EUROGRAPHICS 2001 / A. Chalmers and T.-M. Rhyne
(Guest Editors)

Volume 20 (*2001*), *Number 3*

An Adaptive Method for Indirect Illumination Using Light Vectors

Xavier Serpaggi† and Bernard Péroche‡

Abstract

In computer graphics, several phenomema need to be taken into account when it comes to the field of photo-realism. One of the most relevant is obviously the notion of global, and more precisely indirect, illumination. In "classical" ray-tracing if you are not under the light, then you are in a shadow. A great amount of work has been carried out which proposes ray-tracing based solutions to take into account the fact that "there is a certain amount of light in shadows". All of these methods carry the same weaknesses: high computation time and a lot of parameters you need to manage to get something out of the method. This paper proposes a generic computation method of indirect illumination based on Monte Carlo sampling and on the sequential analysis theory, which is faster and more automatic than classical methods.

1. Introduction

The field of 3D computer graphics has been investigated since the early 1980's. Since then, a lot of works has been done to reproduce as well as possible the outside world. One important step was to manage to find a suitable model to light behaviour. Two different ways of research were developed, each of them trying to reach this same goal.

The first one is called *ray tracing* and is able to produce high quality pictures with stunning specular effects, but no indirect lighting. Unfortunately, because of these specular surfaces, each time you move the camera, you need to recompute the whole picture.

The other one is called *radiosity* and renders diffuse surfaces very aptly. At the cost of a very long pre-process phase, you are able to navigate interactively into a purely lambertian virtual environment and there is no need to recompute anything between frames.

Both of these methods have pros and cons, and both are still used nowadays. Radiosity has found solutions to take specular effects into account[14], and of course, ray-tracing is now able to correctly render scenes with indirect illumination[7, 19]. Taking all these kinds of surfaces into account is almost the same as taking all kinds of illuminations into account. This is what is called the global illumination problem. To solve this problem, several techniques are used and one of the most widely spread is Monte Carlo sampling.

In this paper we present a generic computation method of global illumination in ray-tracing, which permits to reduce the time spent to render a single picture. To achieve this, we use the vector based approach presented by Zaninetti et al.[19], combined with sequential analysis techniques introduced by Wald[1], which helps us to compute indirect illumination, only where it is relevant. In section 2 we will present a short survey of techniques usually used to solve this problem. In section 3 we will discuss the original Light Vectors method as presented by Zaninetti et al., in order to show its advantages and its weaknesses. Then, in section 4 we will present the tool used to improve this algorithm: sequential analysis. Section 5 will present our method in depth, with implementation issues in section 6. Then, some results will be given in section 7 before we conclude in section 8.

2. Previous Work

The very first work about global illumination was done by Goral et al.[4] in 1984. The principle, taken from the field of heat transfer and called *radiosity*, aimed at finding an equilibrium state of light transfer between all the components of the scene to be rendered. These first algorithms, which were only able to render scenes with lambertian-only surfaces and

† LISSE, École Nationale Supérieure des Mines de Saint-Étienne, 158, cours Fauriel, 42023 Saint-Étienne Cedex 2

‡ LIGIM, bât. 710, université Claude Bernard, Lyon 1, 43, Boulevard du 11 novembre 1918, 69622 Villeurbanne Cedex

demanded a huge pre-processing phase to initialize the computation, enabled the user to interactively navigate through the scene.

In this paper we are more interested in *ray tracing*[3] algorithms, and the first one in this field – thus, not limited to lambertian surfaces – able to solve the global illumination problem, was proposed in 1986 by Kajiya[6]. He introduced *the rendering equation* as well as *path tracing*. The former is a recursive analytic equation that expresses light interactions between the elements of the scene, and the latter is a numerical based method, using Monte Carlo sampling, that solves this equation. The principle is to gather incoming radiance on a pixel with the help of paths leaving the camera through this pixel and randomly traced in the scene. This method was a major improvement thanks to its ability to take every kind of materials into account. Its main drawback is the poor quality of the pictures obtained. Because of Monte Carlo sampling, there is a lot of noise, and if you want to get rid of it, you need to trace a lot of paths per pixel, which leads to high computation times.

Other physically based solutions were proposed, including *bi-directional path tracing*[11, 13, 15, 16]. This method fills in some of the gaps left by Kajiya's. Here, paths are traced simultaneously from the camera and from the light sources. Each of them interacts with the objects in the scene. Then, visibility wise connections are made between paths from the camera and paths from the light sources to let the energy flow. This method leads to a better solution than Kajiya's in the sense that it is less noisy for the same amount of computation. However, high quality pictures request a great number of paths to be traced.

A new version of Monte Carlo sampling was introduced by Veach et al. in 1997, called *Metropolis light transport*[18]. Each light path between the camera and the light sources is mutated into several other paths according to different strategies. Then, each mutation is accepted or rejected according to its final contribution to the picture. The higher the contribution, the more the path is likely to be elected to be used in the final picture. The advantages of this method, compared to other Monte Carlo based methods, is that it is unbiased and able to take a large variety of physical phenomena into account.

All of the methods presented above had been thought to be models of what actually happens when light interacts with matter. This had been done because of the computers used at these times. But nowadays, computers grow more and more powerful, so why not simply try to recreate the real behaviour of light? We have known for long now that light can be represented by particles called photons, navigating and interacting with objects, to finally give us a picture of what reality looks like. At first glance, such a representation seems to lead to algorithms much simpler than the ones used and, undoubtedly, physically correct. Unfortunately, computers are yet not powerful enough, to think to implement such methods. Only few methods are proposed, and the most famous is Jensen's *photon map*[17]. Here Monte Carlo techniques and photons particles are put together to create the photon map, which produces good results, but requires a lot of memory and computation time.

A recent method presented by Zaninetti et al.[19] and based upon Ward's[7, 10], Arvo's [12] and Jensen's[17] works, proposes a vector approach to solve the global illumination issue. It combines Monte Carlo sampling and vector representation of light to obtain pictures with a low level of noise, avoiding high computation times. To reach such a goal, each kind of lighting (direct, indirect and caustic) is replaced by *light vectors* according to its class. Thus, under certain criteria, it is possible to interpolate radiance where other methods recompute it. This gives fast results and, thanks to the interpolation process, less noisy. But the time needed to compute a single picture is still too high for the method to be used in an interactive scheme, as we wish to.

In the following sections we introduce improvements to the Light Vector method presented by Zaninetti et al.[19] (using Wald[1] work and encouraged by the results given by Maillot[9]) which only concerns indirect illumination.

3. Light Vectors

A *Light Vector (LV)* is used to replace, whenever possible, the illumination showering an area, by a virtual light source, for each point it is computed at. Then, under certain circumstances, you can interpolate illumination for neighbouring pixels rather than recompute it. The major effect can be seen as shorter time is necessary to compute a single picture. *LVs* have been divided into three different classes, depending on the kind of illumination they represent:

- *Direct Light Vectors (DLVs)* replace the light that hits only one object before being caught by the camera ($L(S|D)E$ paths),
- *Caustic Light Vectors (CLV)* replace the light that hits, at least first, one specular then one non-specular object before reaching the camera (LS^+DE paths), and
- *Indirect Light Vectors (ILV)* replace the light that hits at least two non-specular objects before the camera ($LD(D|S)^+E$ paths).

(L,S,D,E) regular expressions were defined by Heckbert[8]: L represent the light sources, E is the eye, S and D are respectively specular and diffuse surfaces, and X^+ means one or more reflections on surface of type X.

This ray tracing method (*LVs*), although it is based on Monte Carlo sampling, is by far less costly and gives smoother results than classical methods. What follows concentrates on *Indirect Light Vectors* only.

The method to compute these *ILVs*, as described by Zaninetti[19], is divided into two parts.

Computation of a seed: *ILVs* are computed, as described in the following section, on a given number of pixels over the picture and put into a kd-tree for further reference (interpolation process).

Rendering of the picture: The indirect illumination is evaluated for the remaining pixels. Here, two cases arise. Whether the indirect illumination can be interpolated using the *ILVs* computed during the first phase and there is nothing to be done, or the interpolation criteria cannot be satisfied and a new *ILV* is computed from scratch before beeing inserted into the kd-tree.

As we can see, complete computations of *ILVs* as described below, are only done during the first phase and when criteria are not fulfiled in the interpolation process.

3.1. Computing *ILVs*

The rendering equation (1), expressed in terms of radiance, shows that the radiance emitted by a point x of the scene in direction $\vec{\omega}_r$ depends on self-emitted radiance L_e of this point, on incoming radiance L_i to this point from all directions $\vec{\omega}_i$ of half-space Ω, and on the *bidirectional reflectance distribution function* f_r. The latter is a function which models the behaviour of the material at point x when seen through direction $\vec{\omega}_r$ and exposed to light coming from direction $\vec{\omega}_i$. From now on, we will use the *BRDF* acronym for this function.

$$L_r(x,\vec{\omega}_r) = L_e(x,\vec{\omega}_r) + \\ + \int_\Omega f_r(x,\vec{\omega}_i \rightarrow \vec{\omega}_r) L_i(x,\vec{\omega}_i) cos\theta_i d\omega_i \qquad (1)$$

If we split down this equation, three different types of illumination will appear (we do not show the caustic part of illumination on purpose):

$$L_r(x,\vec{\omega}_r) = L_{dir}(x,\vec{\omega}_r) + L_{spec}(x,\vec{\omega}_r) + L_{ind}(x,\vec{\omega}_r)$$

Here, $L_{dir}(x,\vec{\omega}_r)$ is the direct-only part of the illumination leaving point x through direction $\vec{\omega}_r$. $L_{spec}(x,\vec{\omega}_r)$ is the perfectly specular part and $L_{ind}(x,\vec{\omega}_r)$ represents the indirect component, the one we are interested in.

A discrete version of the indirect part of the illumination, in the rendering equation, may be expressed as follows:

$$L_{ind}(x,\vec{\omega}_r) \approx \\ \approx \frac{2\pi}{MN} \sum_{j=0}^{M-1} \sum_{k=0}^{N-1} f_{rd}(x,\vec{\omega}_{j,k} \rightarrow \vec{\omega}_r) L_i(x,\theta_j,\phi_k) cos\theta_j \qquad (2)$$

Where M and N represent the subdivision, at constant solid angle, of the hemisphere centered at point x; M divides the azimuthal angle and N divides the zenithal angle. In equation (2), f_{rd} represents the non-perfectly specular part of the *BRDF*.

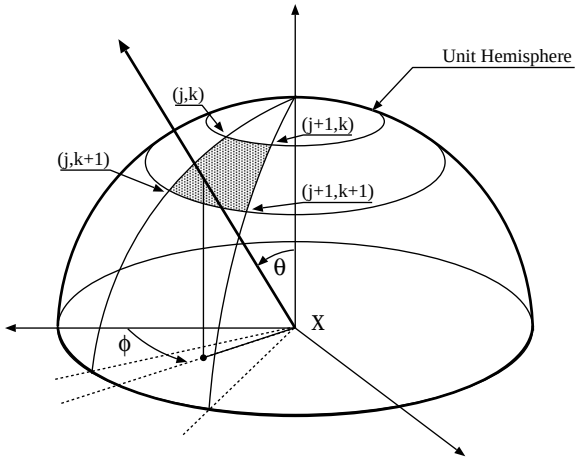


Figure 1: *Sampling of the hemisphere with uniform solid angle.*

As shown by figure 1, each (j,k) cell is a shooting window for a ray which will give us information on incoming indirect radiance at point x within direction (θ_i,ϕ_k).

Once the process has been through all the cells of the hemisphere, it is possible to give average values of the radiance P_0 and direction $\vec{D}_0$ of incoming indirect illumination at point x. These two news elements are used to find the irradiance P and direction $\vec{D}$ of the virtual light source that, when seen through direction $\vec{\omega}_r$ and attenuated by the *BRDF* at point x, gives the same radiance that the original illumination. P and $\vec{D}$ actually represent one *ILV* , as expressed in equation (3).

$$L_{ind}(x,\vec{\omega}_r) = P f_{rd}(x,\vec{D} \rightarrow \vec{\omega}_r) \qquad (3)$$

3.2. Pros and Cons of the Original Method

The method summed up above is very efficient when compared to classical Monte Carlo methods. First, it renders pictures that are less noisy, but it also permits to save a lot of computation time (see table 1 in section 7 for a comparison between the two methods). However, we believe that some points need improving and amongst them:

- The way the seed is computed. This seed is mandatory for the interpolation process to go on smoothly.
- The criteria for the interpolation need to be refined to take some effects, such as secondary light sources, into account.

- The number of rays traced to compute a single *ILV* which is always the same, regardless of illumination distribution.

The last problem is the one we are interested in. This paper introduces a new method which finds a solution to trace fewer rays to compute one *ILV* , leading to shorter computation times. The way rays are traced is not randomly selected and depends on illumination distribution. The information gathered by the rays needs to be significant and usable by our new algorithm, that is why we have chosen to settle our optimisation upon homogeneity (see section 5). The root hypothesis is that some areas, seen by point x, contain less information than others, thus do not need to be sampled as precisely to give relevant information. This will be formalized using statistical methods gathered under an unique name: *sequential analysis*[1], which will allow us to *incrementally* classify each area as homogeneous or not.

4. Sequential Analysis

Sequential Analysis can be seen as a set of statistical methods whose particularity is to be able to give a result with an indefinite number of observations. This greatly differs from classical methods in which the number of samples has to be given before the beginning of the process. Whether the test goes on or not, depends only on the set of the previous obervations. So, *on an average*, this method requires fewer samples than others to give a relevant result on the nature of the studied population.

The proposed technique is only a branch of Wald's sequential analysis[1]: the *sequential probability ratio test* (or *SPRT*). This test, compared to the best classical methods based on a fixed number of samples, gives better results, saving up to 50% of observations to reach the same conclusions.

4.1. Sequential Probability Ratio Test

In his book, Wald introduced this test as "the sequential probability ratio test for testing a simple hypothesis H_0 against a single alternative H_1". This may also be expressed differently:

Let us consider a population P which is known to be composed of only two kinds of individuals T_0 and T_1. Let us call p the unknown percentage of individuals belonging to kind T_1. Our goal is to figure out if the population is rather of kind T_0 or of kind T_1.

To do so, we choose two relevant probabilities p_0 and p_1 ($0 < p_0 < p_1 < 1$), that will help us decide of which kind the population is: the population will be classified as T_0 if $p = p_0$, or as T_1 if $p = p_1$. To settle this, we choose two hypotheses H_0 and H_1. The former is the hypothesis that $p = p_0$, and the latter is the hypothesis that $p = p_1$.

If we consider an individual x, randomly chosen amongst the population P, we can call $f(x, p)$ the function describing its probability of appearance within P. So, $f(x, p_0)$ represents this probability when H_0 is true, and $f(x, p_1)$ represents this probability when H_1 is true.

With this background we can state that the probability of one randomly chosen sample $(x_1, \cdots, x_m)$ (m is an integer strictly greater than 0 and $x_1, \cdots, x_m$ are independent), is equal to:

$$p_{0m} = f(x_1, p_0) \cdots f(x_m, p_0)$$

when H_0 is true, and to

$$p_{1m} = f(x_1, p_1) \cdots f(x_m, p_1)$$

when H_1 is true.

If we choose two constants A and B, with $0 < B < A$, the *SPRT* may be written as:

– if $\frac{p_{1m}}{p_{0m}} \geq A$, the test ends with hypothesis H_0 rejected (so, hypothesis H_1 is accepted) and the population is classified as T_1,
– if $\frac{p_{1m}}{p_{0m}} \leq B$, the test ends with hypothesis H_0 accepted, and the population is classified as T_0,
– else the test goes on, taking a new individual into account.

As we can see, each time the sample is modified, the ratio $\frac{p_{1m}}{p_{0m}}$ is recomputed and compared to A and B.

There are two interesting points concerning the *SPRT*. The first one has already been mentioned, and is the size of the sample that may vary depending on the population. The second one is that Wald proved that the probability for the test to end is equal to 1 under certain circumstances (see section 4.2). This way, we are sure to always have a solution to our problem.

4.2. Defining Risks

The two constants A and B defined in the previous section are not arbitrary, but in relation with the two hypotheses H_0 and H_1. Using the *SPRT*, the nature of the population is found by looking at only a few individuals. So, we are likely, from time to time, to make mistakes. We want to have a control over these mistakes, so we try to limit them by setting an error probability for each hypothesis. These new probabilities are called risks and are defined as follows:

α is the probability to reject H_0 when this hypothesis is true, and
β is the risk to accept H_0 when H_1 is true.

As we can see, these risks are not symetrical. This is due to the interval of uncertainty between A and B. Within this interval we cannot clearly determine the class of the population, and we need to update our sample.

The link between A, B, α and β is given by Wald and is the condition under which the test ends with probability 1:

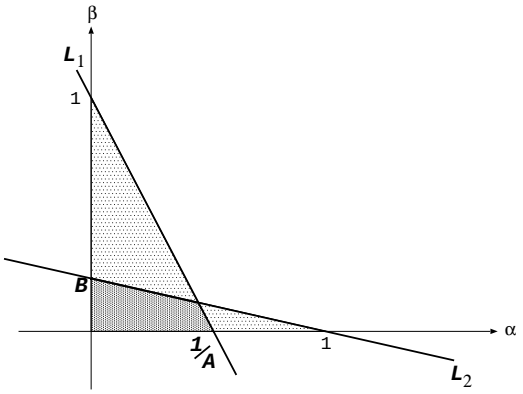


Figure 2: *L_1 and L_2 are strait lines defined respectively by equations $\alpha A = 1 - \beta$ and $\beta = B(1-\alpha)$. The heavy shaded region defines the area in which the test is satisfied. The light shaded regions define areas of uncertainty.*

$$A = \frac{1-\beta}{\alpha} \tag{4}$$

and,

$$B = \frac{\beta}{1-\alpha} \tag{5}$$

These constants can be gathered together on a diagram, as shown by figure 2.

5. A New Method to Compute Indirect Light Vectors

What we are trying to do here, is to improve the way *ILVs* are computed and, as a result, the *quality* of the pictures. There may be two sides to this improvement: on one hand, the computation process has to be shortened in time and, on the other hand, the results of this computation must be sharper in respect of the correctness of the rendered pictures. Both of these improvements have to fulfil the condition that the algorithm has to be as generic as possible. This way, we do not use any kind of optimisation, and are able to render scenes regardless of the geometric description or the *BRDF* used.

In ray tracing, an obvious way to reduce computation time is to reduce the number of rays traced. But, as we want to keep information consistent, these few rays have to be traced, as much as possible, where it counts. This allows us both to reduce the time needed to compute a single *ILV* and to obtain a better information on the incoming illumination. Then, what we do here is to try to reduce the number of rays traced for each hemisphere, reducing the number of rays traced in the whole scene.

Now we need to define what "where it counts" means. What matters, when sampling an unknown function, is to find singularities. The intervals where the function is constant or so, need less attention. To distinguish these two kinds of intervals, we use the terms homogeneous and non-homogeneous. We will call homogeneous an area in which a percentage P_h of its sample have a radiance L_x so that $L_x \in [L_m - \varepsilon, L_m + \varepsilon]$ (with L_m the average radiance over the area, and ε an arbitrary constant).

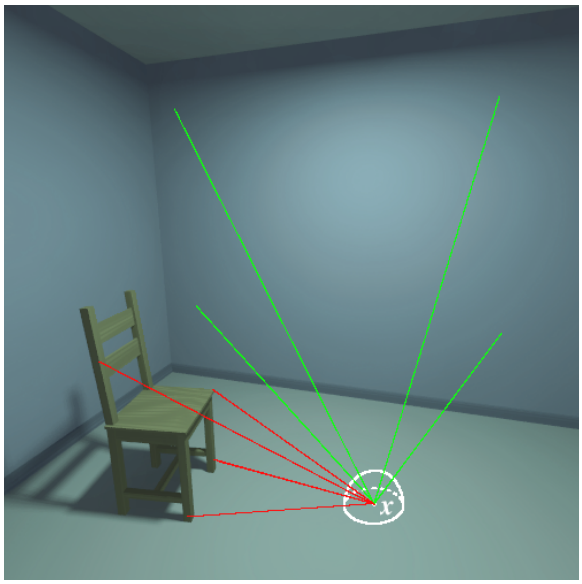


Figure 3: *All the areas "seen" by an* ILV *are not homogeneous. If we consider illumination reaching point x, the one flowing through the red tunnel is much more disrupted than the one flowing through the green tunnel.*

For example, on figure 3, we may notice that the incoming indirect radiance at x is not homogeneous: it is very chaotic within the dark grey lines, the chair interfering with its field, but very calm within the light grey lines.

Here, the idea is to use the *SPRT* to try to determine, as fast as possible, that in the latter area, radiance varies slowly, to concentrate efforts on the former.

More precisely, we do not consider the incoming radiance at x, but the reflected radiance at that point, once it has been through the *BRDF*. This is the radiance actually seen, whether by the camera, or by a previous *ILV* in the process[19], and we do not need to wonder if the shape of the *BRDF* (specular or diffuse part, peaks, ...) will alter the radiance, it is already done.

As seen in section 4.1, the *SPRT* requires a population and two hypotheses to work. The population is represented by all the reflected radiances at x and the hypotheses are:

H_0 **:** the sampled area is homogeneous (as defined above), and is classified as T_0.

H_1 : the sampled area is non-homogeneous and is classified as T_1.

The hemisphere centered at x is split, a priori, into cells having the same solid angle. Each of these cells is likely to be an area in which the *SPRT* is used.

If one of these areas is found homogeneous, it is not necessary to trace rays anymore and we can focus on the next ones. But, if we cannot figure out of what kind the area is, we need to update our sample, taking a new individual into account. This last step is repeated until we can take a decision. Eventualy, if the area is found non-homogeneous, we split it into four sub-areas, in which the *SPRT* is applied.

Every sub-area is then tested against homogeneity, and this recursive process ends only when all the areas, and sub-areas are classified T_0. Once this goal is reached the computation of the *ILV* can occur giving the direction and power of the virtual light source used to replace indirect illumination at point x. Only one *ILV* by sampled hemisphere is created, regardless of the number of cells.

5.1. Radiance Evaluation

The way *ILVs* are computed with the original method ensures the hemisphere sourrounding point x is evenly divided and each cell carries exactly one ray. Thus, the radiance computation can be done quite easily, affecting the same part of solid angle to each ray, this part being a fraction of the solid angle of the hemisphere.

Our method, thanks to its adaptive behaviour, traces several rays per cell and may lead to cell subdivision. So, it is not obvious to know which solid angle to affect to each ray. However, it is very important to know it for the radiance to be computed correctly.

A first approximation is to give rays a part of the solid angle of the cell they are in. This gives good results, but forces us to keep a very detailed structure of the subdivision through all the computation process. Moreover, at every cell's border, problems arise, each cell being unable to "see" its neighbours.

What is needed is an algorithm able to divide the hemisphere, giving each ray the maximum solid angle without overlapping. As the rays are traced randomly into each cell, this algorithm needs to be as generic as possible. We cannot use any hypothesis to try to simplify this process. That is why we have decided to build a Voronoï diagram over the hemisphere[5], each site being the intersection between a ray and the unit hemisphere centered at x. This way, the original subdivision is not taken into account leaving aside all the problems it carries.

As stated above, the diagram is built over the hemisphere and not on a projective plane. The projection from a hemisphere onto a plane is neither distance or angle conservative,

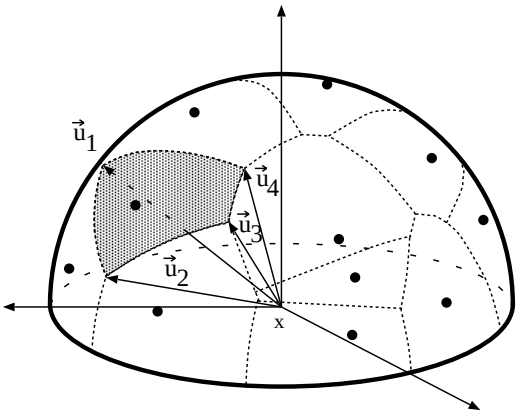


Figure 4: *Voronoï diagram over the hemisphere and vectors used to compute the associated solid angle.*

so the areas of the Voronoï cells would not have been relevant.

The diagram is built up once all the rays belonging to the *ILV* have been traced and all the cells have been found homogeneous. This allows us to get rid of the original subdivision of the hemisphere, and then all the rays are affected the right solid angle, regardless of borders between cells.

This is an a posteriori stratified sampling of the hemisphere. The stratification is simply adapted to the nature of incoming radiance and *BRDF*. This permits to take into account every kind of *BRDF*, not only analytical ones, but even real ones (after sampling).

The next step is, using the Voronoï cells, to compute the solid angle associated with each ray. If we refer to figure 4, we can state that this solid angle ω_s, is given by the well known formula[2]:

$$\omega_s = \sum_{i=1}^{N} cos^{-1}\left(\frac{(\vec{u}_i \wedge \vec{u}_{i+1})\cdot(\vec{u}_i \wedge \vec{u}_{i-1})}{\| \vec{u}_i \wedge \vec{u}_{i+1} \| \| \vec{u}_i \wedge \vec{u}_{i-1} \|}\right) - (N-2)\pi$$

6. Implementation

Applying the *SPRT* to *ILVs* computation is not straightforward. Some tunings have been necessary. That is what is exposed in this section.

6.1. Initialising

Homogeneity, as defined above, use both the average (L_m) and the max (L_{Max}) value of the radiance over the area. The value ε actually represents a percentage of L_{Max}, so the test is able to locally adapt itself to each area.

Every time a new individual is added to the sample, L_{Max} has to be modified and L_m recomputed. This is not very suitable as we want to have an incremental method. To allow

this, the values of both L_m and L_{Max} are computed using a first sample of ten individuals, randomly chosen over the area. Then, these two values are used as long as the test goes on in this area.

In the case we cannot determine which kind of area it is, two new individuals are taken into account, instead of only one. This is done to avoid to recompute p_{0m} and p_{1m} too often. A value of two seems to be a reasonable compromise between *too much recomputation* and *too much rays traced*. This is linked to the fact that A and B are chosen to describe a very narrow area of uncertainty, as exposed below.

6.2. Calibrating

Once the test has been initialized, p_{0m} and p_{1m} have to be compared with A and B, so we need to set these two constants. There are no rules to find values for them because they are to be adapted to every case. So, A and B, through α and β, have been found thanks to experimentation: we have rendered an empty room with a planar area light source at the very centre of the ceiling, modifying the values of α and β, so that the picture appears noiseless and is computed as fast as possible. The idea is that if these constants are usable for a very simple case like this one, they should be a good starting point for more complex scenes. This assertion is highly subjective, but, as exposed in section 7, seems to work quite well.

After conducting a lot of tests, we decided to keep these values: $\alpha = 0.44$ and $\beta = 0.55$. Then, equations (4) and (5) are used to find the values of A and B: $A = 1.0227$ and $B = 0.9821$.

The chosen values for α and β define a very narrow area of uncertainty as defined on figure 2. This allows the method to quickly find a result, falling in one of the terminal states, homogeneous or non-homogeneous, with a few samples.

The last constant related to our implementation of the *SPRT* is ε. The value for this constant is set to 0.25, that is 25% of L_{Max} on each side of L_m. We do think this value, although it may appear huge, can be justified by the well known hypothesis in global illumination: *indirect lighting changes slowly over a surface*.

6.3. Limiting the recursion depth

When an area is found non homogeneous, it is split up into four sub-areas in which the *SPRT* is applied. This recursive mechanism, in our implementation, is limited, in order to avoid endless computations. As, when sampling the hemisphere, we are not in a discrete case, an area can be divided as long as it is not found homogeneous, leading to an enormous amount of rays traced, without a lot of interest.

Our tests have shown that dividing a cell further than 3 times (64 sub-cells) is useless. The quality of the final picture does not seem to be improved when viewed on a monitor, neither is the interpolation process. With such a value an hemisphere is divided, in the worst case, into $3 \times 4 \times 64 = 768$ cells. With, at least 10 rays per cell, it is a minimum of 7680 rays that are traced, more than sufficient to describe the indirect incoming illumination (about 1 every $8e^{-4}$ sr).

7. Results

	Cornell Box (figure 6)	Corridor (figure 7)
Brute Force Monte Carlo	17h 37mn no *ILVs* 1024 rays / pixel	13h 39mn no *ILVs* 1024 rays / pixel
Original Method (sub-figures *a*)	14mn 26s 900 *ILVs* 1024 rays / *ILV*	9mn 27s 1580 *ILVs* 1024 rays / *ILV*
Our Method (sub-figures *b*)	5mn 51s 2045 *ILVs* 207 rays / *ILV*	9mn 52s 2636 *ILVs* 174 rays / *ILV*

Table 1: *Comparison for time and number of* ILVs *between the three methods*

All the results explained below have been obtained with a *270MHz R12000 MIPS processor* and 512Mb of memory. Each picture is 600×600 pixels size.

The scenes used to show results are very simple on purpose. Very complex scenes are not useful to visually compare resulting pictures, and visual comparison is the only one we have at the moment.

The first two pictures, shown by figure 6, present a Cornell Box-like scene. It is a scene with few, but big objects and many transitions between light and shadow. In that case, our method seems to be very clever at finding discontinuities in the incoming indirect illumination. *ILVs* are computed with only a few rays per hemisphere (an average of 207 over the entire picture). The original method (cf. section 3.1) manages to render a visually equivalent picture only if at least 1024 rays are traced to compute each *ILV* .

Considering this scene, our algorithm leads to a much higher number of *ILVs* computed. This is to be put in relation with the interpolation process. Our method gives a better description of the incoming indirect illumination than the original method, including its average direction. Because of that, interpolation criteria between *ILVs* are less often satisfied, and new *ILVs* are evaluated from the beginning. This point makes us believe that the interpolation process is not fully adapted anymore to our algorithm, and needs to be updated.

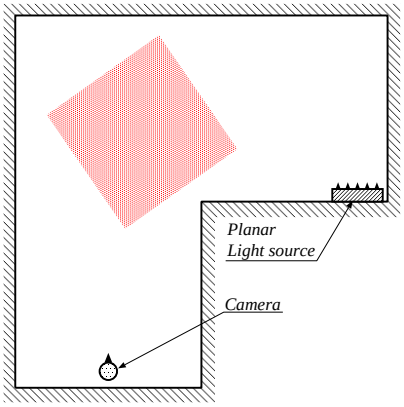


Figure 5: *Diagram (top view) of the scene used to render pictures on figure 7*

The two pictures of figure 7 (whose layout is drawn on figure 5) emphasize the fact that, even if the computation time is almost the same, the quality of the picture is enhanced. The indirect light distribution is described more sharply and this can be seen more precisely on two regions of the pictures. This first one is the shadow of the red cube on the left wall of the corridor which is rendered more precisely. This shadow is due to direct illumination. The second one is the shadow of the cube on the floor on the left hand side of the cube. This one is an indirect shadow, due to the reflection of the light on the diffuse background wall. This one is the most difficult to catch (a shadow of indirect light) and our method manages to do it. We can see on picture *(a)* that this shadow is missing and the cube appears floating.

7.1. Remarks about computation time

We may notice that the difference of computation time between the two methods is not a linear function of the number of rays per *ILV* traced. This may be easily explained if the time spent to find the homogeneous regions of each hemisphere is taken into account. The *SPRT* is a costly process and that is why the criteria used to rule on hemogeneity is very simple: radiance distribution over an interval. We have experienced a variance test of the incoming indirect radiance, but it has been found too costly considering the results which were almost the same as the one obtained with our method.

Moreover, computation times given here for such simple scenes may appear *outdated*. Some recent methods seem to be much efficient than ours, but most of them are hardware based. We want to put forward that we do not use any kind of optimisation (the *CSG* description of the scenes prevents us from using hardware acceleration), and we want our algorithm as generic as possible. This way, our method can easily be used with any kind of scene, *BRDF* or illumination description as well as ported on any kind of platform.

8. Conclusions and Future Work

In this paper we have presented a new method to compute *Indirect Light Vectors*. This method allows to compute scenes taking indirect illumination into account, faster than the original method, by reducing the number of rays traced. This is made possible by using the *Sequential Probability Ratio Test* which helps us find regions where indirect illumination is almost constant and then requires fewer rays to be approximated. We have also shown that, even if computation times are not always reduced, illumination description is improved, catching effects the original method was unable to render.

In the future, we plan to use the *SPRT* to spread the seed over the picture in a more appropriate way. We believe a good seed will automatically improve the interpolation process, dividing the scenes into regions where indirect illumination is quite constant.

References

1. Abraham Wald. *Sequential Analysis*. Wiley Publications in Statistics, (1947). 1, 2, 4

2. Marcel Berger. *Géométrie 5/La sphère pour elle-même, géométrie hyperbolique, l'espace des sphères*. CEDIC/FERNAND NATHAN, (1977). 6

3. Whitted Turner, "An Improved Illumination Model for Shaded Display", *CACM*, **23**(6), pp. 343–349 (1980). 2

4. Cindy M. Goral, Kenneth K. Torrance, Donald P. Greenberg and Bennett Battaile, "Modelling the Interaction of Light Between Diffuse Surfaces", *Computer Graphics (SIGGRAPH '84 Proceedings)*, **18**(3), pp. 213–222 (July 1984). 1

5. J. M. Augenbaum and C. S. Peskin, "On the Construction of the Voronoï Mesh on a Sphere", *Journal of Computational Physics*, **59**, pp. 177–192 (1985). 6

6. J. T. Kajiya, "The Rendering Equation", *Computer Graphics (SIGGRAPH '86 Proceedings)*, **20**, pp. 143–150 (August 1986). 2

7. Gregory J. Ward, Francis M. Rubinstein and Robert D. Clear, "A Ray Tracing Solution for Diffuse Interreflection", *Computer Graphics (SIGGRAPH '88 Proceedings)*, **22**(4), pp. 85–92 (August 1988). 1, 2

8. P. S. Heckbert, "Adaptative Radiosity Textures for Bidirectional Ray Tracing", *Computer Graphics*, **24**(4), pp. 145–154 (1990). 2

9. J-L. Maillot, L. Carraro and B. Péroche, "Progressive Ray Tracing", *Third Eurographics Workshop on Rendering*, pp. 9–20 (May 1992). 2

10. Gregory J. Ward and Paul Heckbert, "Irradiance Gradients", *Third Eurographics Workshop on Rendering*, pp. 85–98 (May 1992). 2

11. Eric P. Lafortune and Yves D. Willems, "Bi-directional Path Tracing", *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques*, pp. 145–153 (December 1993). 2

12. J. Arvo, "The Irradiance Jacobian for Partially Occluded Polyhedral Sources", *Computer Graphics*, **28**, pp. 343–350 (1994). 2

13. E. P. Lafortune and Y. D. Willems, "A Theoretical Framework for Physically Based Rendering", *Computer Graphics Forum*, **13**(2), pp. 97–107 (June 1994). 2

14. François X. Sillion and Claude Puech, *Radiosity and Global Illumination*, Morgan Kaufmann Publishers, San Francisco, (1994). 1

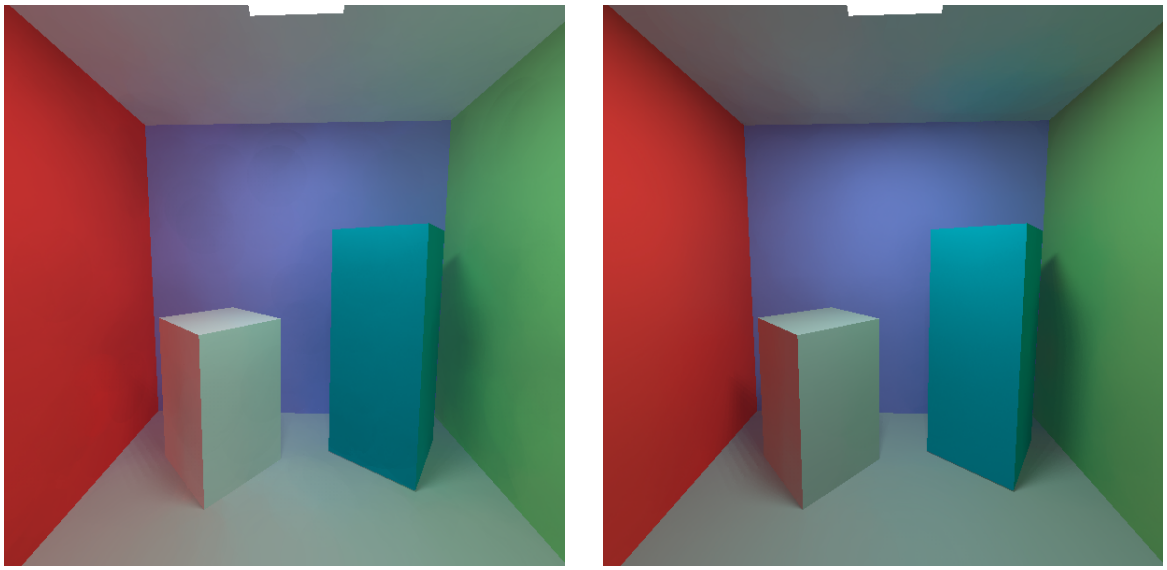
15. Eric Veach and Leonidas Guibas, "Bidirectional Estimators for Light Transport", *Fifth Eurographics Workshop on Rendering*, pp. 147–162 (June 1994). 2

16. Eric Veach and Leonidas J. Guibas, "Optimally Combining Sampling Techniques for Monte Carlo Rendering", *Computer Graphics (ACM SIGGRAPH '95 Proceedings)*, pp. 419–428 (1995). 2

17. Henrik Wann Jensen, "Global Illumination using Photon Maps", *Eurographics Rendering Workshop*, pp. 21–30 (June 1996). 2

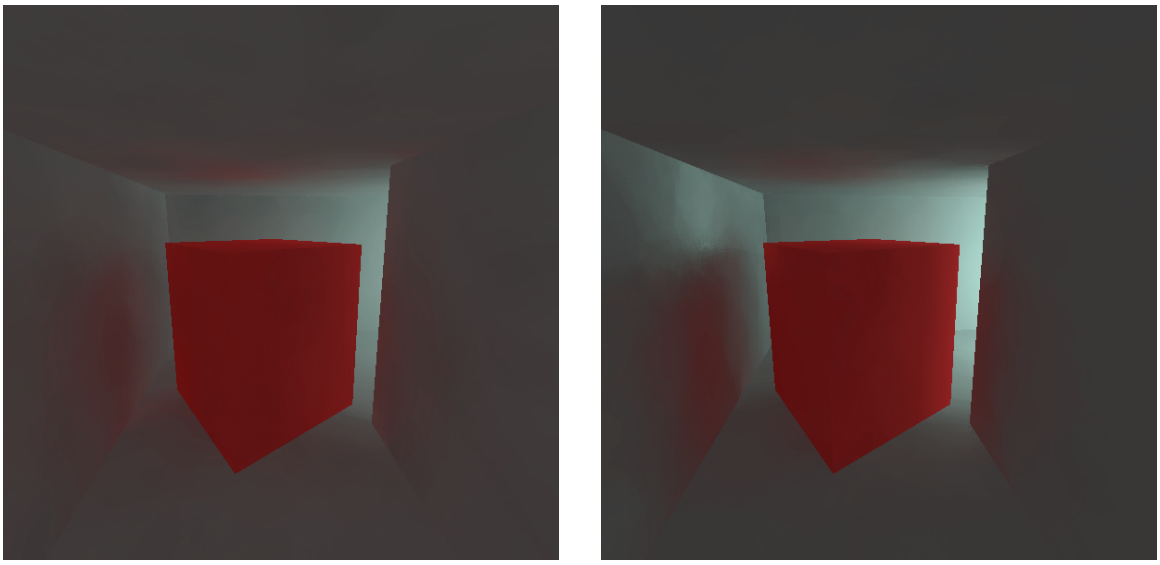
18. Eric Veach and Leonidas J. Guibas, "Metropolis Light Transport", *Computer Graphics (SIGGRAPH '97 Conference Proceedings)*, pp. 65–76 (August 1997). 2

19. Jacques Zaninetti, Xavier Serpaggi and Bernard Péroche, "A Vector Approach for Global Illumination in Ray Tracing", *Computer Graphics Forum*, **17**(3), pp. 149–158 (Septembre 1998). 1, 2, 5



(a) *(b)*

Figure 6: *Comparison of the original* ILVs *computation method (a) and our method (b) on a Cornell Box-like scene.*



(a) *(b)*

Figure 7: *A corridor where indirect illumination prevails. Our method (b) is not faster, but the illumination is rendered in a better way.*