A Density Estimation Technique for Radiosity

M. Lastra C. Ureña J. Revelles R. Montes
Departamento de Lenguajes y Sistemas Informáticos
E.T.S.I. Informática. Universidad de Granada.
C/Periodista Daniel Saucedo Aranda s/n E-18071 (Granada-Spain)
mlastral,almagro,jrevelle,rosana@ugr.es

Abstract

Radiosity computation on scenes including objects with a complex geometry, or with a large number of faces and meshes with very different sizes, is very complex. We present a new method (based on the Photon Maps method [7]) where density estimation on the tangent plane at each surface point is performed for irradiance computation by using photon paths (line segments traveled by a ray) instead of photon impacts. Therefore we improve the results for scenes containing small objects which receive only a few impacts. Also, geometry is completely decoupled from radiosity computation.

Keywords

Radiosity, Density Estimation, Monte Carlo, Rendering

1 Introduction

In this article a method for computing the radiosity value at any point of a scene is presented. This process is divided into two phases: particle tracing and density estimation [7, 13]. During the particle tracing phase all the rays that are generated must be stored, along with some additional information. This information (a list of rays, each of them, carrying a certain amount of energy) is used to perform density estimation at any point in order to compute the irradiance. This density estimation is done on the plane tangent to the surface point by computing how many rays intersect a disc on the tangent plane centered on that point. Several optimizations are used to reduce the number on ray-disc intersections computed and to speed up the intersection test computation.

2 Motivation and Previous Work

Let's suppose a scene composed of various polygon meshes. The size of each mesh is very different, and the size of each polygon is proportional to the size of the mesh it belongs to. There are several methods that could be used to compute radiosity values. Hierarchical radiosity [6] with adaptive meshing could be used, but the number of small faces makes it difficult to control the mesh resolution inside a mesh. Also, the minimum number of top-level patches can not be smaller than the number of different meshes, which might be large. If hierarchical radiosity with clustering is used [15], then, small faces would still make it very difficult to create a set of patches with adaptive resolution inside large meshes.

Another solution is to use a particle tracing pass and to count hits on each face [1]. The main problem of this technique is that variance is inversely proportional to face size [12]. Therefore, small faces or small meshes either cause high variance or require a huge number of photons. The hits density might be enough for larger faces, but not for the smaller ones. Therefore, a few faces will appear quite bright but most of them will appear almost black.

All the problems mentioned come from the usage of patches for two different purposes. The patches are at the same time: elements of the geometric model of the scene and elements used in the reconstruction of the radiosity function. Therefore, solutions where the geometry is decoupled from the radiosity seem to be more suitable.

These kind of solutions have been proposed by several authors, who introduced particle tracing and photon maps or density estimation [13, 7]. However, there is again a drawback of this technique. Small objects cause high variance or require a huge number of photons. A certain hits density might be enough for larger objects but not for smaller ones [18].

Finally let us consider another way of calculating radiance values presented in [5]. Using particle tracing and storing the approximated distribution in a uniform 3D grid, density estimation is fully decoupled from geometry. The 3D structure allows to approximate the radiosity value on any point inside the cube, with any orientation. However, only a low resolution approximation to radiosity is achieved, because of the storage requirements imposed by the 3D grid.

3 Density Estimation on the Tangent Plane

As stated before, this rendering method can be divided in two phases: a photon tracking phase and a density estimation phase. During the first phase, photons are shot from the light sources in the scene. It can also be viewed as rays being shot from the light sources of the scene. When a photon strikes a surface it can be reflected or absorbed. At this stage of our work, no transparent objects are considered and therefore no particles are transmitted to the interior of any surface. The BRDF and reflectivity of this surface determines both whether a photon should die (be absorbed) or should be reflected, and also the direction in which it is reflected.

The second phase uses the information gathered in the previous phase in order to compute the irradiance at any point of the scene. Once the irradiance value is known at a point x, its radiosity value can be computed directly. In order to estimate the irradiance at a given point x, a disc, on the tangent plane of that point, is used. This disc is defined by the normal at x, a center point which is x and a value for the radius. At any point the same radius will be used. This restriction will be overcome in future work. Adding the energy carried by each particle or ray which intersects that disc, and dividing by the area of the disc, an estimation for the irradiance at x is obtained.

The expression of the radiance leaving a point x in direction ω_o [9] is defined in expression 1:

$$L_r(x,\omega_o) = \int_\Omega f_r(x,\omega_o,\omega)\; L_i(x,\omega)\; cos(\theta)\; d\omega \qquad (1)$$

where θ if the angle between the incident direction ω and the normal vector at x, $L_i(x,\omega)$ is the incident radiance from direction ω at x, and $f_r(x,\omega_o,\omega)$ is the BRDF at x. If we only use diffuse surfaces (that is, $f_r(x,\omega_o,\omega) = \rho(x)/\pi$), equation 1 can be simplified and radiosity (equation 2) values are computed:

$$L_r(x) \;=\; \frac{\rho(x)}{\pi}\int_\Omega L_i(x,\omega)\; cos(\theta)\; d\omega \;=\; \frac{\rho(x)}{\pi}E(x) \qquad (2)$$

where $\rho(x)$ is the reflectivity at x and $E(x)$ the irradiance at x. Equation 2 shows that in order to compute the radiosity values at a point x it is only necessary to know the irradiance value at that point. This irradiance is approximated by adding the energy carried by the rays that intersect a disc, centered at x, and dividing by the area of that disc.

After a photon is created or reflected, it follows a straight path until it hits another surface or leaves the scene. We call *ray* each of these segments visited by a particle. A *ray* is defined by a origin o and a direction vector v. During the first phase, a vector of m rays are generated $\{(o_1,v_1),(o_2,v_2),\ldots(o_m,v_m)\}$

Let $P = \{1\ldots m\}$ be the set of indexes of all the rays generated during the photon tracking process. $I_r(x) \subseteq P$ is defined as the set of the indexes of all the rays that intersect the disc centered at x with radius r . After these definitions, the expression of the estimated irradiance is:

$$E(x) \;\approx\; \frac{\sum_{i\in I_r(x)}\phi_i}{\pi r^2} \qquad (3)$$

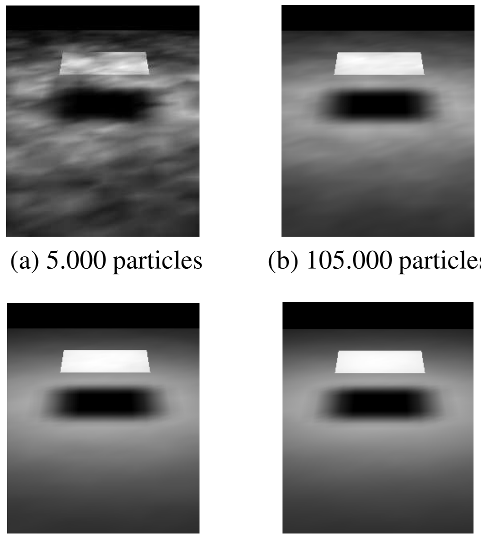
where ϕ_i is the energy carried by the i-th ray.

The bandwidth used by the kernel density estimation is the radius of the disc used. When a small disc is used the estimation will be more accurate but more rays are needed to reduce noise. On the other hand, if a big disc is used the estimation will be less accurate but noise is reduced [17].

In figure 1 a scene is shown with a plane projecting a shadow onto another plane. The radius used was 0.01 and 240 millions of particles were shot (distributed in 100 passes). This scene was used to illustrate how the size of the disc and the number of rays affect the resulting image. The before mentioned image can be used as reference.

Figure 1. Radius=0.01. 100x2.400.000 rays

In figure 2, the same radius (0.05) was used for every image, but the number of rays were gradually increased. This results in noise reduction as the number of rays increases.



(a) 5.000 particles (b) 105.000 particles

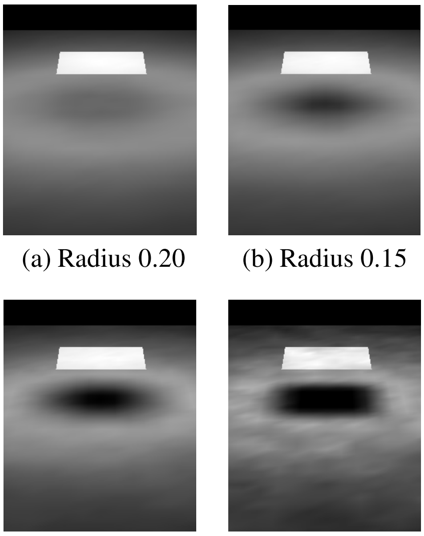
(c) 1.005.000 particles (d) 2.455.000 particles

Figure 2. Radius=0.05

Finally, in figure 3, 55.000 particles were shot and the radius was gradually reduced. The smaller the radius, the more accurate was the image and also the more noise appeared. In figure 4 the same process was repeated, but using 2.450.000 particles.

3.1 Error analysis

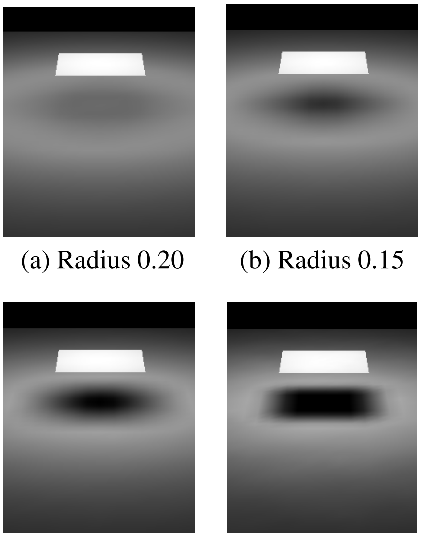
The described algorithm has three sources of error. We enumerate them here:



(a) Radius 0.20 (b) Radius 0.15

(c) Radius 0.10 (d) Radius 0.05

Figure 3. Particles=55.000



(a) Radius 0.20 (b) Radius 0.15

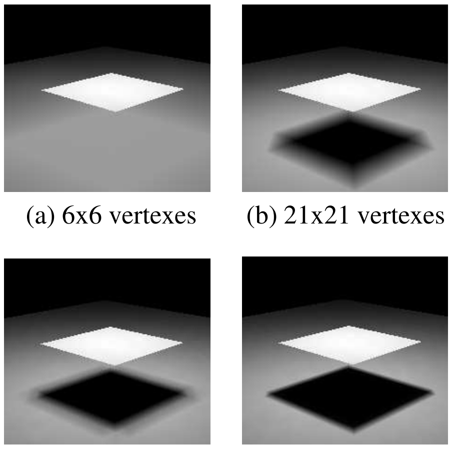
(c) Radius 0.10 (d) Radius 0.05

Figure 4. Particles=2.455.000

1. Our target is to compute irradiance at surface points. However it is not possible to do that for each surface point (there are infinite surface points), therefore it is necessary to select a finite set of points (mesh vertexes in our system) for irradiance computation. Then irradiance is not computed at points in between those selected, and some class of interpolation is done. We call this the *discretization error*. This error is lowered by using more sample points with smaller distances between them. Of course, the algorithm is lineally dependant on the number of sample points. In figure 5 a scene is shown where the number of mesh vertexes in the lower plane is gradually increased and it can be observed how *discretization error* decreases. Figure 5(a) shows an extreme case where no sample points are under the shadowed area.

2. We wish to compute the irradiance at selected points. By using particle-tracing we cannot approach the exact irradiance at those points. All we can do is to compute the average irradiance on a disc centered at that point. We call this error the *density estimation error*, because it is inherent to density estimation algorithms. It can be reduced by choosing a smaller radius r for the discs.

3. Computation of the average irradiance inside a disc is not exact, because we do that computation by using a stochastic or Monte-Carlo algorithm based on random walks. Therefore we do not get the exact value but a sample from a random variable whose mean is that value. The deviation from the sample to the target valued is characterized by the variance. Thus we simply call this kind of error *variance*. These can be diminished by increasing the number of particles n or by increasing disc radius r



(a) 6x6 vertexes (b) 21x21 vertexes

(c) 51x51 vertexes (d) 71x71 vertexes

Figure 5. Radius=0.01. Particles=2455000

We define Φ as the total energy emitted in the scene, that is:

$$\Phi \;=\; \int L_e(x,w)\,cos(n_x,w)\,dA(x)\,d\sigma(w)$$

Let $A = \pi r^2$ be the area of the disc used for density estimation at some point x. Let E be the average irradiance in the disc. It can be shown that the variance V is defined by the following expression:

$$V \;=\; \frac{\Phi^2}{n}\left(\frac{e+\epsilon}{\pi A} - e^2\right)$$

where e is equal to E/Φ, that is, e is a *normalized* version of E, n is the number of particles. With respect to ϵ it can also be shown that this value is approximately equal to $2ep$, where p is the average number of times a particle leaving the disc hits the disc again. This last number is near zero for small discs. Thus ϵ is usually also near zero.

Is is of course not possible to compute the value of V, but its expression shows that the variance is inversely proportional to the area of the disc used.

3.2 Selecting radius and number of particles

In our system we select mesh vertexes and perform irradiance computation on them. This means that we do not attempt to reduce discretization error, although that can be done by using some algorithm to try to find the best resolution for sampling irradiance. We focus on the other two sources of error: density estimation error and variance. Both errors can be characterized by two parameters which can be changed as desired. These are disc radius and number of particles.

In figure 2 we see how variance decreases when the number of particles increases. The noise in figure 2(a) is produced by a large variance due to a insufficient number of particles. Thus this shows the impact on the variance produced by the number of particles.

Figure 4 shows a sequence of images with low variance (due to a large number of particles), but with varying density estimation error due to varying radius. We observe that the effect of large radius is that the irradiance function appears *blurred* and details are lost. This shows the influence of radius onto density estimation.

Computing time and storage is mainly dependent on the number of particles. We can fix this value according to our quality requirements and our availability of time and/or memory. Then the radius should be made inversely proportional to the number of particles.

To illustrate this figure 3 shows again sequence of images obtained with a fixed number of particles and with varying radius. However in this case this number of particles is moderate (55.000). For a very large radius, (figure 3(a)) density estimation error becomes dominant, and shadow disappears. On the other hand, for a very small radius, variance is visible in the form of noise (figure 3(d)).

3.3 Photon Tracking process

The photon tracking process simulates the energy flow in the scene. This simulation is implemented by shooting particles from the light sources and by following the path in the scene of each particle until it gets absorbed or leaves the scene. Each particle goes through finite a sequence of *states*. The state of a particle can be described by a ray $r = (x, w)$ and a *weight* ϕ, which is a real value. The first state is called the *initial* state. At each step of the simulation, a particle goes from one state to another. This is usually called a *transition*. The ray of each state is selected stochastically, according to some probability density function, while the weight can be obtained as a function of previous weight and current ray.

Selection of the first state is made stochastically by using a probability density function called p_e. The value $p_e(x, w)$ is the (differential) probability for selecting ray $r = (x, w)$ as the first ray for a particle. Transitions are governed by the transition probability density function p_t. The value $p_t(r, s)$ is the (differential) conditional probability for selecting ray s for the next state when r is the ray of current state. Note that both p_t and p_e are probability density functions. This means that there exists two probability measures (P_e and P_t) such that

$$p_e(x, w) = \frac{dP_e(x, w,)}{dA(x)d\sigma(w)} \quad p_t(x, w, y, u) = \frac{dP_t(x, w, y, u)}{dA(y)d\sigma(u)}$$

where A area measure and σ is solid angle measure.

If $r = (x, w)$ and $s = (y, u)$, we can write $p_t(x, w, y, u)$ instead of $p_t(r, s)$. This value is cero when y is not the first visible point from x in direction w

3.3.1 Transition probabilities and particle weight update

Suppose the initial state of a particle is (x_0, w_0). Then its initial weight is:

$$\phi_0 \;=\; \frac{1}{n}\,\frac{L_e(x_0, w_0)}{p_e(x_0, w_0)}\,\cos(n_0, w_0)$$

where n_0 is the normal at x_0 and n is the total number of particles shot. After each bounce, the weight of a particle needs to be updated. Let assume that k-th state of a particle is (x_k, w_k), and the weight at that step is ϕ_k (with $k \geq 0$). For the next state the ray (x_{k+1}, w_{k+1}) is selected. Necessarily, point x_{k+1} is the first visible point from x_k in direction w_k. Direction w_{k+1} is selected at random. The next weight ϕ_{k+1} is obtained as:

$$\phi_{k+1} \;=\; \frac{f_r(x_{k+1}, w_{k+1}, -w_k)\,\cos(n_{k+1}, w_{k+1})}{p_t(x_k, w_k, x_{k+1}, w_{k+1})}\,\phi_k$$

where n_{k+1} is the normal at point x_{k+1}.

3.3.2 Monochromatic particles in diffuse scenes

The general case has been explained in the previous section, but in our application, scene surfaces are diffuse, this means that $f_r(x_{k+1}, w_{k+1}, w_k)$ can be written as $\rho(x_{k+1})/\pi$, where $\rho(x_{k+1})$ is the hemispherical-hemispherical reflectivity at x_{k+1} (this value is between 0 and 1). Emission of energy is also diffuse, thus $L_e(x_o, w_o)$ does not depend on w_o.

Moreover, we use two simple versions of p_e and p_t. With respect to p_e, it is defined as

$$p_e(x_0, w_0) \;=\; \frac{1}{\Phi}\,L_e(x_0, w_0)\,\cos(n_0, w_0)$$

the factor $1/\Phi$ is introduced because p_e must be normalized. (Φ is the total energy emitted in the scene) The above selection of p_e implies that particles do follow a diffuse distribution when created (directions near the normal are more often selected than directions near the horizon). This is also the case for the selection of an outgoing direction after a reflection because we use the following version of p_t:

$$p_t(x_k, w_k, x_{k+1}, w_{k+1}) = \rho(x_{k+1}) \frac{cos(n_{k+1}, w_{k+1})}{\pi}$$

factor $1/\pi$ above is also introduced for normalization. With these definitions, we can simplify the expressions for ϕ_0 and ϕ_{k+1}. We obtain:

$$\phi_0 = \frac{\Phi}{n} \qquad \phi_{k+1} = \phi_k$$

Thus all monochromatic particles have the same weight (Φ/n) at all steps when only diffuse surfaces are used.

Note that $\rho(x_{k+1})$ is a factor of p_t. This value is the survival probability. This means that with probability $1 - \rho(x_{k+1})$, the particle is absorbed after k states, and with probability $\rho(x_{k+1})$ it is reflected (at point x_{k+1}).

3.3.3 Coloured particles

The above formulation is valid for monochromatic light. However, in reality each photon has a wavelength, and both ρ and L_e depend on wavelength, producing coloured light. Thus we cannot assume that $L_e(x_0, w_0)$ nor $\rho(x_{k+1})$ are real values but functions of wavelength.

In order to account for colour, we have two options: one of them is to assign a wavelength to each particle. However this has a drawback, because it may cause uncorrelated noise at each wavelength. This implies that noise is more perceptible than it is for monochromatic light.

Second option we have is to use vectors for particle weights. In this case, each particle weight is a vector with one real value for each color component (three in our case, corresponding to the rgb colour model). This implies that particles are coloured particles instead of monochromatic or single-wavelength particles. This is an artificial concept introduced to perform the light transport simulation . The vector-weight of a particle can be interpreted as its *colour*. Each component of the colour of a particle is updated after each reflection by using the same formulation introduced previously.

Note that $\rho(x)$ is also a vector, with one real value for each component. This implies we can no longer use the value $\rho(x)$ as the survival probability. Instead of this, we need to use a survival probability which is equal for all wavelengths or colour components (otherwise some colour components may be absorbed and other may be reflected, which has no sense in this context).

Thus, we will assume that weights ϕ_k or ϕ_{k+1} are vectors, and this is also the case of hemispherical reflectivity $\rho(x)$. In our system, transition probabilities are equal to:

$$p_t(x_k, w_k, x_{k+1}, w_{k+1}) = S(\phi_k, x_{k+1}) \frac{cos(n_{k+1}, w_{k+1})}{\pi}$$

where $S(\phi_k, x_{k+1})$ is a real value (between 0 and 1) which depends on both previous weight vector ϕ_k and point x_k. Thus this value can be interpreted as the survival probability after step k. With this version of p_t, updating of vector-weights is done by using the following formulation:

$$\phi_{k+1} = \frac{\rho(x_{k+1})}{S(\phi_k, x_{k+1})} \phi_k$$

for any vector a and scene-point x, real value $S(a, x)$ is defined as:

$$S(a, x) = \frac{lum(a\,\rho(x))}{lum(a)}$$

For any vector v encoding a colour, $lum(v)$ is the luminance of v. Luminance of a colour v can always be obtained as a weighted sum of the components of v, that is, the luminance of v is the dot product of v and another vector c (whose components sum to one) [4].

Note that, for any colour a, luminance of $a\,\rho(x)$ is always smaller or equal to the luminance of a. Then we deduce that for any point x, $0 \leq S(a, x) < 1$. This ensures we can use $S(\phi_k, x_k)$ as the survival probability at step k.

3.4 Ray-disc Intersection calculation

In order to perform a density estimation on the tangent plane, ray-disc intersections need to be computed. First, it is necessary to obtain the intersection point i of the ray and the plane which contains the disc and then to check if the distance between the intersection point i and the center of the disc x is less or equal to the radius. If the distance squared is greater than the radius squared , the ray does not intersect the disc. This method implies several operations which can introduce a great overhead when many of these intersections tests are computed.

Because the disc size is generally relatively smaller, as compared to the size of the scene, only a small subset of rays will intersect each disc. A ray-triangle pre-intersection test, which is computed faster then the ray-disc one, could be used to discard rays that can not intersect each circle. If no intersection is found, then no intersection point needs to be computed. If a ray does not intersect a triangle containing the disc, then it cannot intersect the disc and therefore the ray-disc intersection test is not necessary. Therefore, time consumption is reduced unless most of the rays intersect the disc (which does not happen in our application).

Several ray-triangle intersection algorithms have been tested: one based on Plücker coordinates and the algorithms developed by Möller [10], Segura [11] and Badouel [2]. Also a performance analysis has been made about these algorithms in order to use the most efficient one. The use of Plücker coordinates proved to be the best method for our application.

3.4.1 Plücker coordinates

Plücker coordinates are used to accelerate ray-disc intersection tests. Any directed l line in 3D space can be represented using Plücker coordinates [16, 3, 14] as a six-tuple Π_l. Using the permuted inner product of two of these six-tuples the relative orientation of the two directed lines r_1,

r_2 can be obtained. If this product,$\Pi_{r1} \odot \Pi_{r2}$, is zero then the two lines are coplanar which means they intersect or are parallel. If the permuted inner product is positive then r_1 goes counterclockwise around r_2 and vice versa. If the permuted inner product is negative then then r_1 goes clockwise around r_2 and vice versa.

In figure 6 these two situations are illustrated. On the pictures on left side r_2 goes counterclockwise around r_1 and on the pictures on the right side r_2 goes clockwise around r_1. In figures $6(b)$, and $6(d)$ line r_1 is represented as a dot and is perpendicular to the paper and pointing into the paper.

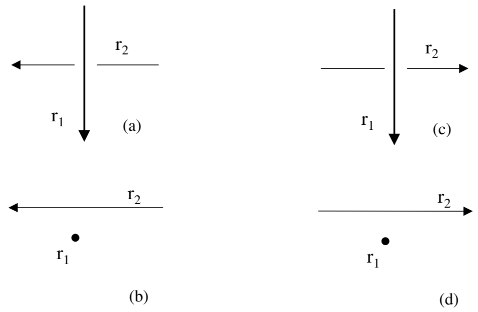


Figure 6. Relative orientation of two lines

Plücker coordinates can be used to test whether a ray intersects a polygon. If the edges of a polygon are represented using Plücker coordinates, a ray intersects that polygon if and only if it hits one of the edges or it goes clockwise or counterclockwise around all the edges. It is important to state that the ray must go around in the same way for all the edges, or otherwise it does not intersect the polygon.

In our application intersections at the back side of the polygon should be discarded. Thus, even if a ray goes in the same direction around all the edges only one orientation (counterclockwise) corresponds to a ray-polygon intersection at the front face.

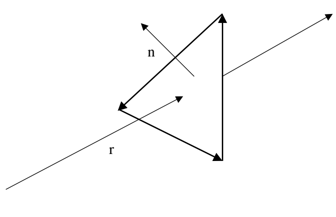


Figure 7. Ray-triangle intersection with oriented edges

In our implementation, we use a triangle formed by three directed lines for each disc. Each triangle is constructed counterclockwise (as in figure 7), which means its directed edges go counterclockwise when looking at the triangle's front face (the normal n comes towards the reader). If a ray intersects one of these triangles it must go counterclockwise around the three edges. Therefore, if the inner product of the Plücker coordinates of the ray and one edge is negative, then that ray does not intersect the triangle at the front face and neither the disc.

The ray-disc intersection test is therefore performed as follows. For each disc used for density estimation a triangle containing that disc is defined. The three edges are converted to directed lines. These three lines are then represented with Plücker coordinates. Also, each ray has previously been represented with Plücker coordinates. Before the ray-disc intersection test is computed, the ray is tested to go counterclockwise around the three edges, which means the permuted inner product of each edge and the ray must be positive or zero.

3.5 Ray Discretization

A fast ray-disc intersection test is not enough to efficiently process thousands or millions of rays at each vertex. The irradiance computation would not be efficient if every ray had to be tested for intersection with the disc associated to each point. In our application, the disc radius will always be small in comparison to the scene size. Therefore, most of the rays will not intersect each disc but the large number of intersection tests calculated would produce a remarkable overhead. A fast way to reject most of the rays that do not intersect each disc is needed. We have developed a ray discretization scheme to achieve this.

In order to explain how the discretization scheme works, first an initial assumption will be made: that only rays with the same direction are used. This simplification will help to understand the general case explained later. The rays can have different origins, but the same direction vector. Under these conditions we could precompute the intersection point of each ray with a plane perpendicular to this direction. The area of the plane where these intersections can be found must be inside the scene bounding sphere and can be divided using a 2D uniform grid. Thus, each cell in this grid will contain a list of rays whose intersection with the perpendicular plane are located inside its associated area. Each ray will therefore be in the ray list associated to one, and only one cell.

If a disc is projected (along the rays direction) onto the before mentioned plane, the projection of the disc will fall over some of the grid cells. The rays that can intersect the disc are only those whose intersection belongs to that set of grid cells covered by the projected disc. This way, the problem of finding which rays intersect a disc is reduced form three to two dimensions. Potential intersections are searched in a 2D grid.

It is necessary to extend the previous model to allow rays with arbitrary directions. The rays will be grouped considering its direction. Rays separated by a short arc length will be in the same group. Also, each group will have a representative direction vector v_i. A unit sphere, represent-

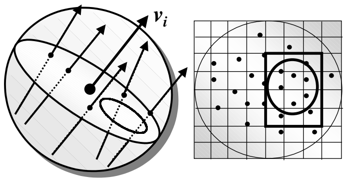


Figure 8. Ray discretization

ing all possible ray directions is first divided into patches. Each patch represents a set of directions. The direction of the vector that goes from the center of the sphere to the center of a patch will be the representative direction of all the possible directions in that patch. In figure 8 the direction indicator vector and the perpendicular plane associated are shown.

Once the correct patch or group for a ray is found, the ray is intersected with a plane that is perpendicular to the representative direction vector of that patch and that has already been divided by a 2D uniform grid. The ray is than added to the list of the cell that contains the intersection point. This process is repeated for every ray.

After that preprocessing, each disc is projected onto the 2D uniform grid associated to each patch or ray group and only ray-disc intersection tests need to be computed for the rays which are on the list of the grid cells that fall under the projected disc.

There is still one thing to consider about the projection of the disc. Rays in the same group are not parallel but the arc length between any of them and v_i is smaller or equal to an angle α_i (for the i-th patch). This fact must be taken into account when projecting the disc. Now a ray may intersect the disc outside the projection. This situation can be seen in figure 9.

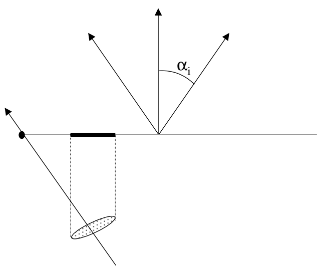


Figure 9. Disc projection

Therefore the projection of the disc needs to be adjusted taking into account the range of possible directions in a patch. This can be achieved by computing the maximal area on the plane where an intersection of one of the rays of the patch and the disc can happen.

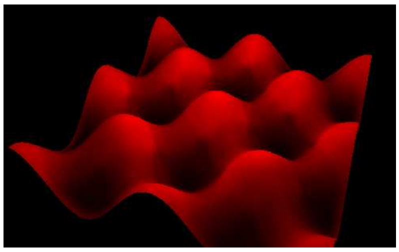
3.6 Avoiding Artifacts

The density estimation technique presented is an approximate technique which may produce certain errors in the irradiance estimation, and therefore, visible artifacts may arise in the final image.

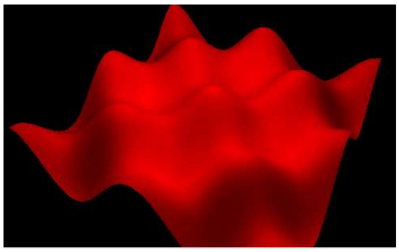
Two kind of artifacts will be discussed here. One of them appeared in non convex parts of a mesh and the other one arose when part of the disc was unreachable for any ray. In the next sections each artifact will be explained with more detail and solutions will be proposed.

3.6.1 Concave meshes artifact

This artifact produced a bias in concave meshes or parts of a mesh. Then non convex parts of the scene resulted much darker as they should as if no rays, or only a small amount of them reached these zones. An example is shown in figure 10 on top. On the bottom of this figure the correct image is also shown.



(a) corrected solution



(b) uncorrected solution

Figure 10. Concave meshes bias

Figure 11 will be used to explain why this happens. Point x lies within a concavity and most of the rays mainly intersect the mesh before they can intersect the disc. Therefore all these rays are treated like blocked rays, as if a shadow is projected on x. In order to avoid this undesired effect, not only the ray parameter value of the first intersection of each ray is stored, but also the second one (in case it exists). If a ray intersects with the disc before the second intersection point (the second intersection with an object of the scene in the direction of the ray) and the first intersection point is in the same mesh as x, then the ray is counted as intersecting the disc. Otherwise it is considered that the ray does not intersect the disc.

In figure 12 the intersection of the ray and the disc associated to point x happens after the second intersection point. In this case the energy carried by the ray will not have any

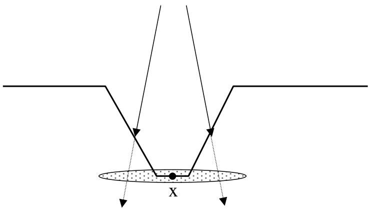


Figure 11. Concave meshes bias

influence on x because it is in a shadowed region for rays coming from that direction.

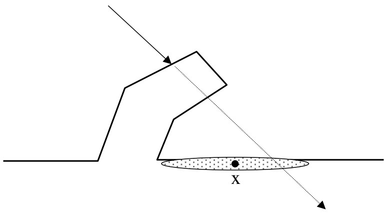


Figure 12. Another concave area

3.6.2 Unreachable regions

There is another kind of artifact that was detected in a scene which was formed by a room and a light source on the ceiling. In this scene, the boundaries of the walls and the floor were much darker than the rest of the scene. The situation is in someway similar to the classical boundary bias as described in [17]. In figure 13 this situation is illustrated.

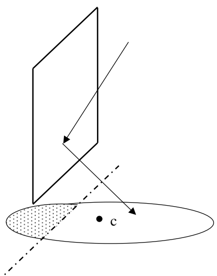
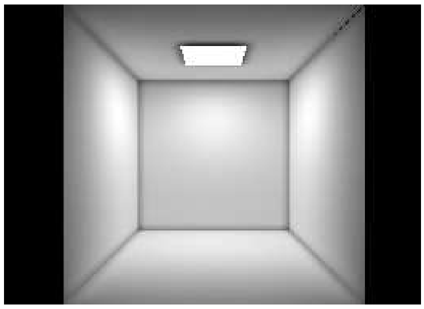


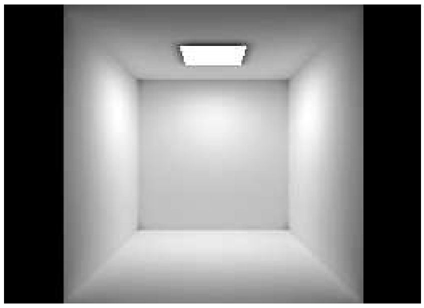
Figure 13. Unreachable regions bias

Any ray reflected or originated on the plane of figure 13 can not reach the shaded part of the disc, but, on the other hand, the energy of these rays gets divided by the total area of the disc. This produces a biased lower value for the irradiance in these areas. In general, this happens when the full disc is not visible from the origin of the ray, because part of it is under the plane tangent to the surface at the ray origin. This line divides the area of the disc that can be reached by a ray and the area where that can not happen. This kind of situation has to be detected (using the distance from the center of the disc to the plane) and for each ray, the area of the disc where such a ray could intersect that disc is calculated and used instead of the whole area.

In figure 14 an example of this kind of bias is shown, as well as the correct image.



(a) uncorrected solution



(b) corrected solution

Figure 14. Unreachable regions bias

4 Results

In figure 15 a complex scene where irradiance estimation on the tangent plane was used to obtain radiosity values is shown. This scene is composed of 46701 vertexes and 4.500.000 primary rays were used, but dividing the whole process into 3 passes of 1.500.000 rays due to the main memory storage requirements. The number of rays used is high to ensure a low noise level. The bounding sphere of the scene has a radius of 10 units and the radius used for each disc was 0.2. The irradiance estimation time for each vertex in each pass was 0.069 seconds on a PC with a AMD Athlon XP 1900+ CPU running Linux.

5 Comparison with Photon Maps

The photon maps technique [7] is a density estimation method which uses photon impacts on the objects stored in a data structure called photon map. The search for photons is performed inside a sphere. The density estimation method presented in this article has been compared with the photon map method using the source code published in [8]. Both algorithms were integrated in the same rendering system and also the same photon tracking implementation was used in both cases.

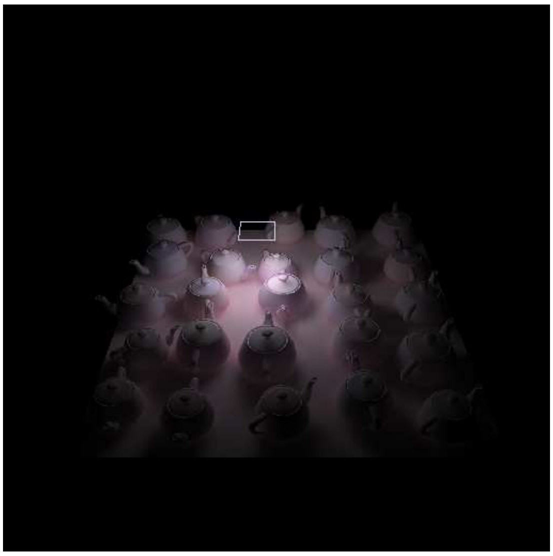


Figure 15. Complex scene

There is an important difference about the density estimation used in each method. When using photon maps, the n nearest photons inside a sphere of radius r centered at point x, where irradiance needs to be estimated, are located. If this sphere contains m photons, where $m > n$, the rest of the photons, $m - n$, are not used. In the method we present all the rays that intersect the disc centered at x are used, thus we use a high value for n was used to force the photon maps method to use all the photons contained in the sphere.

The artifacts or errors produced by the density estimation on the tangent plane, along with its solutions have already been discussed. The photon maps method also produces images with errors under certain circumstances. Some of these situations will be studied.

In figure 16 a plane is shown and density estimation is performed at a point x close to one of its edges. There is a region R of the space where no photon impacts can be found but the irradiance estimate gets finally divided by the volume of the whole sphere. This produces objects with too dark edges. In figure 19(a) an example can be seen. The result is similar to the unreachable regions artifact introduced before (darkening near the edges).

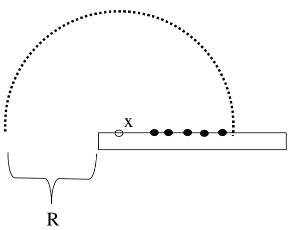


Figure 16. Bias near the edges

Another situation where a biased image is obtained is illustrated in figure 17.

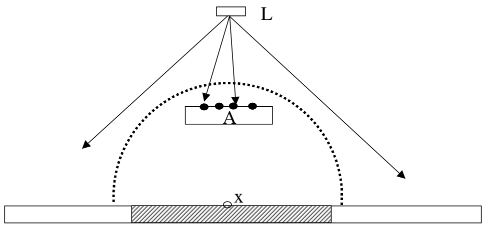


Figure 17. Missing shadow bias

On the top of the figure a light source L can be seen and in the middle an obstacle A, which should produce a shadow on the plane below, has been placed. If the distance between this plane and A is less than the radius of the sphere used for irradiance estimation on a point like x, then the shadow will not be obtained. This happens because photons on A will be used to compute the irradiance estimate at x. This kind of problems do not arise when using rays instead of photon impacts. An example of this situation can be seen in figure 18

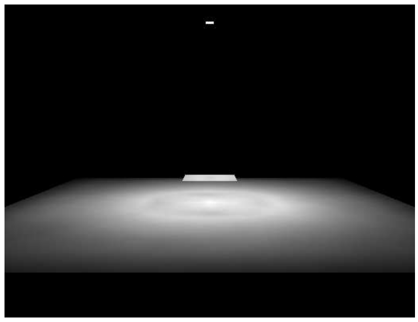
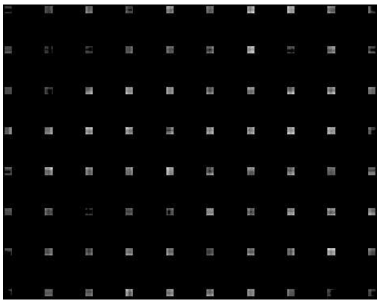


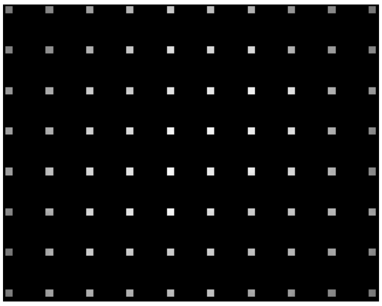
Figure 18. Missing shadow bias

Finally, as stated in the introduction, when small objects are part of a scene, they might appear much darker than they should, because they do not receive enough photon impacts. In figure 19 a scene is shown where a set of little rectangles that are being illuminated from above. The observer is approximately situated under the center of the light source. In figure 19(a), obtained using photon maps, the rectangles result much darker and roughly illuminated as in figure 19(b) which was obtained applying the tangent plane density estimation.

The main advantage of the photon maps method is its speed during the photon location process. Some tests have been done and it outperforms in terms of time consumption the density estimation on the tangent plane because it is more complicated to work in the lines space than in the 3D points one. On the other hand, we are currently working on better methods than the ray discretization presented to obtain a fast selection of the ray candidates to intersect a disc.



(a) Photon Maps



(b) Tangent plane estimation

Figure 19. Particles=50000

6 Future Work

The storage requirements for the rays should be decreased in order to allow more rays to be created. Also, the ray-disc intersection test time could be reduced by more efficient indexing schemes. On the other hand, some extensions to the basic algorithm are also planned: non purely diffuse surfaces and other geometric models should be allowed, the use of more advanced density estimation algorithms and obtention of iterative solutions

7 Acknowledgments

This work has been supported by the research project coded TIC2001-2392-C03-03 (Spanish Commission for Science and Technology)

References

[1] J. Arvo. Backward Ray-Tracing (course Developments in Ray Tracing).ACM Siggraph Course Notes 12. 259-263, 1986.

[2] D. Badouel. An Efficient Ray-Polygon Intersection. Graphics Gems, ed. Andrew Glassner. Academic Press pp. 390-393. 1990 http://www.acm.org/tog/GraphicsGems/.

[3] J. Erickson. Plücker Coordinates. Ray Tracing News 10(3). December 2, 1997.

[4] Foley, et al. Computer Graphics: Principles and Practice, Addison Wesley, 1990.

[5] G. Greger, P. Shirley, P. Hubbard, D.P. Greenberg.The Irradiance Volume. IEEE Computer Graphics and Applications 18(2) pp.32-43, 1998.

[6] P. Hanrahan, D. Salzman, L. Aupperle. A Rapid Hierarchical Radiosity Algorithm. Computer Graphics, 25(4), July 1991. ACM Siggraph'91 Conference Proceedings.

[7] H.W. Jensen.Global Illumination Using Photon Maps. Rendering Techniques'96 (Pueyo, Schroeder, eds.) pp.21-30, Springer-Verlag, 1996.

[8] H.W. Jensen. Realistic Image Synthesis Using Photon Mapping. A K Peters. 2001.

[9] J.T. Kajiya. The Rendering Equation. Computer Graphics, 20(4) pp.143-150. ACM Siggraph'86 Conference Proceedings. 1986.

[10] T. Möller, B. Trumbore. Fast, Minimum Storage Ray-TRiangle Intersection. Journal of Graphics Tools 1(2). pp21-28. 1997. http://www.acm.org/jgt/papers/MollerTrumbore97/

[11] R.J. Segura, F.R. Feiro. Algorithms to Test Ray-Triangle Intersection. Comparative Study. WSCG 2001.Conference Proceedings. Pilsen. 2001.

[12] P. Shirley. Time Complexity of Monte-Carlo Radiosity. Computer & Graphics 16(1), pp.117-120, 1992.

[13] P. Shirley, B. Wade, P. Hubbard, D. Zadeski, B. Walter, D.P. Greenberg. Global Illumination via Density Estimation. Rendering Techniques'95 (hanrahan, Purgathofer eds.) pp.219-23, Springer-Verlag, 1995.

[14] K. Shoemake. Plücker Coordinate Tutorial. Ray Tracing News 11(1). July 11, 1998.

[15] F. X. Sillion. A Unified Hierarchical Algorithm for Global Illumination with Scattering Volumes and Object Clusters. IEEE Transactions on Visualization and Computer Graphics 3(1). 1995.

[16] S. Teller. Computing the Antipenumbra of an Area Light Source. Computer Graphics (Proc. Siggraph '92), 26:139-148, July 1992.

[17] B. Walter, P. Hubbard, P. Shirley, D.P. Greenberg. Global Illumination Using Local Linear Density Estimation. ACM Transactions on Graphics 16(3) pp.217-259, 1997 .

[18] A.Wilkie (advisor: W. Purgathofer). Photon Tracing for Complex Environments. PhD. dissertation. The Institute of Computer Graphics and Algorithms, Technique University of Vienna.