Computation of Irradiance from Triangles by Adaptive Sampling.

Internal report LSI-2000-1
Depto. Lenguajes y Sistemas Informáticos. University of Granada.

C. Ureña
Dpto. de Lenguajes y Sistemas Informáticos,
E.T.S. de Ingeniería Informática. Universidad de Granada.
18071 Granada.
e-mail: almagro@ugr.es

March 15, 2000

Abstract

We introduce an algorithm for sample positioning in a planar triangle, which can be used to make numerical integration of an arbitrary function defined on it. This algorithm has some interesting properties which make it suitable for applications in the context of realistic rendering. We use an adaptive triangle partitioning procedure, driven by an appropriate measure of the error. The underlying variance is shown to be bounded, and in fact it can be controlled, so that it approaches the minimum possible value. We show results obtained when applying the method to irradiance computation, in the context of final-gather algorithms. We also describe a C++ class which offers all required functionality, and we made available its source code.

1 Introduction

One of the tasks in the production of rendering systems is the design of algorithms for computation of reflected radiance from a point (which will be called **p** in the rest of this section) in a direction, due to energy coming from a polygonal source and/or reflector. This problem is related to others, such as the computation of point-to-patch form factors, in the case of polygonal patches. Usually, planar polygons are decomposed in triangles: the problem is then solved for each of the resulting triangles, and results are added.

All those calculations can be modelled as the computation of an integral on the triangle, by using the well known cosine term (which can be absorbed into the projected solid angle measure) which weights incoming radiance onto the point.

With the assumption of full visibility between the point and the patch, the problem becomes simpler to deal with. In this case closed-form expressions for the point-to-patch form factor can be used in diffuse environments [3]. Even for non-diffuse reflectors (such as phong-like surfaces) we can find deterministic algorithms which compute exact form-factor and/or reflected radiance [1].

The problem, however, is not so straightforward to solve in the presence of occluders. For planar polygonal occluders, we can decompose the triangle into sub-triangles, each of them being fully visible or fully occluded, and then solve the problem for each visible triangle [1]. This implies projection and clipping of all possible occluders against the source triangle, which can take relatively large computing times.

Monte-Carlo algorithms have been widely used for this purpose. One option is to use *importance sampling*. The probability measure used for positioning the samples is proportional to projected solid angle measure. This is straightforward when the whole hemisphere is being sampled. Stratification is also possible by using an mapping with constant Jacobian from the unit square $U = [0,1)^2$ to the unit-radius circle. Samples are computed in U and then mapped to the sphere.

However the above cannot be used when we want to sample just a given triangle, because in these cases a probability function should be defined on the triangle. In this case, a mapping can be defined from the unit rectangle to that polygon. If we consider triangular polygons, there exists a constant Jacobian mapping from the unit rectangle to any triangle [7, 8]. This property is interesting, because it leads to uniform samples distributions with respect to area (on the triangle). Moreover, the use of a mapping allows to take advantage of stratified sampling.

However, the above algorithm can lead to high variance because area measure (on the planar triangle) and projected solid angle measure are not related and may differ too much, specially for triangles covering a big solid angle when projected onto $\mathbf{p}$. This problem can be solved by using a combination of sampling strategies [10]. We consider here the solid angle covered on the hemisphere by any planar region when projected over $\mathbf{p}$. Projected solid angle measure is equal to solid angle measure times the cosine factor.

A better technique is to use a mapping from U to the triangle, such that regions with equal areas are mapped to regions with equal solid angle. As the solid angle measure is closer to projected solid angle measure, the variance for near triangles is reduced, and we can use stratification also. A simplified approximation to this mapping was proposed by Wang [11], and the exact formulation was given by Arvo [2]. This algorithm is very useful for the final-gather step in radiosity computation [9].

The solid angle measure differs with projected solid angle measure in the cosine factor. In the case that, for a given triangle, this cosine factor has wide variations then the probability measure used by Arvo's procedure still differs from projected solid angle measure, thus yielding high variance. The ideal probability measure should be one which assigns equal probability to regions with equal projected solid angle. This is usually called *importance sampling* in Monte-Carlo literature, because we also take into account the cosine function, which is a component of the function being integrated.

This can be obtained by designing an appropriate mapping, as before. However, the design of such a mapping is far from simple as it involves finding the primitive function for elliptic integrals.

In this paper we propose an adaptive sample positioning algorithm which places the samples by using a new probability density function on the source triangle. This probability density function (pdf) approaches the ideal one described above. The algorithm proceeds by adaptive splitting of the original triangle, and, at each step, it consider the error involved in sampling by using the current partition. When the error is below a given threshold, each of the resulting sub-triangles are sampled by using Arvo's algorithm. Measuring the error is done by a function which gives an upper bound of the variance involved for each subtriangle. A C++ class has been designed which packages all required functionality. This class yields a set of samples for any given input triangle.

The article is structured as follows: first (Section 2) we introduce the problem to be solved, and the notation. In next section we find the error involved and we also show how it can be bounded. In section four we give the details of the algorithm and its implementation in C++. Last section (five) includes the results obtained for irradiance computation, including sample distributions and some images.

2 Problem statement.

2.1 Measures on C

Consider any plane Π in R^3, and a planar triangle T contained in Π. We assume that all the points in T have non-negative Z coordinates. These coordinates are expressed with respect to a local coordinate system whose origin is in $\mathbf{p}$, and whose Z axis is aligned with the normal at $\mathbf{p}$. In the case that a triangle has a portion of it with negative Z coordinates, this portion can be clipped and discarded, because its $\sigma_\perp$ measure will be zero, and thus it won't contribute to the integral.

We call S to the spherical triangle obtained as the projection of T onto the unit-radius hemisphere Ω (with center at the origin), and we also define E as the vertical projection of S onto the unit-radius circle C in the plane $z = 0$ (see Figure 1). Thus E is

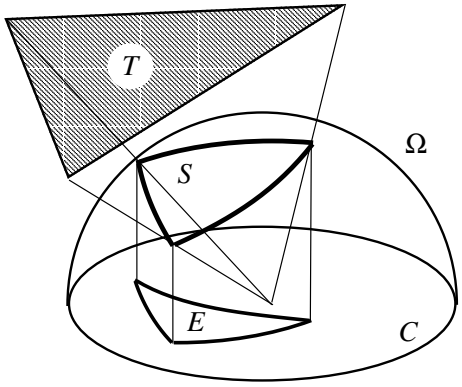

Figure 1: Regions $S \in \Omega$ and $E \in C$ for an example planar triangle T

a planar region bounded by three curves. Each one of those curves is an arc of ellipse

[4]. There is a natural correspondence between these three sets, because any point (or any region) in any of them can be mapped to its projection on the other two.

Thus, any measure function defined in one of these sets is also defined in the other two, by using the previously mentioned correspondence. For any fixed triangle T (which is included in a plane called Π) we can conceive three different measures on the unit-radius circle C:

1. The area measure on the plane Π (containing T), which will be called A. For any $C' \subseteq C$, the value $A(C')$ is the area of the projection of C' onto Π.

2. The area measure on Ω, also called *solid angle* measure, and usually written as σ. For any $C' \subseteq C$, the value $\sigma(C')$ is the solid angle covered by the projection of C' onto Ω.

3. The area measure on C, which is related to σ and called *projected solid angle* measure. We will write this as $\sigma_\perp$. For any $C' \subseteq C$, the value $\sigma_\perp(C')$ is the area of C', measured in the plane $z = 0$.

There is a well known relation between measures σ and $\sigma_\perp$. For any $\mathbf{u} \in C$, it holds that

$$\frac{d\sigma_\perp(\mathbf{u})}{d\sigma(\mathbf{u})} \;=\; \cos(\mathbf{u})$$

where cos is a function on C (thus also defined on Ω and Π), defined as

$$\cos(\mathbf{u}) \;=\; (1 - u_x^2 - u_y^2)^{1/2}$$

where $\mathbf{u} = (u_x, u_y)$. Thus $\cos(\mathbf{u})$ is the Z coordinate of the projection of $\mathbf{u}$ onto Ω.

2.2 Monte-Carlo Integration in C

In computer graphics, there are a number of rendering algorithms which include the computation of the following class of integrals:

$$I \;=\; \int_E f(\mathbf{u})\, d\sigma_\perp(\mathbf{u}) \tag{1}$$

where f is any integrable function on E, with $f \geq 0$. The complexity of this integration obviously depends on the available information about f. We consider the following examples of definitions for f:

- $f(\mathbf{u}) = L_i(\mathbf{u})$ (where $L_i(\mathbf{u})$ is the incoming radiance from direction $\mathbf{u}$ onto the origin). In this case I is the irradiance at the origin due to power coming from region S of Ω.

- $f(\mathbf{u}) = L_i(\mathbf{u})\, f_r(\mathbf{u}, \mathbf{v})$ (where f_r is the BRDF at the origin). Now, I is reflected radiance from the origin in direction $\mathbf{v}$, for any $\mathbf{v} \in \Omega$.

- $f(\mathbf{u}) = V(\mathbf{u})$ (where $V(\mathbf{u})$ is the visibility function between the origin and the projection of $\mathbf{u}$ on T). Then, the value I is the point-to-patch form factor from the origin to T.

These examples show the importance of designing efficient integration techniques for the above integral. In the general case, analytical integration is not available in short computing time, because of the complexity of f. Usually we can just evaluate f at certain points. This leads to the usage of either numerical quadrature rules or Monte-Carlo sampling. We will focus on the Monte-Carlo integration framework.

To approximate integral (1) using Monte-Carlo sampling, we have to use a probability measure P on C which drives sample point placement. We call p to the *pdf* associated to P (with respect to $\sigma_\perp$), that is:

$$p(\mathbf{u}) \;=\; \frac{dP(\mathbf{u})}{d\sigma_\perp(\mathbf{u})}$$

probability measure P should be normalized, that is, $P(E) = 1$. For any valid measure P, we obtain a secondary estimator I' of I by combining n primary estimators, obtained by using n independent samples from distribution P. The value I' is obtained as:

$$I' \;=\; \sum_{i=1}^{n} \frac{f(\mathbf{u}_i)}{p(\mathbf{u}_i)} \tag{2}$$

where $\{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$ is a set of samples distributed according to P.

The efficiency of each sampling distributions is related to the underlying variance induced by that distribution. The variance is written as V, and is equal to $(U - I^2)/n$, where U is defined as follows:

$$U \;=\; \int_E \frac{f^2(\mathbf{u})}{p(\mathbf{u})}\, d\sigma_\perp(\mathbf{u})$$

as the value I is fixed, a *good* sampling distribution is one which minimizes U, which obviously depends on P. We will focus on sampling distributions which are totally independent of f. Thus, sample point placement algorithms do not need *a priori* information about f.

We can select P such that $P(R) \propto A(R)$, for any $R \subseteq E$. The probability for placing a sample in any region is proportional to the area of that region when projected over T. This sampling distribution is straightforward to implement and shows poor performance in many cases because of perspective distortion and the (implicit) cosine term in (1). The increase of variance due to perspective distortion can be avoided by using $P(R) \propto \sigma(R)$. In this case, the probability for placing a sample in R is proportional to the solid angle covered by R. This can be implemented by using the mapping proposed in [2]

Finally, the best possible distribution which does not depends on any information about f, is the one which makes $P(R) \propto \sigma_\perp(R)$. As the probability measure is proportional to the integration measure, the variance is smaller and not affected by neither perspective distortion nor by the cosine term. In this case, p is constant and equal to $1/\sigma_\perp(E)$. We define U_0 as

$$U_0 \;=\; \sigma_\perp(E) \int_E f^2(\mathbf{u})\, d\sigma_\perp(\mathbf{u}) \tag{3}$$

The value U_0 is the smallest possible of U for all the measures P independent of f.

A possible way to implement this distribution is to use an area preserving mapping from the unit rectangle $[0,1]^2$ into E. However, this is complicated because of E shape (a planar region bounded by three arcs of ellipse). To achieve this, we need to find the inverse function of a class of elliptic integrals. This is known as the Jacobi function [5], which can be used to compute the sector of an ellipse which has any given area. This function does not have a simple analytical expression, thus is not straightforward to evaluate, and needs to be approximated by numerical methods.

However, as we show in the following section, we can use an adaptive sample placement algorithm whose variance is as close as we wish to the smallest possible value.

3 Adaptive sample placement

3.1 A probability measure.

Consider a partition $\mathcal{P}$ of region E into n sub-regions $\mathcal{P} = \{R_1, \ldots, R_n\}$. We impose $\sigma(R_i) > 0$ for all regions, then it holds $\sigma_\perp(R_i) > 0$.

We will use a probability measure P on E, such that $P(R_i) \propto \sigma_\perp(R_i)$, for all regions R_i on the partition. Thus the probability for placing a sample inside one of those regions is proportional to its $\sigma_\perp$ measure. The value of P is undefined for regions R_i' which are strictly included in a region R_i. In this case, its measure will be proportional to solid-angle measure, that is $P(R_i') \propto \sigma(R_i)$. More generally, for any region Q we have $P(Q \cap R_i) \propto \sigma(Q \cap R_i)\sigma_\perp(R_i)$.

If we include all the necessary normalization constants, we obtain (for all regions $Q \in E$ and all $i \in \{1, \ldots, n\}$) the following property:

$$P(Q \cap R_i) \;=\; \frac{\sigma(Q \cap R_i)}{\sigma(R_i)} \, \frac{\sigma_\perp(R_i)}{\sigma_\perp(E)}$$

as any differential area in E is fully included in just one of the regions R_i, then we can differentiate (with respect to $\sigma_\perp$) the above equality at any $\mathbf{u} \in E$, and we obtain the definition of p

$$p(\mathbf{u}) \;=\; \frac{1}{\cos(\mathbf{u})\,\sigma(R_i)} \, \frac{\sigma_\perp(R_i)}{\sigma_\perp(E)} \tag{4}$$

where i is the unique integer such that $\mathbf{u} \in R_i$.

3.2 Bounded variance and error.

We now show that the variance associated to the above *pdf* can be bounded.

The value U can be bounded by rewriting the integral as the sum for each region in $\mathcal{P}$, and by using the definition of p (4). We obtain:

$$\begin{aligned} U &= \int_E \frac{f^2(\mathbf{u})}{p(\mathbf{u})} \, d\sigma_\perp(\mathbf{u}) \\ &= \sigma_\perp(E) \sum_{i=1}^{n} \frac{\sigma(R_i)}{\sigma_\perp(R_i)} \int_{R_i} f^2(\mathbf{u}) \cos(\mathbf{u}) \, d\sigma_\perp(\mathbf{u}) \end{aligned}$$

For any region $C' \subseteq C$, we define the function m as the maximum value of cos in that region, that is

$$m(C') \;=\; Max\{\; \cos(\mathbf{u}) \mid \; \mathbf{u} \in C' \;\}$$

and we also define the function k as:

$$k(C') \;=\; m(C')\,\frac{\sigma(C')}{\sigma_\perp(C')} - 1 \tag{5}$$

it is straightforward to show that $\sigma_\perp(C') \leq m(C')\sigma(C')$, thus it holds that $k(C') \geq 0$. By using the previous definitions, we can state the following bound for U:

$$\begin{aligned} U &\leq \sigma_\perp(E) \sum_{i=1}^{n} \frac{\sigma(R_i)}{\sigma_\perp(R_i)}\, m(R_i) \int_{R_i} f^2(\mathbf{u})\, d\sigma_\perp(\mathbf{u}) \\ &= \sigma_\perp(E) \sum_{i=1}^{n} [\,1 + k(R_i)\,] \int_{R_i} f^2(\mathbf{u})\, d\sigma_\perp(\mathbf{u}) \\ &= U_0 + \sigma_\perp(E) \sum_{i=1}^{n} k(R_i) \int_{R_i} f^2(\mathbf{u})\, d\sigma_\perp(\mathbf{u}) \end{aligned}$$

Let us define $K(\mathcal{P}) = Max\{k(R_1), \ldots k(R_n)\}$. Although $K(\mathcal{P})$ obviously depends on $\mathcal{P}$, will simply write K in what follows. We get:

$$\begin{aligned} U &\leq U_0 + K\,\sigma_\perp(E) \sum_{i=1}^{n} \int_{R_i} f^2(\mathbf{u})\, d\sigma_\perp(\mathbf{u}) \\ &= U_0 + K\,\sigma_\perp(E) \int_{E} f^2(\mathbf{u})\, d\sigma_\perp(\mathbf{u}) \\ &= (\,1 + K\,)\, U_0 \end{aligned}$$

By using the previous bound, and the equality $V = (U - I^2)/n$, we have:

$$V \;\leq\; V_0 + K \left[V_0 + \frac{I^2}{n} \right]$$

where $V_0 = (U_0 - I^2)/n$. Note that V_0 is the smallest possible value for the variance V. Thus we have obtained a bound for the variance in terms of K and the minimum variance V_0.

Variance itself is not always a good measure of error. We can use another alternative measures. In the context of realistic rendering, an already proposed error metric is the fraction of pixels in a image whose percentual error with respect to the reference solution is above a fixed value.

If we obtain a result I', the percentual error is $|I' - I|/I$ (when $I > 0$). If this value is higher than a previously established threshold a, we will say that the result is erroneous. If $I = 0$ then the error is 0 because all unbiased estimators of I will be 0, as function f is non-negative everywhere.

When using Monte-Carlo algorithms we cannot know *a priori* when I' will be erroneous, but we can give the probability for this to happen. Moreover, if the variance is

bounded, we can also bound this probability. If we assume that $I > 0$, the Chebychev inequality [6] (p.26) states that, $\forall x > 0$ it holds:

$$\text{Prob}\{\,|I' - I| \;\geq\; x\} \;\leq\; \frac{V}{x^2}$$

By manipulating the above inequality, we obtain a bound for the probability for a result to be erroneous:

$$\text{Prob}\left\{\frac{|I' - I|}{I} \;\geq\; a\right\} \;\leq\; \frac{1}{n a^2}\left[(1+K)\frac{U_0}{I^2} - 1\right]$$

With the above inequality, we see that the probability for the result to be erroneous can be decreased when K decreases, down to a minimum value, which happens when $K = 0$. This value depends on n, I and U_0.

4 Sample Positioning Algorithm.

With the above results it is straightforward to derive an adaptive algorithm which ensures bounded variance. Our goal is to place a set of n random samples inside a given region E, in such a way that U is below $(1+K)U_0$, for any fixed value of K. Region E and values n and K are inputs for the procedure. The output is the set of samples $\{\mathbf{u}_1,\ldots,\mathbf{u}_n\}$, and the probabilities of those samples $\{p(\mathbf{u}_1),\ldots,p(\mathbf{u}_n)\}$

Region E is the projection of a planar triangle. For these kind of regions, we can easily compute the solid angle, $\sigma(E)$, by using the internal angles of the spherical triangle [2]. The point to patch form factor, $\sigma_\perp(E)$, is obtained by using the analytical expression of the form-factor [3]. More details about this can be read in subsection 4.2. We can also compute the maximum value of the cos function inside the region, $m(E)$, by using a procedure detailed later, in subsection 4.3. As we can obtain $\sigma, \sigma_\perp$ and m, we can easily get $k(E)$ for any E, by using its definition (5).

The proposed algorithm starts by computing $k(E)$. If this value is below K, then we obtain the samples by using $P \propto \sigma$, with the method described in [2]. In this case, we set $p(\mathbf{u}_i) = 1/[\sigma(E)\cos(\mathbf{u}_i)]$. This expression is obtained after rewriting (4) with $R_i = E$.

When $k(E) > K$, we split region E in a set D of four sub-regions, named as $D = \{D_1, D_2, D_3, D_4\}$, with $D_i \subset E$. This splitting is done by halving each edge of the original triangle, and then obtaining the projection of each sub-triangle. Figure 2 shows a triangle before and after being partitioned in four.

After this we compute $\sigma_\perp(E)$ and $\sigma_\perp(D_i)$, for $i \in \{1\ldots,4\}$. The probability for placing a sample inside region D_i is proportional to p_i, where p_i is defined as $\sigma_\perp(D_i)/\sigma_\perp(E)$. One option is to assign each sample to a region independently. This leads to a valid unbiased algorithm, with the desired properties. As a result, we obtain a vector $N = \{n_1, n_2, n_3, n_4\}$ such that $n = \sum_{i=1}^{4} n_i$, and where n_i is the number of samples for region D_i. However, this raw approach can be improved, as detailed in Section 4.1

Once the vector N has been obtained, the sample positioning procedure is recursively called for each of the four sub-regions in turn. By using this algorithm, we

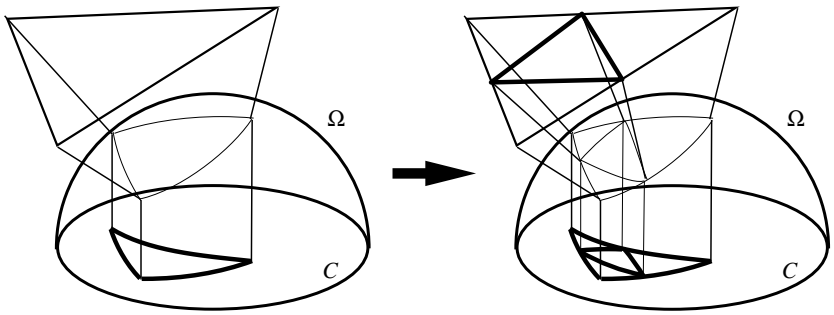

Figure 2: The four sub-triangles $D_1 \ldots D_4$ resulting from original triangle E

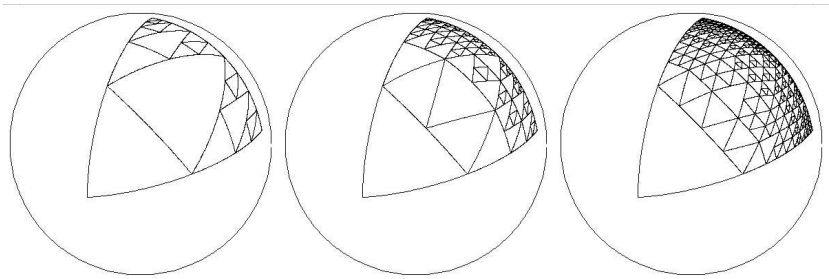

Figure 3: Three partitions, obtained with $K = 0.3, 0.15$ and 0.075 (left to right)

obtain partitions with the desired properties. In Figure 3, we can see the partitions obtained with this algorithm, for different values of K

Let us assume that sample $\mathbf{u}_j$ has been obtained from region E_i with probability $p'(\mathbf{u}_j)$ (this value is an output obtained from the recursive call). Thus the output probability $p(\mathbf{u}_j)$ returned to the caller is equal to $p'(\mathbf{u}_j)\sigma_\perp(E_i)/\sigma_\perp(E)$, because this is the probability for selecting E_i times the probability for selecting $\mathbf{u}_j$ inside E_i. The algorithm always ends because k function always approaches zero as the region it applies to shrinks.

This adaptive sampling algorithm can be implemented a function which accepts E, n and K as parameters. The function returns a pair: first component is a vector of points, and second one a vector of real probabilities. All this data can be used to estimate I, by implementing the expression for the secondary estimator I' (2). In Figure 4 you can see the code for the ADAPTIVESAMPLING function, which implements this.

4.1 Distribution of samples in sub-regions

When computing the vector $N = \{n_1, \ldots, n_4\}$ it is possible that some sub-regions are not sampled at all, and even all the samples are placed in a single sub-region of the current region.

```
ADAPTIVESAMPLING ( E : Triangle, n : Natural, K : Real )

if k(E) < K then
    {u_1,...,u_n} := SolidAngleSampling ( E )
    for i in {1,...,n}
        p_i := 1/[σ(E) cos(u_i)]
    return [{u_1,...,u_n},{p_1,...,p_n}]
else
    D := SplitTriangle ( E )
    N := DistributeSamples ( E , D )
    j := 1
    for i in {1,2,3,4}
        if n_i > 0 then
            s := j + n_i - 1
            [{u_j,...,u_s},{p_j,...,p_s}] := AdaptiveSampling ( D_i , n_i , K )
            for t in {j, j+1,...,s}
                p'_j := p_j [σ_⊥(D_i)/σ_⊥(E)]
            j := j + n_i
    return [{u_1,...,u_n},{p'_1,...,p'_n}]
```

Figure 4: Code for function AdaptiveSampling

A better option, which always reduces variance, is to ensure that at least a minimum number of samples is placed in each sub-region. This minimum number is defined as:

$$n_i \;=\; \text{floor}(p_i n)$$

where *floor* is the function which yields the highest integer smaller or equal to its argument. At least n_i samples are assigned to region E_i, but when $\sum_i n_i < n$, it is still necessary to assign all the remaining samples. To keep the algorithm unbiased, the probability for assigning one of the additional remaining samples to region E_i is made proportional to the value q_i, where the vector q is defined as follows:

$$q_i \;=\; \begin{cases} p_i n - n_i & \text{if } p_i n \;>\; n_i \\ 0 & \text{otherwise} \end{cases}$$

Thus a region i is selected with this probability, and then n_i is increased in 1. If $\sum_i n_i$ is still smaller than n, the algorithm is repeated, until each sample has been assigned to exactly one sub-region. Since we only have four sub-triangles, this loop is repeated three times at most.

Note that the probability for a single sample being assigned to region E_i is still $\sigma_\perp(E_i)/\sigma(E_i)$, thus the algorithm is unbiased. We have changed just the distribution of the vector $\{n_1, n_2, n_3, n_4\}$, making some combinations unable to happen (in fact, we avoid all combinations such that some region i receives less than $\text{floor}(p_i n)$ samples).

In Figure 4.1 you can see the pseudocode for the DistributeSamples function, which implements the algorithm described in this subsection.

DISTRIBUTESAMPLES ($E, D_1, \ldots, D_4$: Triangle, n : Natural)

```
for i in {1,2,3,4}
    p_i := σ⊥(D_i)/σ⊥(E)
    n_i := floor(p_i n)
while n_1 + n_2 + n_3 + n_4 < n
    for i in {1,2,3,4}
        q_i := max( 0 , p_i n − n_i )
    i := RandomSelect ( q_1, q_2, q_3, q_4 )
    n_i := n_i + 1
return {n_1, n_2, n_3, n_4}
```

Figure 5: Code for function DistributeSamples

4.2 Computation of σ and $\sigma_\perp$

When we create a new triangle E, we need to compute the values $\sigma(E)$ and $\sigma_\perp(E)$. This operation should not take a long time, because it is repeated very often. Let $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2$ be the three normalized vertices of a triangle E. We call $\mathbf{n}_i = \mathbf{v}_i \times \mathbf{v}_{i\oplus 1}$ to the three normals to each edge. The arc length α_i of each edge is the angle between $\mathbf{v}_i$ and $\mathbf{v}_{i\oplus 1}$, that is: $\alpha_i = \arccos(\mathbf{v}_i \bullet \mathbf{v}_{i\oplus 1})$. The internal angle between edge i and edge $i \oplus 1$ is called β_i. It can be shown that $\beta_i = \pi - \arccos(\mathbf{n}_i \bullet \mathbf{n}_{i\oplus 1})$. With these definitions, we have:

$$\sigma(E) \;=\; \left[\sum_{i=0}^{2} \beta_i\right] - \pi \qquad\qquad \sigma_\perp(E) \;=\; \frac{1}{2}\left[\sum_{i=0}^{2} n_{iz}\,\alpha_i\right]$$

where n_{iz} is the z component of $\mathbf{n}_i$

When we split a triangle in four, we can save computations by using the relations between the new sub-triangles $\{D_0, \ldots, D_3\}$ and the original one E. Let us assume that D_0 shares a vertex $\mathbf{v}_0$ with E, and call α'_i (resp. β'_i) the arc lengths (resp. internal angles) of D_0, and $\mathbf{n}'_i$ to the normals of D_0. We have $\alpha'_0 = \alpha_0/2$, $\alpha'_2 = \alpha_2$, $\beta'_0 = \beta_0$, $\mathbf{n}'_0 = \mathbf{n}_0$, and $\mathbf{n}'_2 = \mathbf{n}_2$. This implies that computation of $\sigma(D_0)$ and $\sigma_\perp(D_0)$ can be done in short time. Similar relations holds for the other three subtriangles.

4.3 Computation of m

For any region E, the value $m(E)$ is the maximum of the cos function over E. Evaluation of m for triangles is needed for the computation of function k. The maximum value of cos for the whole circle S is obviously 1, and happens on the origin $(0,0)$. Thus if $(0,0) \in E$ then $m(E) = 1$. Checking whether this condition holds is straightforward.

It can be shown that when $(0,0) \in E$ then the sign of the z component of all $\mathbf{n}_i$ is the same (positive or negative, depending on the orientation of the three vertices).

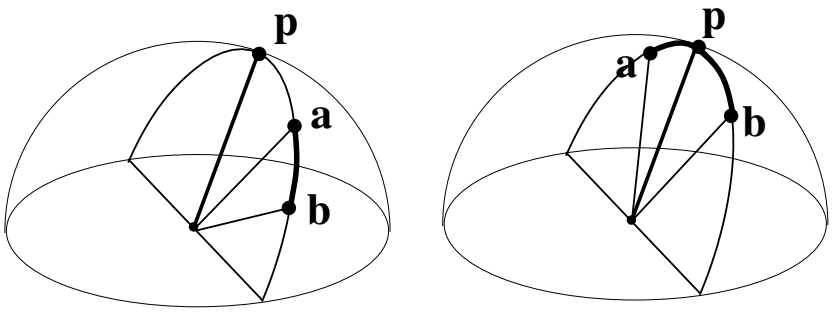

Figure 6: Two cases when computing the maximum z value for an arc.

When $(0,0) \notin E$, the point with maximum z is the boundary of E. In this case, we get the maximum z of the three arcs, and after this we select the maximum of them. Two cases arise when computing the maximum of an arc: either this maximum is one of the two vertex which define (we call this **a** and **b**) the arc (Figure 6, left) or it is one point in the interior of the arc (Figure 6, right). Second conditions holds only when the arc includes the vector **p** (see Figure 6), which is the point with max z for the whole circumference containing the arc. This can be checked by using cross products. When the z sign of $\mathbf{a} \times \mathbf{p}$ is different to the z sign of $\mathbf{b} \times \mathbf{p}$, the max is **p**, and in any other case, the max is either **a** or **b**.

5 Results.

In Figure 7 you can see three different distributions of samples one the unit-radius circle C, and for an example triangle E. The number of samples is $n = 900$ in all the cases.

First one (left) has been obtained by using $P \propto \sigma$ and with stratified sampling [2]. Region E is divided in 30×30 sub-regions, and a sample is placed in each of those regions. By using stratification, the distribution we obtain is quite regular (it has low discrepancy). However, samples density is clearly higher near the circumference, because the σ measure does not depends on the cosine function.

The second distribution in Figure 7 has been obtained by using adaptive sampling with $K = 0.1$. Now, samples density is more uniform with respect to area, as measured on the circle. Unfortunately, the sub-regions obtained from the (recursive) splitting process are very big. As samples are placed at random inside each of those regions, the distribution of distances between samples is not regular. The we obtained some large zones with no samples, and other with clusters of very near samples (high discrepancy).

Finally, the last distribution in that Figure (right) has been obtained also by using adaptive sampling, but with $K = 0.001$. The distribution is again uniform with respect

to area measure, but now the distances between samples are also uniform. This is the best of the three distributions.

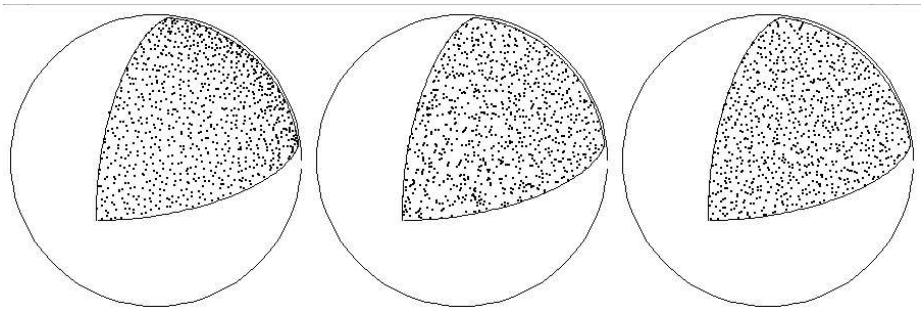

Figure 7: Stratified sampling with $P \propto \sigma$ (left), adaptive sampling with $K = 0.1$ (center) and $K = 0.001$ (right)

The adaptive sample placement algorithm has been used for irradiance computation in a sample scene, which is depicted in perspective view in Figure 8. The source is a vertical triangle, and the receiver is the horizontal planar rectangle. An array of rectangles partially occludes the light from source to receiver. Radiance emitted from the source is not constant: it follows a rectangular pattern. The irradiance function on the receiver is complicated to compute because the pattern on the source interacts with the occluders. Thus we can resort to Monte-Carlo sampling on the triangle.

In Figure 9 we can see three images of the irradiance function on the receiver, as seen from the top, with the occluders removed. First one (top) has been obtained by using $P \propto \sigma$, with 36 samples per pixels. Second one (middle) has been obtained the same way, but using stratified sampling (the spherical triangles where divided in 6×6 regions). Third one (bottom) has been obtained with adaptive sampling, also with 36 samples per pixel, and $K = 0.001$.

We can see that the second image show less noise than first one, thus stratified sampling really pays off, with the same number of samples. However the third image shows less noise than the second one, with the same number of samples. Thus variance has been reduced by using adaptive sampling, as predicted.

Note that the adaptive sampling algorithm usually takes longer than stratified sampling, because of the need to split the triangles. However, in most applications the bottleneck is in the evaluation of $f(\mathbf{u}_i)$ for any given $\mathbf{u}_i$, which usually involves computation of visibility in complex scenes. This means that, in order to reduce noise, it is better to improve the samples distribution before starting to increase the number of samples. In the case that evaluation of f is fast enough, stratified sampling with an increased number of samples may be preferable to adaptive sampling.

6 Implementation

The described algorithm has been implemented in C++, and tested on a Linux-based PC with the GNU compiler. The source program can be downloaded at

`http://giig.ugr.es/~curena/abstr/cgf99`

The class `ETriangle` holds all relevant parameters about a triangle (vertices, solid angle, form factor), and implements adaptive sampling. An `ETriangle` is constructed by giving its three vertices and a value for K. These vertices should have non-negative Z coordinate. Their coordinates are relative to the local coordinate system, whose origin is in the target point $\mathbf{p}$, and whose Z-axis is aligned with the normal at $\mathbf{p}$.

A call to `GetSamples` yields a set of samples, by giving the number of those we wish. This function is recursive, and implements the described sampling procedure. For each sample, we obtain a `struct` containing its position and probability.

7 Acknowledgements

This work has been funded by grant number TIC98-0973-C03-01 from CICYT (Science and Technology Committee of the Spanish Government). We wish to thank the comments and suggestions from the anonymous reviewers of earlier versions of this paper.

References

[1] J. Arvo. Application of irradiance tensors to the simulation of non-lambertian phenomena. In *Computer Graphics Proceedings, Annual Conference Series*, pages 335–342. ACM Siggraph, 1995.

[2] J. Arvo. Stratified sampling of spherical triangles. In *Computer Graphics Proceedings, Annual Conference Series*, pages 437–438. ACM Siggraph, 1995.

[3] D.R. Baum, H.E. Rushmeier, and J.M. Winget. Improving radiosity solutions through the use of analitically determined form factors. *Computer Graphics*, 23(3):325–334, 1989. ACM Siggraph 89 Conference Proceedings.

[4] B. Bian. Hemispherical projection of a triangle. In *Graphics Gems III*, pages 314–317. Ed. AP Professional, 1995.

[5] K. Itô, editor. *Encyclopedic Dictionary of Mathematics*, volume 1, chapter Elliptic Functions, pages 526–530. The MIT Press, 1993.

[6] M.H. Kalos and P.A. Whitlock. *Monte-Carlo Methods. Volume I: Basics*. Jhon Wiley & Sons, 1985.

[7] S.N. Pattanaik and S.P. Mudur. Computation of global illumination by monte carlo simulation of the particle model of light. In *Proceedings of the 3rd Eurographics Workshop on Rendering*, pages 71–83, 1992.

[8] P. Shirley, C.Y. Wang, and K. Zimmerman. Monte-carlo techniques for direct lighting calculations. *ACM Transactions on Graphics*, 15(1):1–36, 1996.

[9] C. Ureña and J.C. Torres. Improved irradiance computation by importance sampling. In *Rendering Techniques'97 (Proceedings of the 8th EG Workshop on Rendering)*, pages 275–284. Springer Verlag, 1997.

[10] E. Veach and L.J. Guibas. Optimally combining sampling techniques for monte-carlo rendering. In *Computer Graphics Proceedings, Annual Conference Series*, pages 419–428. ACM Siggraph, 1995.

[11] C. Wang. Physically correct direct lighting for distriution ray-tracing. In *Graphics Gems III*, pages 307–313. Ed. AP Professional, 1995.

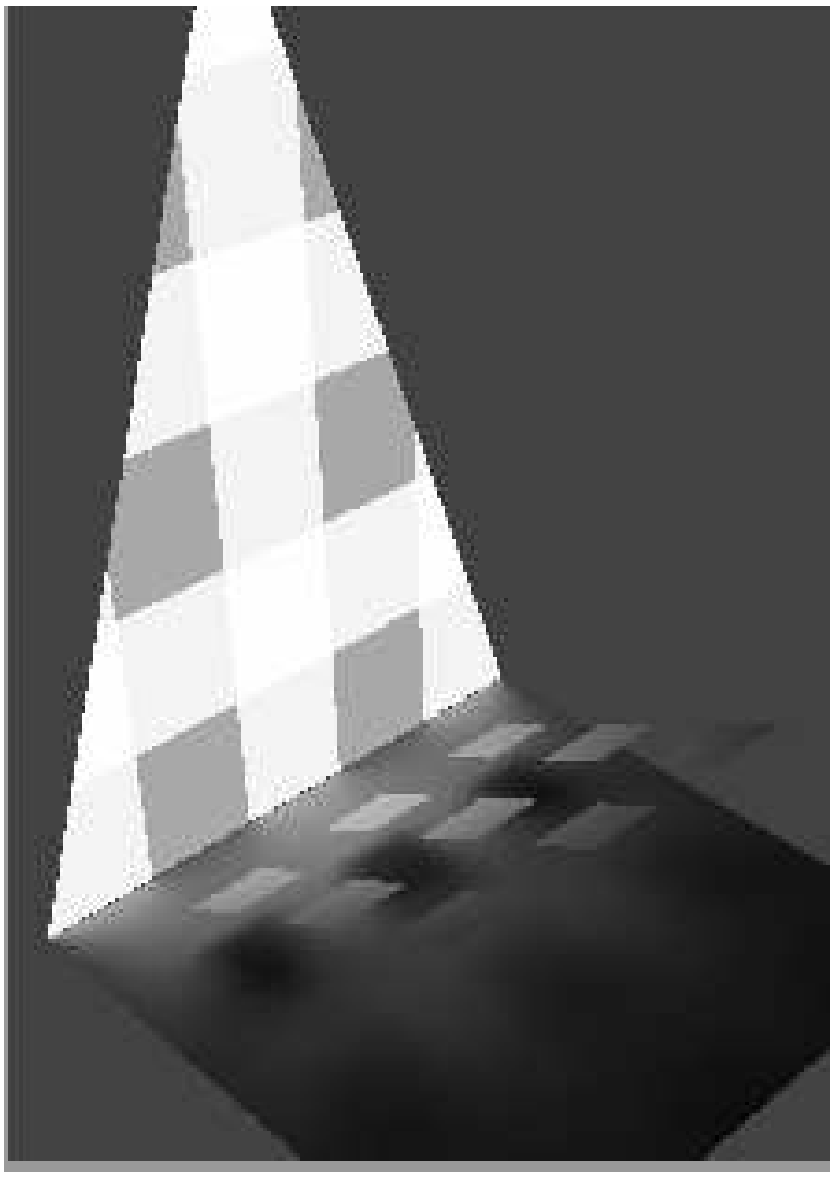

Figure 8: Perspective view of sample scene

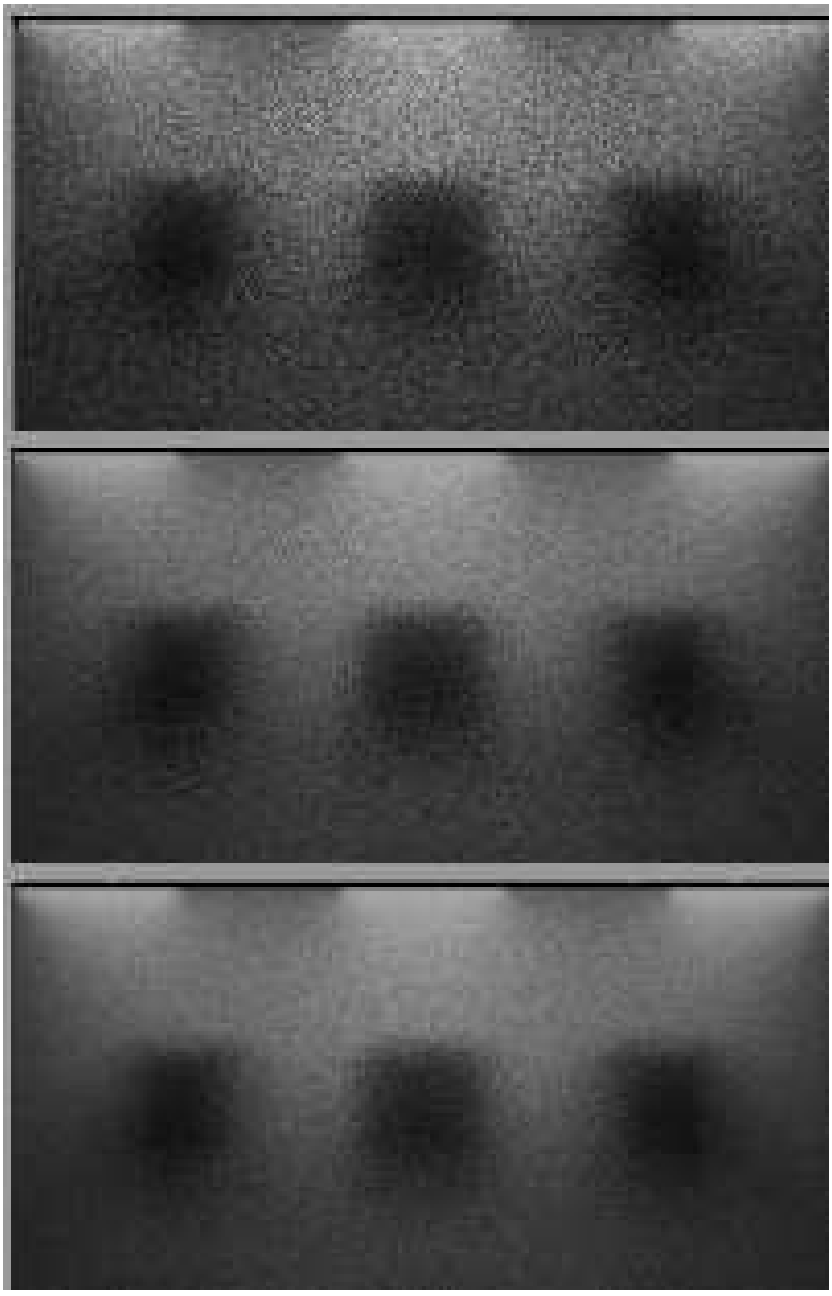

Figure 9: Results of irradiance computation for the horizontal rectangle