Sampling Precomputed Volumetric Lighting

Janne Kontkanen[1], Samuli Laine[1,2]

[1]Helsinki University of Technology / TML, [2]Hybrid Graphics Ltd.

Abstract

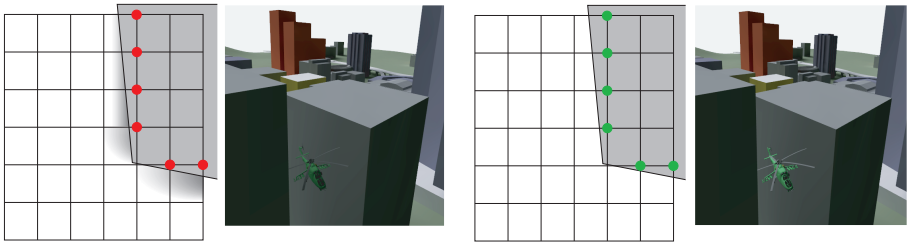
Precomputing volumetric lighting allows realistic mutual shadowing and reflections between objects with little run-time cost: for example, using an irradiance volume the shadows and reflections due to a static scene can be precomputed into a 3D grid and this grid can be used to shade moving objects at run-time. However, a rather low spatial resolution has to be used to keep the memory requirements acceptable. For this reason, these methods often suffer from aliasing artifacts.

In this article we introduce a new sampling algorithm for precomputing lighting in to a regular 3D grid. The advantage of the new method is that it dramatically reduces aliasing while adding only a small overhead for the precomputation time. Additionally, the run-time component does not have to be changed at all.

1 Introduction

With precomputed volumetric lighting, we refer to the class of techniques that precompute the illumination effects caused by a rigid object or scene to the surrounding space. Then at run-time this precomputed information that is fixed in the coordinate frame of the object or scene is used to shade other surfaces that are in the vicinity. Techniques that fall into this category are, for example, irradiance volume [2], neighborhood transfer [9] and ambient occlusion fields [4]. These are described in more detail in Section 2.

Usually these methods precompute the lighting information at the nodes of the 3D grid. Then the run-time lookup at an arbitrary location is done by fetching the values at the neighboring nodes and using tri-linear interpolation. By its nature, this approach is prone to aliasing. Most distracting artifacts tend to occur where the scene geometry intersects the volume. For example, a node inside a solid wall is not a good representative for the illumination outside the wall and due to tri-linear interpolation the darkness leaks from the wall to the open space. The irradiance volume in the Figures 1(a-b) illustrates this. Sometimes artifacts can be diminished by carefully positioning the grid, but this is of course not always possible. Since geometry discontinuities introduce infinite frequencies into the lighting, there is no finite resolution that is sufficient to capture all the detail.



a) b) c) d)

Figure 1: A helicopter is lit by an irradiance volume that captures the reflections and shadows cast by the scene. **a)** 2D illustration of the leakage problem. Darkness is leaking from a solid structure into the open space due to interpolation. The red dots represent the nodes that cause a poor estimate of lighting outside the solid. **b)** The same artifact shown in 3D scene. The helicopter is lit by a conventional irradiance volume and some of the grid nodes are inside the buildings, thus giving a bad estimate for lighting incident on the helicopter. As a result, the helicopter appears too dark. **c-d)** The approach suggested in this article solves the problem by precomputing the interior nodes (green) in such a way that they only account for illumination outside the solid. For more comprehensive results and a comparison against a reference solution, see Figure 6.

In this article, we present an algorithm that reduces aliasing in two ways. The first is fairly standard pre-filtering. This removes or attenuates the high frequencies that cannot be represented by the grid. In practice, this is done by modifying the precomputation stage so that instead of shooting rays from the nodes of the grid, we shoot them from random locations and accumulate the results of the ray queries according to the tri-linear weighting functions of the grid nodes. This alone diminishes the problem shown in Figure 1(a-b), but does *not* eliminate it.

Fortunately, the first idea prepares the way for the second. While we cannot capture the discontinuities introduced by the polygon boundaries, we can redefine the problem so that we get rid of most of these discontinuities. Note that in most applications the polygonal surface represents the boundary between solid and empty space and the solid objects are not allowed to overlap. In addition, it is common that the camera is not allowed inside the solids. If either or both of these restrictions is accepted, then it is only required to reproduce the illumination on one side of the polygonal boundaries i.e. in the empty space. In this article, this domain is called *the domain of interest*.

During precomputation, we need to generate rays that originate uniformly from the domain of interest. To do this, we propose an efficient heuristic algorithm that does not require this domain to be explicitly specified. Instead, the domain is defined as a part of the space with no visibility to back-faces of the geometry. This definition has the practical advantage that the geometry does not have to be closed. While the definition imposes some requirements for the

geometry, these are in practice manageable (see Section 7). Figure 1(d) shows the new technique in practice.

2 Background and Applicability

The irradiance volume [2] is an important technique for computer games, because it can be used for realistic scene-to-object-lighting including arbitrary reflections and shadowing. To our knowledge, the first computer game using an irradiance volume-like approach was *Max Payne 2* by Remedy Entertainment [1]. The method is especially useful for indirect illumination and large area light sources.

In the irradiance volume the irradiance distribution function (i.e. irradiance as a function of the normal of the receiving surface) is precomputed and stored into the nodes of the 3D grid. Then this grid is used to illuminate moving diffuse objects at real-time. In the original article the irradiance distribution function was stored by discretizing it with respect to direction. However, a more up-to-date version of the technique is obtained if the irradiance distribution function is expressed in the spherical harmonics basis [7]. Also, the technique can be generalized for non-diffuse reflections by storing radiance instead of irradiance. In this article, we demonstrate our method with irradiance volume using spherical harmonics as the directional basis.

In addition to irradiance/radiance volume, the new method is applicable to neighborhood transfer [9] and ambient occlusion [10, 5]. Neighborhood transfer can be considered as a radiance volume that has been parameterized by distant environment lighting. The technique is used to capture the reflections and shadows that an object or scene casts to their surroundings in the presence of dynamic lighting.

The ambient occlusion shadows cast by an object to another can be efficiently rendered in real-time if the ambient shadows are precomputed into the surrounding space of each shadow caster as done by Kontkanen and Laine [4]. The authors use heuristic radial models to store the 3D data into a 2D cube-map and thus the new sampling method is not directly applicable. However, the new method can be applied if the ambient occlusion is precomputed into a regular 3D grid.

3 Irradiance Volume with Spherical Harmonics

In this section, we briefly describe how we upgraded the classical irradiance volume to use spherical harmonics as directional basis. This is fairly straightforward, but the formalizations we develop are needed in the subsequent sections.

First, consider a single point $\mathbf{x}$ in space and the illumination incident upon it. Ramamoorthi and Hanrahan [7] showed that the irradiance depends strongly only on the first nine spherical harmonic coefficients of the radiance distribution. According to this, we store the radiance distribution with nine coefficients and

Table 1: The first nine spherical harmonics basis functions Y_j and the convolution coefficients A_j for transforming the radiance to irradiance. x, y and z are the components of the normalized direction vector. b_i is a standard tri-linear weighting function of a node i.

$Y_0 = 0.282209$	$A_0 = 3.141593$	$b_i(x, y, z)$	=	$(1 - dx)(1 - dy)(1 - dz)$
$Y_1 = 0.488603\ x$	$A_1 = 2.09395$	dx	=	$\min(\lvert x - x_i\rvert / w_x, 1)$
$Y_2 = 0.488603\ y$	$A_2 = 2.09395$	dy	=	$\min(\lvert y - y_i\rvert / w_y, 1)$
$Y_3 = 0.488603\ z$	$A_3 = 2.09395$	dz	=	$\min(\lvert z - z_i\rvert / w_z, 1)$
$Y_4 = 1.092548\ xz$	$A_4 = 0.785398$			
$Y_5 = 1.092548\ yz$	$A_5 = 0.785398$	x_i,y_i,z_i	=	location of node i
$Y_6 = 1.092548\ xy$	$A_6 = 0.785398$	w_x,w_y,w_z	=	width of the grid cell
$Y_7 = 0.315392\ (3x^2 - 1)$	$A_7 = 0.785398$			
$Y_8 = 0.546274\ (x^2 - y^2)$	$A_8 = 0.785398$			

convert it to irradiance in the run-time look-up (Section 6). This is expressed by the following equation[1]:

$$E(\mathbf{x}, \mathbf{n}) \approx \sum_{j=1}^{9} L_j(\mathbf{x}) A_j Y_j(\mathbf{n}), \tag{1}$$

where $E(\mathbf{x}, \mathbf{n})$ denotes the irradiance at location $\mathbf{x}$, given a normal vector of the receiving surface $\mathbf{n}$. L_j is the j:th spherical harmonic coefficient of the radiance distribution. A_j refers to the convolution coefficients for converting the radiance to irradiance (Table 1), and Y_j is the j:th spherical harmonic basis function (Table 1). Since we store radiance instead of irradiance, the data can be used for illuminating non-diffuse surfaces as well, but for simplicity we consider the diffuse case only. Of course, it would be equally justified to directly store the irradiance distribution functions in the spherical harmonics basis.

At run-time we need to reconstruct the irradiance according to Equation 1 at arbitrary locations in space. Thus, we need to have access to the spherical harmonic coefficients $L_j(\mathbf{x})$ at any location $\mathbf{x}$. For this, we spatially discretize the functions $L_j(\mathbf{x})$ into a 3D grid. In the look-up we express them as the weighted averages of the values stored in the neighboring nodes:

$$L_j(\mathbf{x}) \approx \sum_{i \in B(\mathbf{x})} c_{ij} b_i(\mathbf{x}), \tag{2}$$

where c_{ij} is the coefficient corresponding to the node i and the spherical harmonic j. The function b_i is the standard tri-linear weighting function of the node i (for definition see Table 1). $B(\mathbf{x})$ refers to the set of nodes i for which $b_i(\mathbf{x}) > 0$. Substituting (2) to (1) gives:

$$E(\mathbf{x}, \mathbf{n}) \approx \sum_{j=1}^{9} A_j \left(\sum_{i \in B(\mathbf{x})} c_{ij} b_i(\mathbf{x}) \right) Y_j(\mathbf{n}), \tag{3}$$

[1]Equation 1 corresponds to Equation 7 in [7]. However, for a detailed derivation, an interested reader should read both [7] and [8].

which is all that is needed at run-time when reconstructing the approximation of the irradiance. For completeness, the above reconstruction is given as pseudo-code in Section 6. In the next section we discuss how to determine the coefficients c_{ij}.

4 Sampling

In the most straightforward implementation of irradiance volume with spherical harmonics, the coefficients c_{ij} in Equation 3 would be determined by projecting the incident illumination to spherical harmonics separately at each node location $\mathbf{x}_i$:

$$c_{ij} = \int_{\Theta} L(\mathbf{x}_i, \omega) Y_j(\omega) d\omega \tag{4}$$

We refer to this as *point sampling*. However, the problem with this method is that a single location $\mathbf{x}_i$ is often a bad representative for the illumination in its neighborhood. Thus, instead of point sampling, we compute the coefficients c_{ij} by integrating the radiance against both the spherical harmonics and the tri-linear weighting functions according the following double integral:

$$c_{ij} = \frac{\int_{\Delta} \left(\int_{\Theta} L(\mathbf{x}, \omega) Y_j(\omega) d\omega \right) b_i(\mathbf{x}) d\mathbf{x}}{\int_{\Delta} b_i(\mathbf{x}) d\mathbf{x}} \tag{5}$$

where Δ is the spatial domain covering the domain of interest of the irradiance volume (see the next section) and Θ refers to the set of all possible directions. The denominator is necessary because in our method, the boundary of Δ is determined by scene geometry, and thus the basis functions b_i are not normalized in the domain[2].

In practice, we estimate Equation 5 using Monte Carlo integration by tracing random rays distributed in both spatial and angular dimensions as described in Section 6.

There are several advantages of using Equation 5 instead of Equation 4. First is anti-aliasing: in Equation 5 each c_{ij} is a weighted average of the light distribution on the support of the corresponding basis function b_i. Thus the method removes high frequencies that cannot be represented by the spatial basis. Second advantage is that Equation 5 gives fine control over the integration domain Δ. We use this opportunity to carefully define the domain to address the problem of leaking illumination.

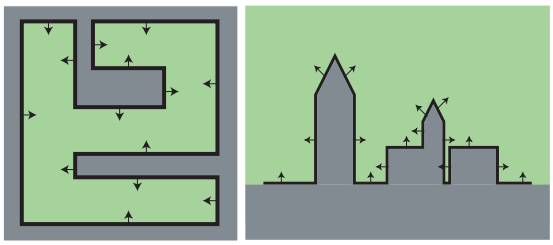


Figure 2: 2D illustrations of the domain of interest in different scenes. The small arrows denote the normal vectors of the scene geometry, black contour represents the surfaces and the domain of interest is light green. The space that is not a part of the domain is draw with dark gray. **Left**: A horizontal cross section of a maze. **Right**: A vertical cross section of a small town, intended to be viewed only from outside.

5 Domain of Interest

The domain of interest, denoted Δ, must contain all the space in which the incident lighting is queried at run-time. In practice, this means every part of space that visible dynamic objects are allowed to enter. In a typical application at least solid structures are outside of this domain.

On the other hand, efficient Monte Carlo integration of Equation 5 relies on the ability to quickly generate samples from the domain of interest Δ. Since we will use rejection sampling this means that we need to be able to quickly tell whether an arbitrary point belongs into the domain of interest or not. We refer to this procedure as *point classification*.

We define the domain of interest as *all the space from which no back-face of the scene geometry is visible*. 2D illustrations of this are shown in Figure 2. This definition excludes the solid structures from the domain of interest, and has an advantage that the point classification is relatively easy, as explained in Section 6.

The requirement that the back-faces of the scene geometry are not exposed to the viewer is reasonable for most applications. For example, it does not imply that the geometry must be closed, watertight or non-self-intersecting. In Section 7 we give a detailed error analysis and show that the classification is robust in practice even in the presence of different kinds of geometry imperfections such as cracks.

[2]Note also that Equation 5 is not a *projection* to the spatial basis $b_i(\mathbf{x})$, but simply an average weighted according to $b_i(\mathbf{x})$. The basis $b_i(\mathbf{x})$ is not orthogonal, and evaluating the minimum error projection would require integrating against dual basis functions with infinite support. This is not feasible in practice, and the result might be visually objectionable due to ringing artifacts.

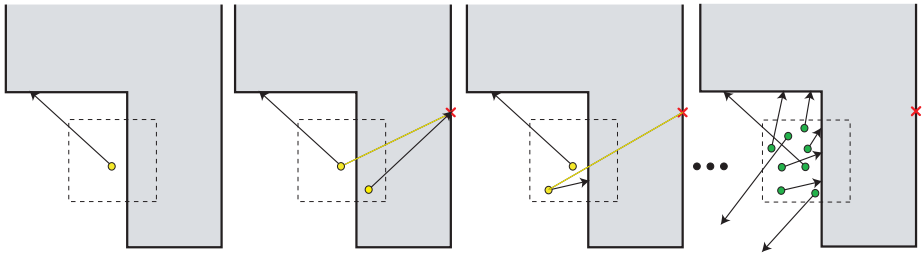


Figure 3: Algorithm for point classification. In **(a)**, first potential sampling ray origin (yellow) is created and in absence of known back-face-points a ray is shot towards random direction. This yields an intersection with front-face and for now, the sampling ray is preserved. In **(b)**, second sampling ray is generated and it hits a back-face. Thus the sampling ray is discarded and a back-face-point (red cross) is created. Previously generated sampling rays are tested against this new back-face-point by shooting query rays (yellow) towards it. In this case, the query ray does not hit a back-face, thus the existing sampling ray is preserved. In **(c)**, third sampling ray origin is created, and tested against known back-face-points. Since this does not yield intersections with a back-face, a sampling ray is shot towards a random direction. In **(d)**, the process has been completed and only the rays originating from the domain of interest remain (green).

6 Implementation

In this section we describe both the precomputation and run-time stages of our method. However, most of the section is devoted to the precomputation, since the run-time component is not affected by the new sampling method.

Precomputation

The double integral in Equation 5 can be estimated by Monte Carlo integration:

$$c_{ij} \approx 4\pi \frac{\sum_{s=1}^{N} L(\mathbf{x}_s, \omega_s) Y_j(\omega_s) b_i(\mathbf{x}_s)}{\sum_{s=1}^{N} b_i(\mathbf{x}_s)} \tag{6}$$

In the above, the nominator and denominator of Equation 5 have been approximated separately with the same set of samples $\mathbf{x}_s$ and the terms $1/N$ cancel out. In practice, the evaluation of this equation is done by shooting randomly oriented rays from random locations inside the domain of interest. For fast convergence, we used Halton quasirandom sequence [3] to generate both the directional samples ω_s and spatial samples $\mathbf{x}_s$.

Since the domain of interest Δ is defined implicitly by the scene geometry as explained in Section 5, we can not directly generate samples (ray origins) from it. Instead, we use rejection sampling, i.e., we generate uniformly distributed samples in a simpler bounding volume and reject those that are not within the

desired domain. To do this, we need a method to tell whether an arbitrary point lies within the domain of interest or not.

Recall from Section 5 that if back-facing geometry can be seen from location $\mathbf{x}$, it does not belong into the domain of interest and thus a ray that originates from such a location should be ignored. A naïve algorithm would check each candidate ray by shooting numerous query rays from its origin to all directions and test whether any of them hit back-facing geometry. However, this is not affordable in practice, and we developed a more efficient algorithm by utilizing the spatial coherence.

We divide the space into cells and process one cell at the time. For convenience, we chose to use the irradiance volume cells. The goal of the algorithm is to produce a set of randomly oriented rays so that the origins are inside the part of the cell that overlaps the domain of interest.

The algorithm traces two kinds of rays. *Sampling rays* are shot to random directions and are, in the end, used to integrate the incident illumination into the cell. *Query rays* are shot to explicitly test whether there is visibility from a potential sampling ray origin to back-facing geometry. *Back-face-point* refers to a location on a back-facing surface that is known to be visible from the cell.

Processing a single cell starts by clearing the list of known back-face-points and the list of sampling rays. Then *potential sampling ray origins* are repeatedly generated. Whenever visibility from a potential sampling ray origin to back-face-point is detected, the sampling ray is thrown away and all the sampling ray origins generated so far are tested by shooting query rays towards this new back-face-point. In the end only rays that have origins in the domain of interest remain. The full description of the procedure is given in Figure 3.

The results of the ray intersection computations done for the point classification algorithm can be re-used when computing the incident illumination. In cells with no visibility to back-facing geometry, the algorithm does not cause any notable overhead: Only the rays that are required to integrate the illumination are traced. On the other hand, in the rest of the cells the algorithm often needs only one back-face-point. A complete pseudo-code for doing this is given in Algorithm 2.

Run-time

Since the run-time component is not affected by the new sampling scheme for precomputation, it can be straightforwardly implemented using graphics hardware. The look-up is done according to Equation 3 and the resulting pseudo-code is given in Algorithm 1.

7 Error Analysis

In this section, we focus on the possible sources of error in our method. For the purposes of discussion, we divide the errors in three classes according to their

Algorithm 1 Pseudo-code for the run-time look-up.

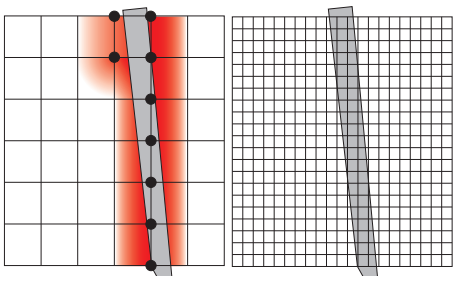
input: location x, surface normal n
  $cell$ = getCell(x)
  for all corners of cell, *node* **do**
    $i \leftarrow$ index of *node*
    $L = 0$
    for $j = 1$ to 9 **do**
      $L = L + A_j Y_j(n) b_i(x) c(i, j)$
  return L

cause: errors due to lack of resolution, errors due to geometry imperfections, and errors due to small support.

Lack of Resolution: It may happen that the support of a single node covers an area with abrupt changes in lighting. In this case the node may be unable to represent the lighting accurately. For example, a low-resolution grid is unable to capture the illumination on both sides of a thin wall (see Figure 4(a)).

In some cases this problem can be alleviated by aligning the grid suitably with the geometry, but this is not in general the case. The only way to actually solve this undersampling problem is to increase the resolution of the grid (see Figure 4(b)). For a cost-efficient way to choose the resolution, an adaptive scheme should be used [6]. These methods are still under development, but we expect them to be orthogonal to our technique.

For practical irradiance volumes, undersampling the lighting is a necessity. For example, see the interior scene in Figure 6. The thin bars of the bed cast small shadows as seen in the reference image. A faithful reproduction of these shadows would require an excessive resolution. Our method averages the shadows away, while the the conventional point sampling causes a distracting artifact as some of the grid nodes end up inside the bars.



(a) (b)

Figure 4: The effect of a coarse resolution. In **(a)** a too low resolution is used and the support (red) of some of the nodes (marked with black dots) reach to the both sides of the wall. There is no way to precompute these nodes so that potentially very different illumination at the both sides of the wall can be reproduced. In **(b)** high enough resolution is used and none of the nodes have support on the both sides of the wall.

Lack of Support: It is possible that only a small part of the support of a certain spatial basis function falls into the domain of interest. In this case only a small number of sampling rays is used to evaluate the illumination for the corresponding node. Intuitively, this should cause high variance. However, the basis functions that are mostly outside the domain of interest tend to have small weight also when the lighting is reconstructed. Thus, we have not seen the effects of this potential problem in practice.

Geometry Imperfections: The requirement that the back-faces of the geometry should not be visible from the locations where the lighting volume is used is in general reasonable. However, in real applications the geometry is often imperfect, in which case the back-faces may become visible from the domain of interest.

Common imperfections are polygons with reversed surface normals and cracks between polygons. Our method does not tolerate the former, but fortunately the reversed polygons can be easily found by visual inspection if the back-faces are highlighted with a color. Then the back-facing polygons can be reversed manually. The latter, however, requires further attention.

In the presence of a crack (see Figure 5), a sampling ray shot from a cell may accidentally go through the crack and thus generate a back-face-point. This causes the ray to be thrown away and the rest of the sampling locations are then tested against this new back-face-point. Fortunately, since the further query rays originate from different locations, it is very improbable that any of them will go through the crack again. Thus the number of misclassified samples is statistically insignificant and the algorithm is robust in practice even in the presence of cracks.

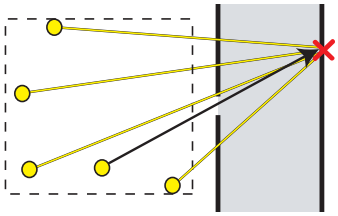


Figure 5: Cracks between polygons might result in rays going through surfaces. This may give false visibility from the domain of interest to back-faces. However, even if a back-face-point is found this way, it is very improbable that the query rays shot to classify the other points in the cluster go through the same crack (In the figure, the size of the crack has been exaggerated for illustrative purposes).

8 Results

To validate our method, we computed irradiance volumes in two different scenes. The city scene represents a relatively large outdoor environment. The lighting is emitted slightly directionally from a sky dome to create smooth but clearly

visible shadows and some color bleeding due to indirect illumination. The results are shown in Figures 6(a-b). Neither the conventional point sampling nor our new method are able to reproduce all the fine detail seen in the ray traced image. Our method gives a result that is blurry, but visually pleasing. On the other hand, point sampling suffers from darkness leaking from insides of the buildings, resulting in visually objectionable artifacts.

Secondly, we computed an irradiance volume for an interior scene (see Figure 6(c)). Light falls from the window and illuminates the room both directly and indirectly. The room contains detail that cannot be captured with an irradiance volume without using an excessive resolution. In this case the volume computed with point sampling shows distracting aliasing and leakage. Our method smoothes out the excessive detail and avoids the leakage problems, giving a more pleasing result.

Both scenes were modeled by artists who were briefly informed about the requirements that the point classification algorithm poses on the scene geometry. After flipping some reversed surface normals, the algorithm worked perfectly on both of the scenes.

Acknowledgements

The authors would like to thank the following people for proofreading and support: Timo Aila, Jaakko Lehtinen, Lauri Savioja and Jani Vaarala. The 3D models were made by Eetu Martola (the city model) and Tuukka Takala (the interior). This work was funded by Anima Vitae, Bitboys, Hybrid Graphics, Nokia, Remedy Entertainment and National Technology Agency of Finland (Tekes).

Algorithm 2 Pseudo-code for the precomputation.

Input
scene : scene geometry
Output
$c(i,j)$: grid of coefficients, i = index of grid node, j = index of spherical harmonic
Data structures
ray,qray : pair (*origin*, *direction*)
isect,isect2 : quadruple (*location,normal,valid,brdf*)
sample : pair (*ray, intersection*)
samples : list of candidate samples
bfpoints : list of locations where a ray hit a back-face
norm() : the normalization factors
Functions
findFirstHit(*ray,scene*) : returns the first intersection of *ray* with *scene*
incidentRadiance(*sample*) : returns radiance incident towards *sample.ray.origin* from *sample.intersection.location*

```
for all grid cells, cell do
    clear list samples
    clear list bfpoints
    for i = 1 to N_samples do
        ray ← random direction and random origin in cell
        bfhit ← false
        for all bfpoints, bfpoint do
            qray ← ray from ray.origin to bfpoint
            isect ← findFirstHit(qray,scene)
            if dot(isect.normal,qray.direction) > 0 then
                bfhit ← true
                break
        if !bfhit then
            isect ← findFirstHit(ray,scene)
            if isect.valid and dot(isect.normal,ray.direction) > 0.0 then
                bfpoints.add(isect.location)
                for all samples, sample do
                    qray ← ray from sample.ray.origin to isect.location
                    isect2 ← findFirstHit(qray,scene)
                    if dot(isect2.normal,ray.direction) > 0 then
                        samples.remove(sample)
            else
                samples.add(pair(ray,isect))

    for all samples, sample do
        L ← incidentRadiance(sample)
        for all corner nodes of cell, node do
            i ← index of node
            for j = 1 to 9 do
                c(i,j) ← c(i,j) + L Y_j(sample.ray.direction) b_i(sample.ray.origin)
            norm(i) ← norm(i) + b_i(sample.ray.origin)

for i = 1 to N_nodes do
    for j = 1 to 9 do
        c(i,j) ← c(i,j)/norm(i)
```

References

[1] Personal communication with Jaakko Lehtinen, one of the authors of Max Payne 2 by Remedy Entertainment, 2003.

[2] G. Greger, P.Shirley, P. M. Hubbard, and D. P. Greenberg. The Irradiance Volume. *IEEE Computer Graphics and Applications*, 18(2):32–43, 1998.

[3] J. H. Halton and G. Weller. Algorithm 247: Radical-inverse quasi-random point sequence. *Communications of the ACM*, 7(12):701–702, 1964.

[4] Janne Kontkanen and Samuli Laine. Ambient occlusion fields. In *Proceedings of ACM SIGGRAPH 2005 Symposium on Interactive 3D Graphics and Games*, pages 41–48, 2005.

[5] Hayden Landis. RenderMan in Production, ACM SIGGRAPH 2002 Course 16, 2002.

[6] Chris Oat. Irradiance Volumes for Games, Presentation at Game Developers Conference, 2005.

[7] R. Ramamoorthi and P. Hanrahan. An Efficient Representation for Irradiance Environment Maps. In *Proceedings of ACM SIGGRAPH 2001*, pages 497–500, 2001.

[8] R. Ramamoorthi and P. Hanrahan. On the relationship between radiance and irradiance: determining the illumination from images of a convex Lambertian object. *Journal of the Optical Society of America A*, 18(10):2448–2459, 2001.

[9] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments. In *Proceedings of ACM SIGGRAPH 2002*, pages 527–536, 2002.

[10] Sergey Zhukov, Andrey Iones, and Grigorij Kronin. An ambient light illumination model. In *Rendering Techniques '98 (Proceedings of the Eurographics Workshop on Rendering)*, pages 45–55, 1998.

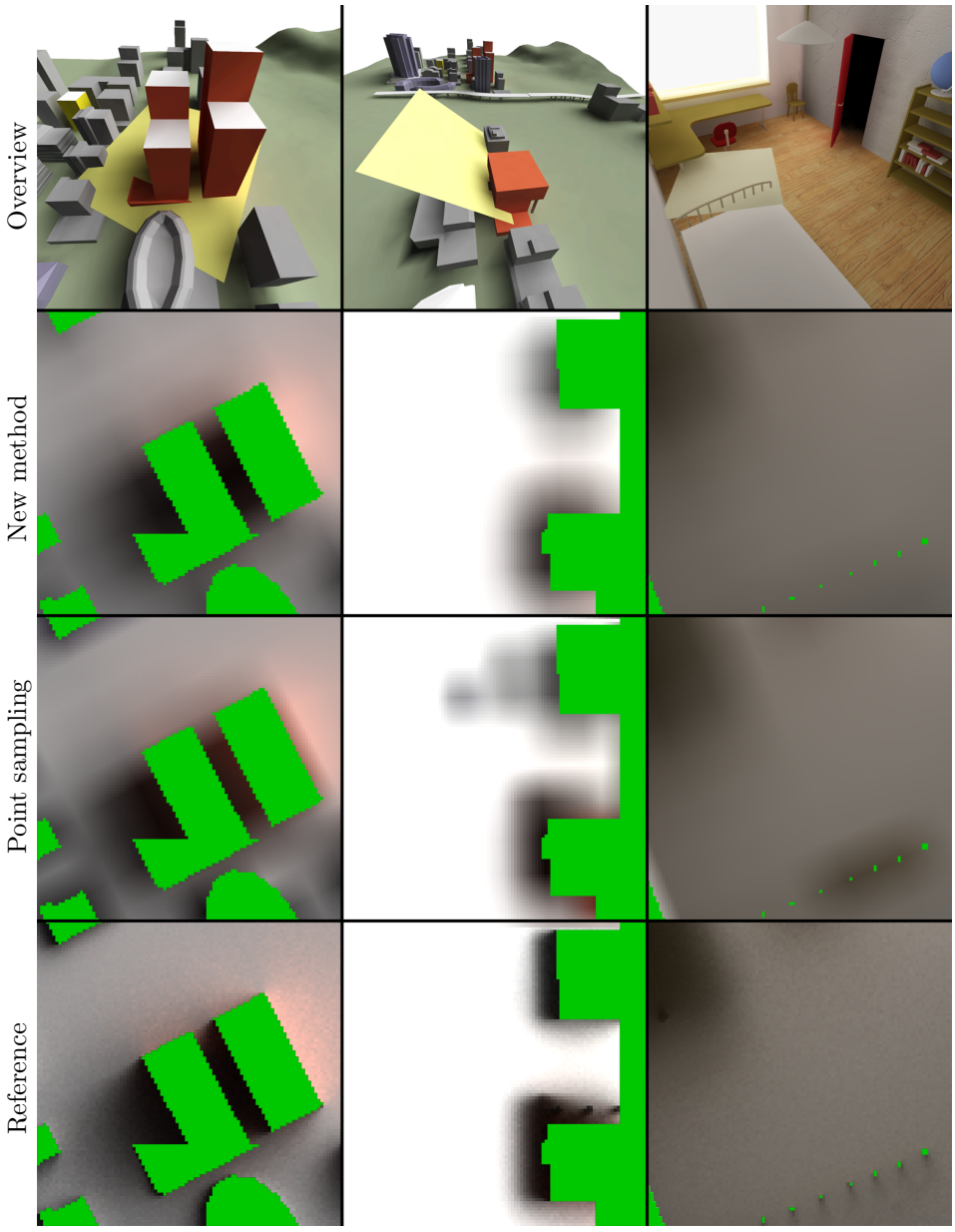


Figure 6: Sampling methods compared. The columns **(a-c)** correspond to different scenes/views. The **top row** shows images rendered using our method. In each image, the flat plate is illuminated by the irradiance volume. The **second row** shows the irradiance incident on the plate. The **third row** shows the corresponding result with conventional irradiance volume and the **last row** is the ground truth obtained by ray tracing. Bright green indicates the parts of the plate that are outside the domain of interest.