1 Constructing the PDFs

For each point $x \in X_p$ we want to define a PDF specific to that point. This is done by computing PDFs for each pass (except the first) based on sample data collected from previous passes. The PDF for a point is a blending of three different functions based on sample data. The first pass has no prior results to use and its purpose is primarily to "seed" the sample data.

Later passes maintain statistics about the results of prior lighting samples in order to evolve and improve the sampling PDFs. This allows the PDFs to automatically adapt to handle conditions such as occluded lights and lights causing glossy highlights. The idea is to keep track of the average contribution and visibility of each light over the block and at each pixel, then adjust the corresponding PDFs accordingly.

We can express this process as follows. Let $R^A_{\ell,j}$ be the set of all light evaluations from points in the set $A$ to points on light $\ell$ up through pass $j$. We can think of this set as consisting of pairs of points $\{\vec{\mathbf{x}}_i, \vec{\mathbf{y}}_i\}$ that define a shadow ray (i.e., $\vec{\mathbf{y}}_i$ is a point on the light source and $\vec{\mathbf{x}}_i$ is a point being illuminated). Let $L(\vec{\mathbf{x}}_i, \vec{\mathbf{y}}_i)$ be the result of the light evaluation which is the same as evaluating (2) using just one sample. The estimated contribution of a light $\ell$ over a set $A$ for samples through pass $j$ is:

$$C^A_{\ell,j} = \frac{1}{\left|R^A_{\ell,j}\right|} \sum_{\{\vec{\mathbf{x}}_i, \vec{\mathbf{y}}_i\} \in R^A_{\ell,j}} L(\vec{\mathbf{x}}_i, \vec{\mathbf{y}}_i). \tag{3}$$

We want to assign probabilities to lights based on their contribution. Our intuition is that we need to consider partial visibility as well as radiance contribution when assigning probabilities. To find the optimum probability, we minimize for variance in terms of contribution and occlusion percentage. Let $u$ be the visible fraction of a light when viewed from the point we are trying to render. The appendix proves that the variance minimizing probability is proportional to $\frac{C}{\sqrt{u}}$. The $\sqrt{u}$ term changes the relative weighting of lights and beneficially increases the sampling of those with fractional occlusion.

Let $u^A_{\ell,j}$ be a fraction where the numerator is the number of visible light evaluations and the denominator is the total number of light evaluations sent from all surface points in the set $A$ to all sample points on the light (or cluster) $\ell$ up through pass $j$. Thus, the PDF for light $\ell$ in pass $j$ should be proportional to:

$$F^A_{\ell,j-1} = \frac{C^A_{\ell,j-1}}{\sqrt{u^A_{\ell,j-1}}}. \tag{4}$$

A PDF for light $\ell$ in pass $j$ can be constructed from the samples from all prior passes as:

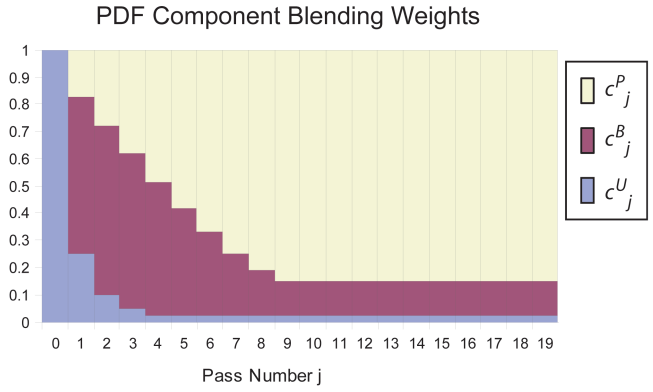



Fig. 2. The weights used to combine PDFs. $c^P_j$, $c^B_j$, and $c^U_j$ refer to the weight assigned to the pixel, block, and uniform PDF for pass $j$, respectively. Initially, we use just the uniform PDF. In later passes we weight the block PDF and the pixel PDF more heavily. Toward the end, we use the pixel PDF almost exclusively.

$$p^A_j(\ell) = \frac{F^A_{\ell,j-1}}{\sum_{k \in S} F^A_{k,j-1}}, \tag{5}$$

where $S$ is the set of lights and/or clusters for this point.

This equation is used to compute the pixel and block PDFs for a pass by setting $A$ to be the points associated with the pixel or block, respectively. Note that we do not actually need to keep all the individual light sample results, instead, we can just keep track of the running sums in (3) for the block and each pixel in it.

Performance is further improved by separating the pixel and block statistics into different sets based on the surface normal of the point being shaded. Normals are categorized into six sets using a cube decomposition of direction space aligned with the world space axes (i.e., +X, -X, +Y, -Y, +Z, -Z). When computing the pixel and block PDFs for a point only data from points with the same normal classification are used.

To construct the PDF for a point during a particular pass, we combine together three different PDFs: the two feedback PDFs $p^B(\ell)$ and $p^P(\ell)$ based on (5), and a uniform PDF $p^U(\ell)$ where all clusters have the same probability. These are combined together using the weights shown in Fig. 2 to get:

$$P_j(\ell) = c^U_j p^U(\ell) + c^B_j p^B_j(\ell) + c^P_j p^P_j(\ell). \tag{6}$$

The exact values of these weights are less important than maintaining a few important properties. The weights must sum to one. The initial pass can only use the uniform PDF because no feedback is yet available. Early passes should weigh the block PDF, $p^B$, most heavily because it is averaged over the most data and converges faster. As more data becomes available, the per-pixel PDF $p^P$ becomes more reliable and should be given larger weight, since it is more locally adaptive. The weights in Fig. 2 were determined empirically, but the algorithm is not very sensitive to their exact values.

### 4.2 Computing and Using the Pixel Estimate

The pixel result at any pass is a combination of pixel results from previous passes. Each pass computes a pixel estimate of the exitant radiance and an associated error estimate for convergence testing. This section describes how to compute these estimates across multiple passes.

The error estimate for pixel $p$ in pass $j$ is computed as the sample variance, $s^2_{p,j}$, of all the light samples for that particular pixel. This sample variance is computed numerically from these samples. Given this sample variance $s^2_{p,j}$ and the sample variance computed from the previous passes $s^2_{p,0} \cdots s^2_{p,j-1}$, the overall sample variance for the pixel, $s^2_p$, is then computed as:

$$s_p^2 = \left( \sum_{i=0}^{j} \frac{1}{s_{p,i}^2} \right)^{-1}. \quad (7)$$

The pixel value $L_p$ at the end of pass $j$, is computed using all the previous passes and their associated sample variances. Let $L_{p,i}$ be the pixel estimate from (2) using only samples from pass $i$. Our total pixel estimate is:

$$L_p = \left( \sum_{i=0}^{j} \frac{L_{p,i}}{s_{p,i}^2} \right) * s_p^2. \quad (8)$$

This combines pixel and error estimates from multiple passes such that variance is minimized [24].

Our convergence criterion tests if $s^2_p/L^2_p$ is less than a user-defined threshold $t$. If this inequality is satisfied, then that pixel is marked as completed and its final value $L_p$ is computed using (8). If after any pass, all the pixels in a block are below the threshold variance, the block is finished.

### 4.3 Light Clusters

In scenes with many lights, the number of samples required to generate good PDFs for sampling lights could be quite large. Given $N_L$ light sources in a scene, $O(N_L)$ samples are needed to build an accurate PDF. Although PDFs from sparse sampling data can be generated, the lack of data from unsampled lights is problematic. We use hierarchical light clusters to aggregate sample data over multiple light sources. These clusters exploit spatial coherence by clustering lights that are spatially near each other. Nearby lights typically have similar visibility, directions, and distance from the point they illuminate.

In each pass, samples are computed for at least a few lights in each cluster. These samples are used to estimate the contribution and hence sampling probability of the cluster as a whole. When a cluster is chosen for sampling, we randomly choose one of its constituent lights based on their relative intensities. As long as all the lights lie within a small solid angle of each other, they are likely to have similar visibility [14] and BRDF values and, thus, choosing among them solely based on intensity is a reasonable approach. In our system, we only use clusters whose bounding sphere subtends a sufficiently small solid angle from the point to be shaded.

#### 4.3.1 Constructing the Cluster Tree

Since the suitability of a cluster depends on its subtended solid angle from the point being rendered, our clustering scheme needs to be locally adaptive. Dynamically computing a new cluster partitioning for each point could be very expensive. Thus, we use a global cluster hierarchy to rapidly compute locally adaptive cluster partitions.

The cluster hierarchy is a tree where the leaves are the individual lights and the interior nodes are light clusters that contain exactly the lights below them in the tree. A greedy bottom up algorithm is used to build the cluster hierarchy by progressively pairing clusters together starting with the pair that has the smallest bounding sphere. To avoid clustering dissimilar lights together we separate out omnidirectional and oriented lights. Thus, omnidirectional lights have their own hierarchy and oriented lights are only clustered with other oriented lights if they have a similar orientation. In the end, we have: one tree for omnidirectional lights, and six trees for oriented lights—one corresponding to each of the six cardinal directions in world space (i.e., +X, -X, +Y, -Y, +Z, -Z). For static environments, the cluster hierarchy is computed only once per scene.

Since the solid angle of environment map regions does not depend on the illuminated point, they do not need a tree. We use a static partition of environment maps into 300 regions using the technique of [14]. Otherwise, these regions are treated analogously to other light clusters.

To choose the clusters to use for illuminating a point, we traverse down through the tree until we find clusters whose bounding sphere subtends a sufficiently small user-specified solid angle from the point to be shaded or until we reach the individual lights (leaves).

### 4.4 Adaptive Antialiasing

To antialias an image, we use the standard technique of supersampling by generating multiple eye ray intersections per pixel. The degree of aliasing varies inversely with the number of samples (intersection points) used.

Many adaptive antialiasing algorithms for ray tracing use adaptive progressive refinement [25] where additional eye rays are traced and shaded until pixel estimates meet a variance threshold. Since our iterative adaptive approach assumes all intersection points are known before we start a pixel, we cannot use this approach.

Instead, we would like to find a small set of representative points per pixel that still accurately represents the discontinuities present in the pixel. Our first priority is to handle the geometric boundaries since they tend to be the most visually apparent. Also, since area lights create soft shadows, shadow antialiasing is less important.

#### 4.4.1 Approach

To antialias our images, while keeping the number of intersection points low for efficiency, we propose the following solution. We initially find many intersection points for each pixel and then group similar intersection points together to create nonuniform sized subpixel regions. Each region has a representative intersection point near the center and an estimate of its subpixel area (see Fig. 3). We will only compute the illumination for these representative points. Points are grouped based on geometric similarity and distance.

We limit the size of these regions to a prespecified maximum radius (in our case, one-quarter of the pixel radius). In pixels with no geometric boundaries, this will generate at least four regions per pixel and should be sufficient for most shadow antialiasing needs. The number of regions per pixel increases with the number of geometric boundaries within the pixel.

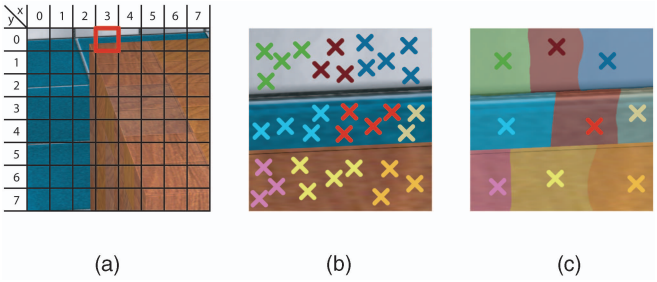



Fig. 3. Regions for pixel antialiasing. (a) An $8 \times 8$ pixel block. (b) Closeup of pixel (3,0) after shooting a set of 32 rays. The rays are grouped (and color-coded) according to our criteria: All rays of one color hit the same surface and are close to each other. (c) A false-color visualization shows regions for this pixel.

#### 4.4.2 Implementation

To find the antialiasing regions, we trace rays from the eye through each pixel. We group similar intersection points into regions according to the following criteria. A ray joins an existing region if its intersection point: 1) lies in the same plane as an existing region and has the same material (BRDF) and 2) lies within the region's radial extent. Otherwise, it starts a new region where the first intersection point becomes the representative point for that region.

The area of a subpixel region is proportional to the number of eye rays in its group. When sampling the lighting at a pixel, subpixel regions are chosen randomly according to their relative areas.

## 5 Results

In this section, we present results for our direct illumination algorithm (Iterative Adaptive Sampling) and compare it to reference solutions in speed and quality.

### 5.1 Reference Solution Implementation

To judge the effectiveness of our method, we have computed images for each of our models using a standard Monte-Carlo algorithm and quasirandom numbers.[1] The reference solutions use a fixed PDF over the light sources for each point; this PDF is proportional to the light's unoccluded irradiance at that point. We also tested a reference solution with a PDF that weights the lights sources uniformly, but found that approach was slower by about a factor of two for our scenes.

This standard approach traces eye rays through each pixel, computes a PDF for the resulting point, then samples this PDF a fixed number of times, and evaluates the resulting shadow rays to the chosen light sources. It achieves both antialiasing and noise reduction by progressively shooting more eye rays through a pixel until its estimate converges. Because we use same pixel convergence, or stopping criteria, for both the reference solution and our method, we can perform equal quality comparisons. We also perform equal time comparisons to show how well the reference solution would perform if given an equal time budget.

1. We used low discrepancy Sobol sequences, but only reset the sequence once per block to avoid the coherent aliasing artifacts often seen if every pixel uses exactly the same quasirandom numbers.

TABLE 1
Model Statistics

| Model | Triangles | Lights | | Env Map |
|---|---|---|---|---|
| | | Point | Area | |
| *Kitchen* | 338K | 0 | 72 | No |
| *Kitchen 2* | 338K | 0 | 72 | 300 Regions |
| *Ponderosa* | 131K | 138 | 0 | 300 Regions |
| *GCT* | 1527K | 613 | 219 | No |

One difficulty in implementing the reference solution is choosing the number of shadow rays to shoot per eye ray. Using too few shadow rays increases the number of PDFs that must be created, while using too many shadow rays increases costs and can cause pixel aliasing from using too few eye rays. In the end, we hand-tuned this parameter for each scene to optimize the reference solution times. Our multipass algorithm does not suffer from this problem since we compute the degree of pixel antialiasing before we begin rendering.

### 5.2 Quantitative Comparison

Our algorithm performs better than the reference solution for two main reasons. The primary performance benefit is a result of evaluating fewer shadow rays. This is possible because of our adaptive PDFs, which yield lower variance results. The secondary benefit comes from lower overhead for PDF construction. By using clusters we do not have to consider each light individually when constructing PDFs which is far more efficient in scenes with many lights.

We tested our algorithm on three different models. One of which, the *Kitchen*, has two different lighting scenarios, thus providing us with a total of four testing environments. The *Kitchen* model is our simplest scene with 72 area lights and 338,000 triangles. *Kitchen 2* is identical to the *Kitchen* except that it also contains lighting from an environment map which is coming in through windows that are behind the camera. The third model, *Ponderosa* is geometrically simple with only 131 thousand triangles, but it contains 138 point light sources as well as direct lighting from an environment map with very complicated occlusion. Our final model, *Grand Central Terminal* is a model of the Grand Concourse Lobby of the famous train station in New York City. It is our most complex model with over 1.5 million triangles and over 800 light sources, of which 219 are spherical area light sources. We have summarized the basic statistics for the four models in Table 1. Renderings of the four environments are shown in Fig. 4.

We rendered our images at $1,024 \times 1,024$ resolution with antialiasing on a dual-processor 1.7 GHz Pentium 4 Xeon computer with 1024MB of memory. Dividing the environment maps into regions and sampling for them is performed using Structured Importance Sampling [14]. For both Iterative Adaptive Sampling and the reference solution we stratify the environment map with 300 regions and use jittering and preintegration. Table 2 shows the reference image times, equal quality image times using our algorithm, and the speed ups achieved.

We also show statistics for the average number of shadow rays per pixel needed by each algorithm in Table 3.


Fig. 4. The four models analyzed for our results.

Evaluating the shadow rays is the most expensive part of computing direct illumination, and the reduction in the number of shadow rays accounts for most of the speed up achieved by our approach. We also get additional savings from our use of light clusters which have several benefits. Clusters reduce the size and cost of the PDFs which we need to create and sample, because there are typically many fewer clusters than lights, and allow us to better stratify our sampling.

The number of lights correlates only weakly with cost in our method. The degree and complexity of light source occlusion as well as how well the lights cluster has greater influence over the actual rendering cost.

We also show the average number of antialiasing regions and rendering passes per pixel in Table 4.

### 5.3 Qualitative Comparison

For the Grand Central model, we show a side-by-side equal time and equal quality comparison in Fig. 5. Due to the limited printing resolution, it is hard to notice any differences in the renderings. We provide closeup shots of the images to bring attention to quality differences and similarities. Notice that there is no perceptible difference in the equal quality comparison even in the closeup of the soft shadow region. For the remaining scenes (Fig. 6 and Fig. 7) we only show equal time image comparisons.

TABLE 3
Same Quality Light Sample Count Performance Results

| | Average Shadow Rays Per Pixel | | |
|---|---|---|---|
| *Model* | Reference | Iterative Adaptive | Reduction |
| *Kitchen* | 1,870 | 278 | **6.7x** |
| *Kitchen 2* | 9,916 | 1,604 | **6.2x** |
| *Ponderosa* | 10,772 | 1,290 | **8.3x** |
| *GCT* | 6,048 | 1,012 | **6.0x** |

### 5.4 Environment Map Techniques

Environment map lighting is often rendered by approximating it by a fixed set of point sources [14]. In scenes with significant occlusion, however, many point sources may be needed, as in Fig. 8, where even with 15,000 sources, the undersampling artifacts have not completely disappeared. Rendering a $1,024 \times 1,024$ Ponderosa image took 24,439 seconds when approximating the environment by 15,000 sources and using Ward's adaptive method [1], without antialiasing. While faster than the reference Monte Carlo solution, this is $4.3\times$ slower than our iterative adaptive solution (see Table 2). Our solution is also higher quality with antialiasing and without environment undersampling artifacts.

### 5.5 Visibility and Occlusion

Table 5 provides details on the visibility of the light sources and environment map regions in the scenes. Note that for all scenes, a large majority of the lights and environment map regions contribute somewhere in the viewpoint. On a per-pixel level, the visibility statistics are quite different. The *Ponderosa* model has the largest disparity where 100 percent of the sources are visible from some point in the image, but on average, only 4 percent are visible from any individual point. Fig. 9 is a false-color visualization of the combined per-pixel visibility of light sources and environment map regions.

### 5.6 PDF Adaptation

Fig. 10 shows how our PDF adapts to local lighting conditions at two different intersection points. One point ($x_1$) has more open visibility while the other point ($x_2$) is mostly in shadow. This environment has 832 lights, but it has been reduced to 103 clusters for point $x_1$ and 81 clusters for point $x_2$. In this figure we can see that the PDF for the first feedback-driven pass, where the block component PDF is weighted heavily, varies greatly from the initial uniform PDF. With additional passes we are able to achieve more accurate PDFs due to a combination of the greater availability of samples and the heavier weighting of the pixel PDFs which better capture local lighting configurations.

TABLE 2
Same Quality Rendering Time Performance Results

| | Rendering Time | | |
|---|---|---|---|
| *Model* | Reference (s) | Adaptive (s) | Speed-up |
| *Kitchen* | 15,364 | 1,726 | **8.9x** |
| *Kitchen 2* | 68,392 | 8,368 | **8.2x** |
| *Ponderosa* | 53,328 | 5,604 | **9.5x** |
| *GCT* | 57,432 | 7,136 | **8.0x** |

TABLE 4
Additional per Pixel Statistics

| | Per Pixel Averages | |
|---|---|---|
| Model | Anti-Aliasing Regions | Rendering Passes |
| *Kitchen* | 10.1 | 2.7 |
| *Kitchen 2* | 10.1 | 4.5 |
| *Ponderosa* | 9.8 | 3.3 |
| *GCT* | 8.8 | 5.5 |

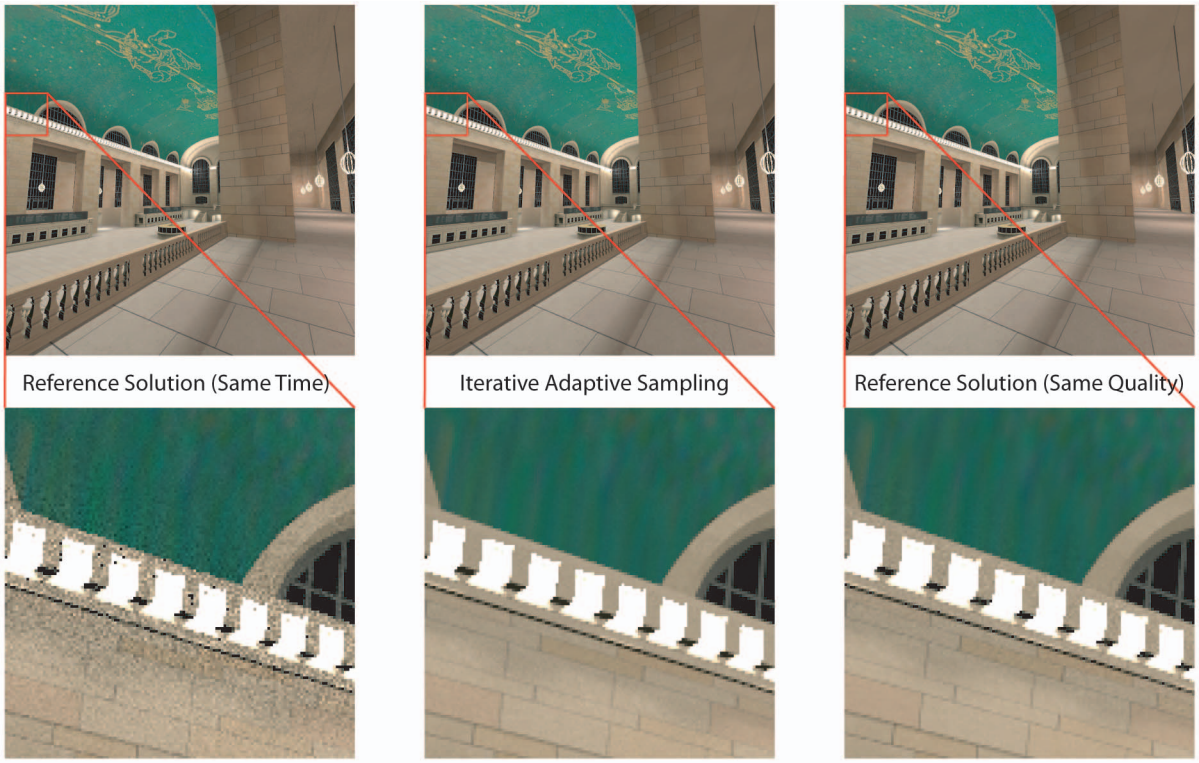



Fig. 5. Grand Central Terminal Qualitative Comparison. The bottom row contains closeups showing how the two algorithms compare when rendering soft shadows caused by many lights and many occluders.

### 5.7 Clustering

As described in Section 4.3, clustering reduces the effective number of lights we have to consider. One issue is choosing the maximum solid angle for our clusters. In [14], the authors determine that 0.01 steradians is a conservative metric for the average visibility feature size. We use a slightly more aggressive value of 0.02 steradians because we will sample each cluster multiple times. A higher cluster size maximum reduces the number of clusters, making PDF construction more efficient, but may reduce spatial coherence within clusters.


Fig. 6. Kitchen qualitative comparison. These images and closeups show how our algorithm is better able to capture glossy highlights.

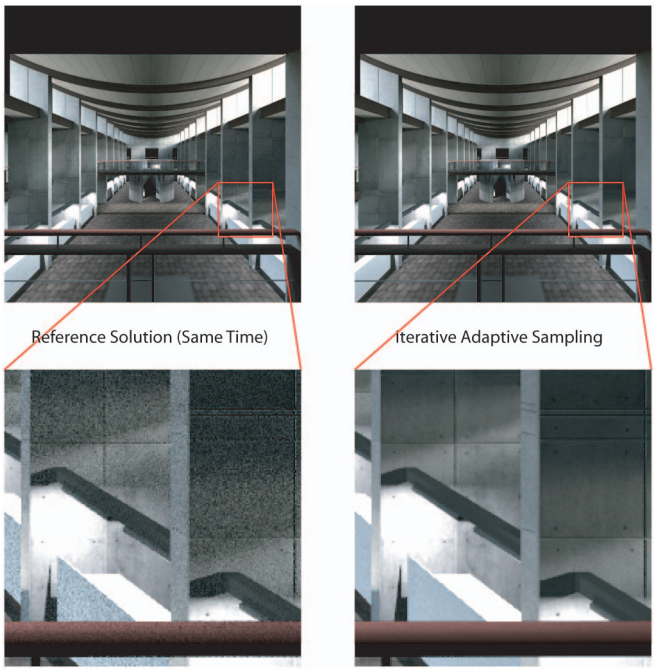



Fig. 7. Ponderosa Qualitative Comparison. In this scene, on average only 4 percent of the lights are visible at any particular point, although all the lights are visible somewhere. The closeups show our better rendering of environment lighting in scenes with complex occlusion.

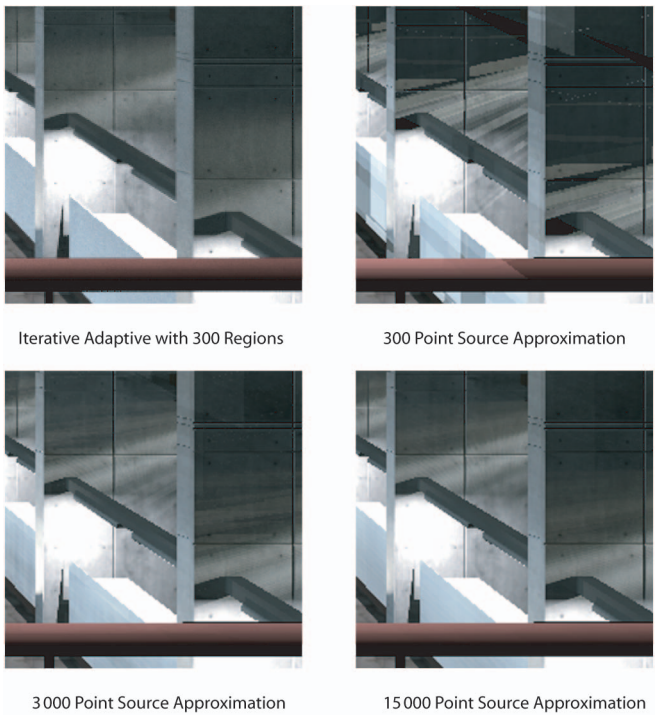



Fig. 8. Comparison with point source approximation of environment map [14]. The 300 source approximation has severe undersampling artifacts which can be reduced by using more sources. Even with 15,000 sources, some undersampling artifacts are still visible.

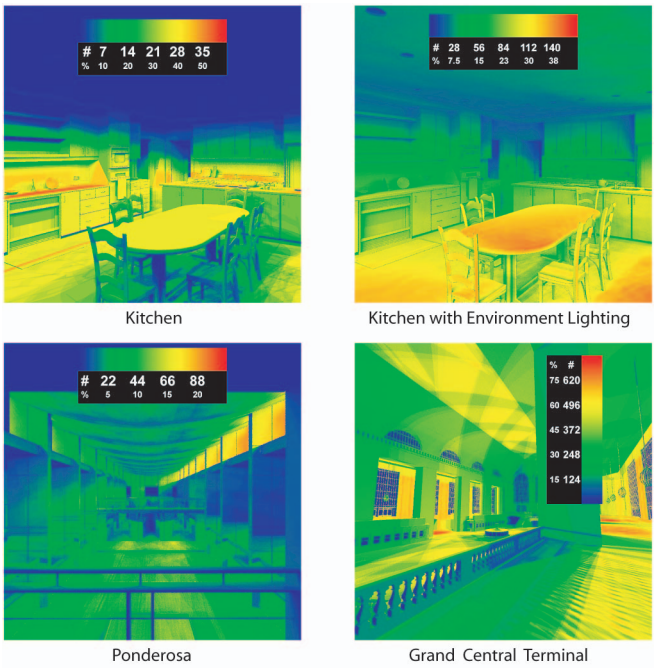



Fig. 9. Average light source visibility false color visualizations. Blue represents surfaces where a great majority of the lights are occluded while red represents surfaces that have a greater number of contributing lights.

Table 6 provides statistics on the average number of clusters per point in each of the four test environments. Remember that in our system we define a cluster as either a group of lights or an environment map region.

### 5.8 Shadow Rays per Pass

We sample the PDFs multiple times per pass based on the number of clusters, $N_C$. For all passes we sample the PDF $1.5 \times N_C$ times per pixel appropriately distributed according to the subpixel area associated with each pixel's intersection points. We choose $1.5 \times N_C$ because when combined with stratified sampling, it places at least one sample per cluster in the initial pass. We also set a minimum of 50 samples per pass to ensure reasonably good variance estimates within each pass.

As an optimization, at the end of a pass, we further sample a pixel if its variance is less than twice the desired variance. In this case, we continue sampling the pixel using the current PDFs associated with the points in increments of 25 shadow rays until we achieve our target variance. This eliminates some overhead in building PDFs and allows us to sample the pixel in smaller increments when we are close to reaching convergence.

TABLE 5
Visibility Statistics

| Model | Light Source Visibility | | |
|---|---|---|---|
| | Per Viewpoint | | Per Pixel |
| | Lights | Regions | Combined |
| *Kitchen* | 62 (86.1%) | N/A | 11.5 (16.0%) |
| *Kitchen 2* | 62 (86.1%) | 197 (65.7%) | 64.1 (17.2%) |
| *Ponderosa* | 138 (100%) | 300 (100%) | 17.5 (4.0%) |
| *GCT* | 822 (98.8%) | N/A | 279 (33.5%) |

### 5.9 Termination Condition

From Fig. 9 and Fig. 11, we can see that the most difficult pixels to render are those that have the greatest amount of occlusion. Though it may seem counter-intuitive, rendering time increases as the number of contributing light sources for each pixel decreases. The required number of light samples and shadow ray evaluations needed to reach convergence for a dark pixel can be unreasonably large, especially if the pixel is near the black point.[2] This is because our metric of relative sample variance is based on Weber's Law, which states that the amount of error the human visual system can perceive in a pixel is proportional to the base luminance of the pixel. This error tolerance is far too conservative for very dark pixels because the limiting factor is actually our display device at those low luminances.

To prevent unnecessary oversampling, we apply a maximum cutoff on the number of shadow ray evaluations for a pixel. For our algorithm, we set a maximum of 15 passes. We chose this maximum because we found that the images do not show any perceptible improvement after this. For the reference solution we set a maximum of 20,000 shadow rays. The ideal solution would be to use a more sophisticated stopping criterion.

On the other hand, it is also important not to prematurely label a pixel as black simply because all shadow rays evaluated so far were occluded. For both our algorithm and the reference solution, we require a minimum number of shadow ray evaluations before determining that a pixel is completely in shadow and therefore black to prevent black speckling in the images.

Since our adaptive algorithm aggregates sample data across multiple pixels and multiple light sources, we can more reliably determine if a pixel is indeed black. In our

2. The black point is the pixel intensity below which all values are mapped to black.

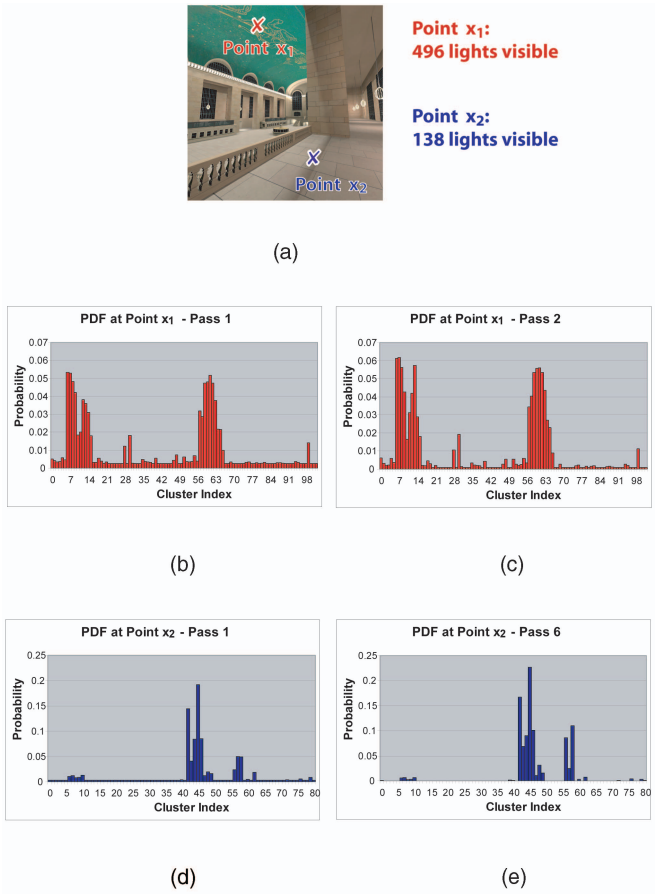



Fig. 10. (a) A rendering of Grand Central Terminal with two points highlighted. (b), (c), (d), and (e) The adaptation of PDFs for the two points. We have omitted the initial PDF (pass 0) for both passes since it is just uniform for all lights. Point $x_1$ requires two additional passes, while point $x_2$ requires six additional passes.

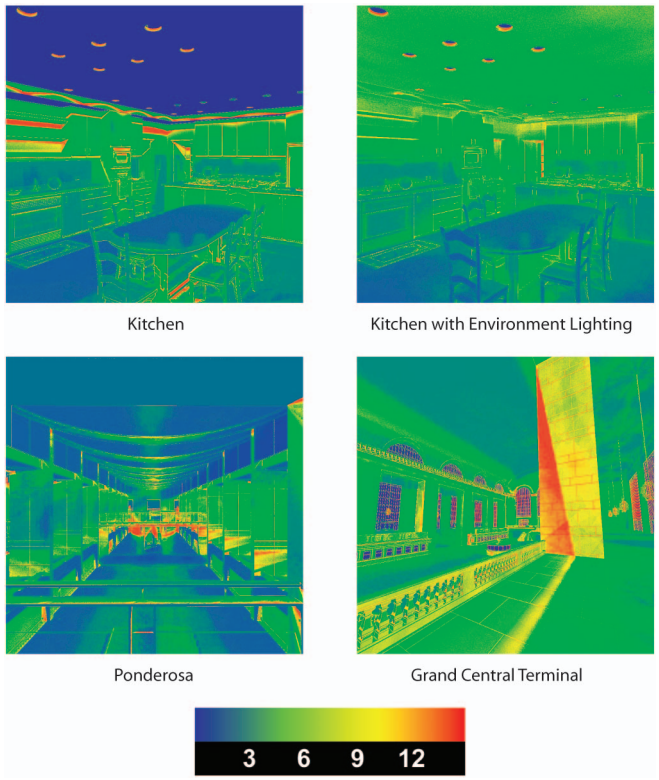



Fig. 11. Number of rendering passes performed per pixel.

algorithm, if we have not sampled a visible light after at least three passes and 250 shadow rays, we stop sampling the pixel and set it to black. This works very well in almost all situations and produced no false-positives. In both the *Grand Central* and *Ponderosa* scenes, the reference solution has difficulty in reliably detecting a fully occluded surface within a pixel unless given the much larger minimum threshold of 2,500 shadow rays. The optimum number for the reference solution is highly scene dependent, but in order to provide a valid comparison, we performed several renderings and set the value as low as possible for each scene.

## 6 Conclusions

We have presented a new iterative adaptive sampling technique for computing accurate direct illumination. It works with virtually any scene even in the presence of complicated geometry, arbitrary BRDFs, many light sources, and complex varying occlusion. Because it is based on feedback, it requires only minimal preprocessing. We have shown that our approach is roughly eight times faster than the standard Monte-Carlo approaches for several complex scenes with significant occlusion when producing images of equal quality. We have also shown how our feedback approach can be integrated with a suitable pixel antialiasing technique and how to best account for partial occlusion in sampling probabilities. We believe that our approach provides a significant advance over the state of the art for computing accurate direct illumination in scenes with complex occlusion.

TABLE 6
Clustering Statistics

| Model | Total Lights | EnvMap Regions | Avg Clusters Per Point | |
|---|---|---|---|---|
| | | | Lights | Regions |
| *Kitchen* | 72 | 0 | 24.4 (34%) | N/A |
| *Kitchen 2* | 72 | 300 | 24.4 (34%) | 175 (58%) |
| *Ponderosa* | 138 | 300 | 47.6 (35%) | 164 (55%) |
| *GCT* | 832 | 0 | 81.0 (10%) | N/A |

## Appendix

### Minimizing Cluster Variance

In this appendix, we derive how to modify our light sampling PDFs to minimize variance in cases of partial visibility. We assume that for each point being shaded the exitant radiance due to the light source is uniform across the source and that the visibility is the only source of variance. Note that our algorithm does not depend on this assumption but, in practice, visibility is typically the largest source of variance.

Consider a point illuminated by two light sources that if fully visible would contribute exitant radiances $L_1$ and $L_2$, respectively. If each light is partially visible by some amount $u_i$, then exitant radiance due to each light will be $u_1L_1$ and $u_2L_2$. We want to find the distribution of samples between the lights that minimizes variance. Let $p$ be the probability of sampling the first light and $1-p$ be the

probability of sampling the second. We derive the following cases for the contribution $M$:

- If $L_1$ is picked (probability: $p$):
  - If $L_1$ is visible (probability: $u_1$), $M = \frac{L_1}{p}$.
  - If $L_1$ is blocked (probability: $1 - u_1$), $M = 0$.
- If $L_2$ is picked (probability: $1 - p$):
  - If $L_2$ is visible (probability: $u_2$), $M = \frac{L_2}{1-p}$.
  - If $L_2$ is blocked (probability: $1 - u_2$), $M = 0$.

We can now compute the variance, using the following formula: $\sigma^2(f) = \langle f^2 \rangle - \langle f \rangle^2$. In our case:

$$\begin{aligned}\langle f \rangle &= u_1 p \times \frac{L_1}{p} + (1-u_1)p \times 0 + \\ &\quad u_2(1-p) \times \frac{L_2}{1-p} + (1-u_2)(1-p) \times 0 \\ &= u_1 L_1 + u_2 L_2 \\ \langle f^2 \rangle &= u_1 p \times \left(\frac{L_1}{p}\right)^2 + (1-u_1)p \times (0)^2 + \\ &\quad u_2(1-p) \times \left(\frac{L_2}{1-p}\right)^2 + (1-u_2)(1-p) \times (0)^2 \\ &= \frac{u_1 L_1^2}{p} + \frac{u_2 L_2^2}{1-p}.\end{aligned}$$

To determine the $p$ that minimizes variance, we differentiate $\sigma^2$ with respect to $p$ and set the result equal to zero. We find that:

$$\begin{aligned}p &= \frac{\sqrt{u_1}\, L_1}{\sqrt{u_1}\, L_1 + \sqrt{u_2}\, L_2} \\ 1 - p &= \frac{\sqrt{u_2}\, L_2}{\sqrt{u_1}\, L_1 + \sqrt{u_2}\, L_2}.\end{aligned}$$

In general, for any number of light sources the probability $p_i$ for light $i$ that minimizes the variance is: $p_i \propto \sqrt{u_i}\, L_i$. If $C$ is the contribution of a light source according to our sample data (which already contains the visibility term) and $u$ is the unoccluded fraction of the light (also according to sample data) then, $C = u\, L$ (assuming the exitant radiance is uniform across the source). Therefore, our probability of selecting a light should be proportional to $\sqrt{u}\, L = \frac{C}{\sqrt{u}}$.

## ACKNOWLEDGMENTS

Thanks for the models go to Jeremiah Fairbanks (Kitchen), Moreno Piccolotto, Yasemin Kologlu, Anne Briggs, Dana Getman (Grand Central), and Lightscape (Ponderosa). This work was supported by US National Science Foundation grant ACI-0205438 and Intel Corporation. The views expressed in this article are those of the authors and do not reflect the official policy or position of the US Air Force, Department of Defense, or the US Government.

## REFERENCES

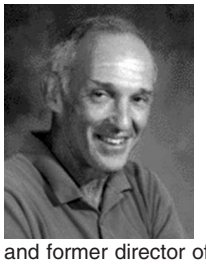
[1] G. Ward, "Adaptive Shadow Testing for Ray Tracing," *Proc. Second Eurographics Workshop Rendering,* pp. 11-20, 1994.
[2] P. Shirley, C. Wang, and K. Zimmermann, "Monte Carlo Techniques for Direct Lighting Calculations," *ACM Trans. Graphics,* vol. 15, no. 1, Jan. 1996.
[3] K. Zimmerman and P. Shirley, "A Two-Pass Realistic Image Synthesis Method for Complex Scenes," *Proc. Eurographics Rendering Workshop,* pp. 284-295, June 1995.
[4] S. Fernandez, K. Bala, and D.P. Greenberg, "Local Illumination Environments for Direct Lighting Acceleration," *Proc. 13th Eurographics Workshop Rendering,* pp. 7-14, June 2002.
[5] E. Paquette, P. Poulin, and G. Drettakis, "A Light Hierarchy for Fast Rendering of Scenes with Many Lights," *Proc. Eurographics '98,* vol. 17, no. 3, Sept. 1998.
[6] D. Hart, P. Dutré, and D. Greenberg, "Direct Illumination with Lazy Visibility Evaluation," *Computer Graphics (SIGGRAPH '99 Proc.),* pp. 147-154, Aug. 1999.
[7] A. Ben-Artzi, R. Ramamoorthi, and M. Agrawala, "Efficient Shadows from Sampled Environment Maps," Technical Report CUCS-025-04, Columbia Univ., June 2004.
[8] J. Zaninetti, P. Boy, and B. Peroche, "An Adaptive Method for Area Light Sources and Daylight in Ray Tracing," *Computer Graphics Forum,* vol. 18, no. 3, pp. 139-150, Sept. 1999.
[9] A.J.F. Kok and F.W. Jansen, "Source Selection for the Direct Lighting Computation in Global Illumination," *Photorealistic Rendering in Computer Graphics,* P. Brunet and F.W. Jansen, eds., pp. 75-82, 1994.
[10] A. Scheel, M. Stamminger, and H.-P. Seidel, "Thrifty Final Gather for Radiosity," *Rendering Techniques 2001: Proc. 12th Eurographics Workshop Rendering,* pp. 1-12, June 2001.
[11] A. Scheel, M. Stamminger, and H. Seidel, "Grid Based Final Gather for Radiosity on Complex Clustered Scenes," *Computer Graphics Forum,* vol. 21, no. 3, pp. 547-556, 2002.
[12] I. Wald, C. Benthin, and P. Slusallek, "Interactive Global Illumination in Complex and Highly Occluded Environments," *Rendering Techniques 2003: Proc. Eurographics Symp. Rendering,* pp. 74-81, June 2003.
[13] T. Kollig and A. Keller, "Efficient Illumination by High Dynamic Range Images," *Rendering Techniques 2003: Proc. Eurographics Symp. Rendering,* pp. 45-51, June 2003.
[14] S. Agarwal, R. Ramamoorthi, S. Belongie, and H.W. Jensen, "Structured Importance Sampling of Environment Maps," *ACM Trans. Graphics,* vol. 22, no. 3, pp. 605-612, July 2003.
[15] T. Kollig and A. Keller, "Efficient Multidimensional Sampling," *Computer Graphics Forum,* vol. 21, no. 3, pp. 557-564, 2002.
[16] D.P. Mitchell, "Consequences of Stratified Sampling in Graphics," *Computer Graphics (Proc. SIGGRAPH '96),* pp. 277-280, Aug. 1996.
[17] G.P. Lepage, "A New Algorithm for Adaptive Multidimensional Integration," *J. Computational Physics,* vol. 27, pp. 192-203, 1978.
[18] W. Leeson and S. Collins, "Yet Another Rendering Framework," Technical Report TCD-CS-1999-41, Trinity College, June 1999.
[19] H.E. Rushmeier and G.J. Ward, "Energy Preserving Non-Linear Filters," *Computer Graphics (Proc. SIGGRAPH '94),* pp. 131-138, July 1994.
[20] I. Wald, T. Kollig, C. Benthin, A. Keller, and P. Slusallek, "Interactive Global Illumination Using Fast Ray Tracing," *Proc. 13th Eurographics Workshop Rendering,* pp. 15-24, June 2002.
[21] R. Xu and S.N. Pattanaik, "A Novel Monte Carlo Noise Reduction Operator," *IEEE Computer Graphics and Applications,* vol. 25, no. 2, pp. 31-35, Mar./Apr. 2005.
[22] D.B. Kirk and J. Arvo, "Unbiased Sampling Techniques for Image Synthesis," *Computer Graphics (Proc. SIGGRAPH '91),* vol. 25, no. 4, pp. 153-156, July 1991.
[23] B. Walter, S. Fernandez, A. Arbree, K. Bala, M. Donikian, and D. Greenberg, "Lightcuts: A Scalable Approach to Illumination," *ACM Trans. Graphics,* vol. 24, no. 3, Aug. 2005.
[24] P. Dutre, P. Bekaert, and K. Bala, *Advanced Global Illumination.* Natick, Mass.: AK Peters, 2003.
[25] J. Painter and K. Sloan, "Antialiased Ray Tracing by Adaptive Progressive Refinement," *Computer Graphics (Proc. SIGGRAPH '89),* vol. 23, no. 3, pp. 281-288, July 1989.

**Michael Donikian** received the BS degree in computer science from Cornell University in 2002. He also received the Master's degree from the Cornell University Program of Computer Graphics under an Air Force Institute of Technology sponsorship. He currently works at Tyndall Air Force Base, Florida as a communications officer for the 53d Test Support Squadron.

**Bruce Walter** received the BA degree in computer science and physics from Williams College in 1991 and the PhD degree in computer science from Cornell University in 1998. He is a research associate in the Cornell Program of Computer Graphics. His research focus is realistic rendering in complex environments and interactive techniques. He was previously a postdoctorate with the iMagis group in Grenoble, France and the lead developer for the initial versions of the trueSpace product at Caligari Corp.

**Kavita Bala** received the BTech degree from the Indian Institute of Technology (IIT, Bombay), and the SM and PhD degrees from the Massachusetts Institute of Technology (MIT). She is an assistant professor in the Computer Science Department and Program of Computer Graphics at Cornell University. She specializes in interactive computer graphics, leading several research projects in interactive rendering, global illumination, and image-based modeling and texturing. In 2005, she cochaired the Eurographics Symposium on Rendering (EGSR); she has also served on numerous program committees including SIGGRAPH, the Point-Based Symposium, and Graphics Interface. She is a coauthor of the graduate textbook *Advanced Global Illumination*. She is a member of the IEEE and the IEEE Computer Society.

**Sebastian Fernandez** received the PhD degree in computer science from Cornell University in 2004. Previously, he received the BS degree in electrical engineering and computer science from the University of California at Berkeley in 1994. His research interests include the modeling and rendering of complex environments. He is currently working at Sportvision Inc. on real-time sports data visualization.



**Donald P. Greenberg** is the Jacob Gould Shurman Professor of Computer Graphics and the Director of the Program of Computer Graphics at Cornell University. He has been a pioneering researcher in computer graphics since 1965 and a recipient of the ACM Steven Coons Award. He was the founding director of the US National Science Foundation Science and Technology Center for Computer Graphics and Scientific Visualization and the originator and former director of the Computer Aided Design Instructional Facility at Cornell University. His specialties include real time realistic image generation, geometric modeling, and color science. He presently teaches the computer graphics courses in computer science, computer-aided design in architecture, computer animation in art, and technology strategy in the Business School. He is a member of the IEEE and the IEEE Computer Society.