A Vector Model for Global Illumination in Ray Tracing

Jacques ZANINETTI   Bernard PEROCHE
Ecole des Mines de Saint-Etienne
Centre SIMADE, Laboratoire LISSE
158, Cours Fauriel - 42023 Saint-Etienne Cedex - France
email : zaninetti@emse.fr peroche@emse.fr

ABSTRACT

This paper presents a method taking global illumination in ray tracing into account. A vector approach is introduced which allows to use any type of material, in particular with directional properties. This vector is decomposed into a direct component associated to light sources and an indirect one which corresponds to light having been reflected at least once. These components are estimated at a small number of points within the scene. A weighted interpolation between known values allows to reconstruct these components for the other points, with the help of a gradient computation for the indirect component. Computed images are thus more accurate, at almost no additional cost, and no discretizing of the geometry of the scene is needed.

Key Words: Rendering Algorithms, Ray Tracing, Shading, Global Illumination.

1 Introduction

Realistic images of scenes can be generated by high quality physically-based rendering algorithms that accurately calculate the distribution of light in an environment.

A well known technique for this purpose is ray tracing [Whi80] that samples light by tracing rays from the eye point into the scene and looking for intersections with the objects in the scene. Thus standard ray tracing can generate images with shadows, specular surfaces and transparency. However, the treatment of diffuse interreflections between surfaces is approximated by a constant ambient term, which does not fit any physical law. Improvements to standard ray tracing were made by adding stochastic sampling to the deterministic source and specular calculation by Cook [CPC84]. This method was later extended to path tracing by Kajiya [Kaj86]. Ward et al. [WRC88] improved on this by devising a coherence method that is based on the fact that the amount of diffusely reflected light received by a surface changes gradually over the surface. Storing sampled values at the first sample points and reusing them for later samples in the same region of the scene considerably reduces the sampling effort. This method led to the well-known *Radiance* rendering system [War94].

Yet, Ward's caching method does not allow to take directional properties of materials into account for the computation of the indirect component. This paper offers a vector approach of the modeled phenomena which will allow us not to be restricted to the diffuse component of illumination. Thus, we shall define the concept of Light Vector, inspired by Arvo's *Irradiance Vector* [Arv94], which allows to find the correct value of reflected radiance again. To permit an accurate simulation of global illumination, two types of Light Vectors are introduced: Direct Light Vectors, which describe light coming directly from light sources and Indirect Light Vectors, which represent light having been reflected at least once before hitting the current point.

The remainder of the paper is organized as follows. Section 2 presents basic definitions in the domain of illumination simulation. Section 3 introduces our fundamental concept, the Light Vector. The direct component of this vector is studied in section 4 and its indirect component in section 5. Some results are discussed in section 6, some further developments are suggested in section 7, and a conclusion is given in section 8.

2 Illumination concepts

In this section, we shall briefly present the main concepts used in our work and we shall define the notations.

Radiance $L(x, \vec{\omega})$ [NRH+77] (whose unit is $Wm^{-2}sr^{-1}$) is, for a given point x, the power per unit projected area perpendicular to vector $\vec{\omega}$ per unit solid angle in direction $\vec{\omega}$.

Another important concept is that of *irradiance* $E(x)$ (whose unit is Wm^{-2}), which gathers the whole energy on a given point x, taking the incident angle, but not the BRDF into account.

$$E(x) = \int_{\Omega} L_i(x, \vec{\omega_i}) \cos\theta_i \, d\omega_i \qquad (1)$$

The concept of *bidirectional reflection distribution function* (whose unit is sr^{-1}) is well known to model reflections on materials :

$$f_r(x, \vec{\omega_i} \rightarrow \vec{\omega_r}) = \frac{dL(x, \vec{\omega_r})}{L(x, \vec{\omega_i}) \cos\theta_i \, d\omega_i}$$

BRDF is also dependent upon wavelength, even if, for writing convenience, this parameter does not explicitly appear in the notation.

In this work, we shall assume that the BRDF can be split into a purely specular component $f_{r,s}$ and another component $f_{r,d}$ which includes a diffuse component and a specular lobe component [HTSG91] :

$$f_r(x, \vec{\omega_i} \rightarrow \vec{\omega_r}) = f_{r,s}(x, \vec{\omega_i} \rightarrow \vec{\omega_r}) + f_{r,d}(x, \vec{\omega_i} \rightarrow \vec{\omega_r}) \qquad (2)$$

We implemented Schlick's analytic model [Sch94], because it is both physically plausible and inexpensive. But any other model fulfilling the physical laws of reflection may be used, without modification.

The *rendering equation*, now well-known in computer graphics, stems from the formulation by Kajiya [Kaj86]. This equation provides the best foundation for building a physically-based rendering system.

$$\begin{aligned} L_r(x, \vec{\omega_r}) &= L_e(x, \vec{\omega_r}) \\ &+ \int_{\Omega} f_r(x, \vec{\omega_i} \rightarrow \vec{\omega_r}) \, L_i(x, \vec{\omega_i}) \cos\theta_i \, d\omega_i \\ &= L_e(x, \vec{\omega_r}) + L_{i,r}(x, \vec{\omega_r}) \end{aligned} \qquad (3)$$

The obvious simplicity of this formulation obscures the great complexity of simulated phenomena. Recursiveness, the need to evaluate the integral at every surface point and an only partial knowledge of the main terms appearing in the formula, practically forbid any analytic solution, even for very simple scenes.

3 A vector model

3.1 Radiance components

The reflected radiance may be split into three components, which will be investigated separately:

$$L_{i,r}(x, \vec{\omega_r}) = L_{dir}(x, \vec{\omega_r}) + L_{spec}(x, \vec{\omega_r}) + L_{ind}(x, \vec{\omega_r})$$

where $L_{dir}(x, \vec{\omega_r})$ is the direct component due to light sources, $L_{spec}(x, \vec{\omega_r})$ is the pure specular component, and $L_{ind}(x, \vec{\omega_r})$ is the indirect component.

Component $L_{spec}(x, \vec{\omega_r})$ is computed by a classical ray tracer, in which reflected and refracted rays are cast when necessary.

3.2 Light Vector

We first introduce the concept of *Light Vector*, which allows to find the correct value of reflected radiance again, for each point in which it is computed.

This concept is derived from the *Irradiance Vector* investigated by Arvo [Arv94] to model partially occluded polyhedral sources. However, the material and the sighting direction have to be taken into account for the computation of this Light Vector. Our goal is to restrict reconstruction errors in the reflected direction, i.e. for the light discerned by the eye. Such a description, which is independent of the radiance components, gives a really homogeneous representation for global illumination.

The Light Vector is composed of two fields: an average incident direction $\vec{D}$ and an energy magnitude P (expressed in Wm^{-2}). P is computed with the help of equation:

$$L_{i,r}(x, \vec{\omega_r}) = P \, f_{r,d}(x, \vec{D} \rightarrow \vec{\omega_r}) \qquad (4)$$

The Light Vector has $n+3$ components, where n depends on the representation model for colors ($n = 3$ in the RGB space, much more with a spectral model).

This vector may be seen as a virtual point light source simulating the effect of the whole incident energy at point x.

When it is unknown, an approximate value of this vector may be deduced from already computed values by a voector interpolation (and not a scalar one like previous methods). By using additional parameters, the interpolation may be refined and validity bounds may be set. By taking the BRDF f_r and a privileged direction $\vec{D}$ into account, errors made are noticeably reduced.

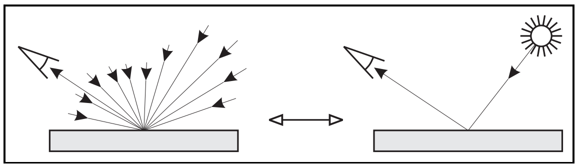


Figure 1: The Light Vector simulates the set of incident contributions by a unique virtual point light source.

Incident radiance may be decomposed into two independent components: a direct one (coming from light sources) and an indirect one (which has been reflected at least once). Thus, we shall introduce two types of light vectors: *Direct Light Vectors*, denoted $\overrightarrow{DLV}$s, and *Indirect Light Vectors*, denoted $\overrightarrow{ILV}$s.

3.3 Computing a picture

3.3.1 Creation of a seed

Standard row by row computation of a picture could introduce significant errors: interpolation between known values would systematically ignore half the neighbourhood of the current point. To avoid this problem, a seed of randomly distributed points in the scene is chosen. For each such point, direct and indirect components of Light Vectors are computed and stored. The storage of these vectors is made with kd-trees [Ben75], which allows to perform efficient search of space neighbourhoods. Several distinct kd-trees are used: one of them contains all the $\overrightarrow{ILV}$s, whereas each of the others contains the $\overrightarrow{DLV}$s associated to a given light source.

3.3.2 Traversal of the set of pixels

The second step is a usual traversal of the picture, with possible generation of reflected and refracted rays. The main difference in relation to standard ray tracing is the search, in a neighbourhood of the current point, of known values for direct and indirect radiances in their respective kd-trees. When it is possible, the result of the interpolation is used in place of a complete calculation. When a complete calculation is made, the result is stored in vector form and may be used for later interpolations.

We shall now explain in the next two sections how $\overrightarrow{DLV}$s and $\overrightarrow{ILV}$s are computed and how the validity of the interpolation used is estimated.

4 Direct component

In this section, we present a method which allows to speed up the computation of direct illumination, by reusing, when it is possible, already computed values in a neighbourhood of the current point. This method is inspired by Jensen's *photon map* [Jen96] and adapted to our vector model. It is especially beneficial when complex light sources, requiring a costly sampling process, are used.

4.1 The concept of Direct Light Vector

The *Direct Light Vector*, denoted $\overrightarrow{DLV}$, represents, at a given point in the scene, the energy supplied by a given light source. It is composed of three fields:

- An average incident direction $\vec{D}$, which is the direction of Arvo's Irradiance Vector;
- A solid angle subtented by the light source;
- An energy magnitude P satisfying equation (4).

Currently, direction $\vec{D}$ is computed by weighting the n sampling directions $\vec{\omega_i}$ of the light source by incident radiance L_i and by the projected solid angle Ω_i associated to $\vec{\omega_i}$: $\vec{D} = \frac{\sum_{i=1}^{n} L_i \Omega_i \vec{\omega_i}}{\sum_{i=1}^{n} L_i \Omega_i}$.

This computation is easily made when shadow rays are cast towards the light source. In particular, when dealing with a point light source, the calculation is reduced to its simplest form. The $\overrightarrow{DLV}$ is stored even when the light source is completely occluded, in order to reduce the number of rays to be generated later.

4.2 Direct interpolation

4.2.1 Search for neighbours

A set of usable and already calculated $\overrightarrow{DLV}$s is chosen before doing the interpolation. This set is made up of the $nmax$ nearest neighbours of the current point, located inside a neighbourhood of radius $rmax$. This set is rejected if the number of $\overrightarrow{DLV}$s found is less than some threshold $nmin$. In this case, a full calculation is started.

The choice of these three parameters is mainly dependent on the size of the scene. In practice, for indoor scenes, $nmin = 5$, $nmax = 50$ and $rmax = 0.08m$ give good results with an initial seed of 5000 $\overrightarrow{DLV}$s.

4.2.2 Validity test

Dispersion between suitable candidates for interpolation must be beyond a given threshold. As the direct component may have great changes, dispersion is measured by calculating the variance of the set of brightness values ($l = 0.299r + 0.587g + 0.114b$ in the rgb color space) of the samples. In practice, a low threshold of 20% is used, which is weak enough to detect quick changes for direct irradiance.

In fact, this method allows to more or less compare the Y component of the vectors in a XYZ color space. The good solution for our problem would be to work with a spectral model in a color space provided with a perceptual metric. But this problem still is a current research theme.

4.2.3 Interpolation

The value of a $\overrightarrow{DLV}$ is obtained by a weighted interpolation of the directions $\vec{D}_i$ and of the energies P_i of the selected sampled vectors. The weighting function f is a tight polynomial approximation of a Gaussian:

$$f(d_i) = [\,2(rmax - d_i)^3 - 3(rmax - d_i)^2 + 1\,]^2$$

where d_i is the distance between the current point and sample i.

$\vec{D} = \frac{\sum_{i=1}^{n} f(d_i)\vec{D}_i}{\sum_{i=1}^{n} f(d_i)}$ and $P = \frac{\sum_{i=1}^{n} f(d_i)P_i}{\sum_{i=1}^{n} f(d_i)}$.

For each light source in the scene, the value of the direct component $L_{dir}(x,\vec{\omega_r})$ of the reflected radiance $L_{i,r}(x,\vec{\omega_r})$ is given by equation (4). The direct component of the reflected radiance at point x is obtained by adding all the components associated to each light source.

5 Indirect component

In this section, we shall explain how the indirect component of the reflected variance - denoted $L_{ind}(x,\vec{\omega_r})$ in section 3.1 - is computed.

5.1 General principle

The indirect component of reflected light is computed with a Monte Carlo method, by sampling an hemisphere centered on the current point. Only light having been reflected at least once is taken into account here. Power coming directly from light sources is evaluated with L_{dir}. Because of the recursive formulation of equation (3) and of the great number of rays to be generated in order to reduce the variance, the computing time is very long.

The following assumption is usually admitted in computer graphics [War94]: *Indirect illumination changes gradually over surfaces.* By agreeing with this hypothesis, we may consider that around every point in the scene, there exists a neighbourhood in which the changes of indirect illumination are less than a given threshold. Then, it is possible to reconstruct the value of a given point from the values of close points, provided the valid neighbourhood of the latter contains the former.

An irradiance gradient is evaluated at the same time as the indirect component, at almost no additional cost. This gradient is not accurate enough to be used outside of a small neighbourhood of the given point, because of the too many unknown data. But it is a very good gauge of the degree of perturbation of the current zone. Thus it is useful to refine the computation of the validity of the zone.

Before doing a complete computation, we must check if there exist already computed valid values. In this case, the current value is estimated by a simple weighted interpolation of known values. The weight assigned to each known value is inversely proportional to the errors that may be introduced. This error estimation is mainly dependent on the distance between the samples, the change in surface orientation, and the gradient. In the opposite case, a Monte Carlo computation is done and the obtained value is stored, in order to be re-used, if necessary.

5.2 Monte Carlo method

5.2.1 Computation of the indirect component

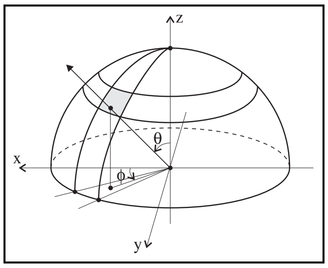


Figure 2: Sampled hemisphere

To make our computation, the hemisphere is descretized into cells. Each cell of the hemisphere has a constant solid angle. In this way, we may access explicitly the directional information for the gradient computation. Angles θ_j and ϕ_k are chosen in such a way that they define a random direction inside each cell (j,k) of the hemisphere. The solid angle subtended by each cell is $\omega_{j,k} = \frac{2\pi}{MN}$, and $\theta_j = cos^{-1}\left(\frac{M-j-X}{M}\right)$, $\phi_k = \frac{2\pi}{N}(k+Y)$, where X and Y are uniform random variables in $[0,1[$ and where M is the number of samples according to the polar angle and N the number of samples according to the azimuthal angle.

So, after discretization of the hemisphere, rendering equation (3) becomes:

$$L_{ind}(x,\vec{\omega_r}) \approx \frac{2\pi}{MN}\sum_{j=0}^{M-1}\sum_{k=0}^{N-1} f_{r,d}(x,\vec{\omega_{j,k}}\rightarrow\vec{\omega_r})L_i(x,\theta_j,\phi_k)\cos\theta_j \quad (5)$$

This sampling method is recursively used, but with a small predefined level of recursiveness.

5.2.2 Average incident direction

Vector $\vec{D}$ shows the average incident direction at point x: $\vec{D}=\int_\Omega L_i(x,\vec{\omega_i})\,\vec{\omega_i}\cos\theta_i\,d\omega_i$.

The computation of $\vec{D}$ may be done in the same loop iteration than (5), for a small cost. To facilitate the interpolation, this vector is normalized before being stored, only its direction being useful for us.

5.2.3 Irradiance gradient

A gradient computation is made when rendering equation (5) is evaluated. As these calculations are done during the same iteration of the loop, the extra computational time is negligible. The gradient is different from the one introduced by Ward and Heckbert [WH92], because we do not use the same sampling of the hemisphere.

The gradient shows indirect illumination variations in a neighbourhood of the given point. Its numerical value gives a good indication of the degree of pertubation of the zone taken into account. It is used to associate, to each $\overrightarrow{ILV}$, a validity zone inside which interpolation is possible.

The gradient is obtained by deriving irradiance expression (1) according to displacements in a (x,y,z) coordinate system tangent to the surface:

$$\begin{aligned}\frac{\partial E}{\partial a} &= \frac{2\pi}{MN}\sum_{j=0}^{M-1}\sum_{k=0}^{N-1}\frac{\partial(L_i(\theta_j,\phi_k)\cos\theta_j)}{\partial a}\\ &= \frac{2\pi}{MN}\sum_{j=0}^{M-1}\sum_{k=0}^{N-1}\left(\frac{\partial L_i(\theta_j,\phi_k)}{\partial a}\cos\theta_j\right.\\ &\qquad\left.+\,L_i(\theta_j,\phi_k)\frac{\partial\cos\theta_j}{\partial a}\right)\end{aligned}$$

with $a \in (x,y,z)$.

However, $\frac{\partial L_i(\theta_j,\phi_k)}{\partial a}$ is unavailable. Thus, we must assume that displacements in the neighbourhood of point x are small enough to admit $\frac{\partial L_i(\theta_j,\phi_k)}{\partial a} = 0$. With this assumption:

$$\frac{\partial E}{\partial a} \approx \frac{2\pi}{MN}\sum_{j=0}^{M-1}\sum_{k=0}^{N-1} -L_i(\theta_j,\phi_k)\sin\theta_j\frac{\partial\theta_j}{\partial a} \quad (6)$$

$\frac{\partial\theta_j}{\partial a}$ may be computed by examining the changes of incident angle for small displacements of the center of the hemisphere: $\frac{\partial\theta_j}{\partial x} = \frac{-\cos\theta_j\cos\phi_k}{R}$, $\frac{\partial\theta_j}{\partial y} = \frac{-\cos\theta_j\sin\phi_k}{R}$ and $\frac{\partial\theta_j}{\partial z} = \frac{\sin\theta_j}{R}$, where R is the distance between the current point and the first object hit in direction (θ_j,ϕ_k).

5.3 Indirect Light Vector

As the interpolation process is a little more sophisticated for Indirect Light Vectors than for Direct Light Vectors, the Indirect Light Vector has more additional fields: the gradient, the tangent coordinate system to the surface, and the average distance to the other objects of the scene.

These data are used to control interpolation errors, and thus allow to reduce the total number of $\overrightarrow{ILV}$s computed with the expensive Monte Carlo method.

5.4 Validity zone

Let B be a point in which $\overrightarrow{ILV}(B)$ must be computed. To be accepted as usable in point B, $\overrightarrow{ILV}(A)$ computed in point A must satisfy several conditions:

- A and B must belong to objects having the same BRDF. This condition is due to the fact that the $\overrightarrow{ILV}$ is used through the BRDF of the material;
- Distance $\|\overrightarrow{AB}\|$ must be less than or equal to the mean arithmetic distance *dmean* between A and the rest of the scene. We impose this condition in order to prevent errors between close objects;
- The relative change of irradiance due to the gradient must be less than or equal to some user-specified accuracy tolerance S_E: $\frac{\overrightarrow{AB}.\overrightarrow{\Delta E}}{E_A} \leq S_E$;
- The surface curvature must be less than or equal to some predefined accuracy tolerance S_{curv}: $\cos(\vec{N_A},\vec{N_B}) \geq S_{curv}$. This condition is warranted by the fact that $\vec{D}$ shows the average incident direction at point x coming from a hemisphere, and not from a sphere.

To fulfill the conditions above, $\overrightarrow{ILV}$s are stored in kd-trees, with one kd-tree associated to all the objects in the scene having the same BRDF.

5.5 Weighted interpolation

5.5.1 Number of vectors

The number n of $\overrightarrow{ILV}$s used for interpolation must range between two predefined bounds, $nmin$ and $nmax$. The choice of these bounds influences the filtering level induced by the interpolation. In practice, empirical values $nmin = 3$ and $nmax = 16$ give good results.

5.5.2 Weighting function

The goal of the weighting function is to give less importance to the samples liable to introduce accuracy errors, following the criteria defined in section 5.4. A weight w is associated to each $\overrightarrow{ILV}$; its value is one when point B coincides with point A, less than one otherwise:

$$w = \frac{1}{1 + \frac{\|\overrightarrow{AB}\|}{dmean}} \cdot \frac{1}{1 + \frac{\overrightarrow{AB}.\overrightarrow{\Delta E}}{E_A}} \cdot \frac{\cos(\vec{N}_A, \vec{N}_B) - S_{curv}}{1 - S_{curv}} \tag{7}$$

5.5.3 Interpolation

A standard vector interpolation scheme is used, with the weights computed in equation (7):

$$\overrightarrow{ILV}(B) = \frac{\sum_{i=1}^{n} w_i \ \overrightarrow{ILV}(A_i)}{\sum_{i=1}^{n} w_i}$$

As the maximum number, $nmax$, of samples is small, this computation is fast.

6 Some results

This method was implemented in the ray tracing software YART, developed at Ecole des Mines de Saint-Etienne by Marc Roelens [Roe93].

The following pictures are intended to illustrate the use of the method described in this paper.

Figure 3 shows a Cornell box. The walls of the room are diffuse, one cube is diffuse and the other is purely specular. With an area light source, the computing time is about 8 minutes on a Silicon Graphics Indigo and for a 512×512 pixels picture size.

The series of pictures in figure 4 proposes a comparison between three methods for a given scene. This scene is composed of a closed diffuse room, a table and six spheres with complete BRDFs. Although it seems very simple, this test scene cannot be rendered correctly by a standard ray tracer. Actually, interesting objects are located on the ground, and only a small part of the ceiling is directly lit up. So, the greatest part of the picture would be made visible by the ambient term, without any physical justification.

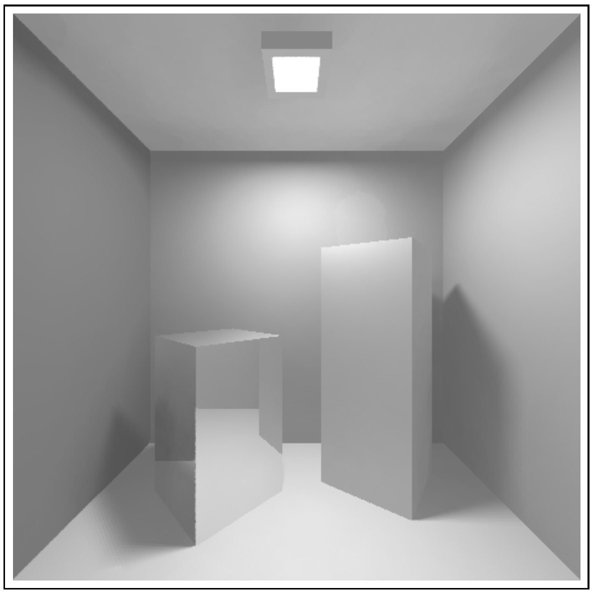


Figure 3: Cornell box.

The left most picture is computed with a naive Monte Carlo method, with 128 samples per hemisphere. For a 400×400 pixels picture size, the computing time is about 3 hours. The right most picture is computed with a standard ray tracer, with an ambient term. The computing time is about 2 minutes. The picture in the center is computed with our method. With 4200 $\overrightarrow{ILV}$s, the computing time is around 10 minutes. Notice that the central picture is much less noisy than the left one and shows a more realistic penumbra under the table than the right one.

Figure 5 shows a more complex scene, with 3D (wood and marble) and 2D (the painting) textures. There is a mirror hung up on a wall and many shiny surfaces. The volumic light source has been simulated by using 32 point light sources randomly located inside a sphere enclosed in a specular reflector. This scene includes about 500 objects. The picture size is 800×600 pixels and the computing time is about 3 hours, with an adaptive antialiasing process and the computation of about 15000 $\overrightarrow{DLV}$s and about 11000 $\overrightarrow{ILV}$s.

7 Further developments

The work presented in this paper is only a first sketch on the subject, and we are considering several developments:

- improvement of the implemented Monte Carlo method, by using an importance sampling, in order to reduce the variance of the calculation of Light Vectors and thus to decrease the requested accuracy;
- use of a spectral model for colors when the rgb model is too rough for the requested accuracy;
- estimation of the error made, by taking psycho-visual data into account ;
- physical computation of caustics [Jen96];
- incremental and adptive improvement of the accuracy, without losing already computed data;
- taking area light sources into account in a more optimized way than point sampling.

8 Conclusion

We have presented a method taking global illumination into account in a ray tracing environment, whose advantages are preserved. No preliminary discretization of the surfaces of the scene is needed and all the types of BRDFs are handled. The overhead is moderate, for an application the aim of which is to produce physically plausible pictures, with an adaptive error control.

References

[Arv94] J. Arvo, *The irradiance jacobien for partially occluded polyhedral sources*, Computer Graphics **28** (1994), 343–350.

[Ben75] J.L. Bentley, *Multidimensional binary search trees used for associated searching*, Communication of the ACM **18** (1975), 509–517.

[CPC84] R. L. Cook, T. Porter, and L. Carpenter, *Distributed ray tracing*, Computer Graphics **18** (1984), no. 3, 137–144.

[HTSG91] X. D. He, K. E. Torrance, F. X. Sillion, and D. P. Greenberg, *A comprehensive physical model for light reflection*, Computer Graphics **25** (1991), no. 4, 175–186.

[Jen96] H. W. Jensen, *Global illumination using photon maps*, Seventh Eurographics Workshop on Rendering, Porto (1996), 22–31.

[Kaj86] J. T. Kajiya, *The rendering equation*, Computer Graphics **20** (1986), no. 4, 143–150.

[NRH+77] F. Nicodemus, J. Richmond, J. Hsia, I. Ginsberg, and T. Limperis, *Geometric considerations and nomenclature for reflectance*, Monograph 160, National Bureau of Standards, 1977.

[Roe93] M. Roelens, *Un environnement pour le tracé de rayons utilisant une modélisation par arbre de construction*, Ph.D. thesis, Ecole des Mines de Saint-Etienne, February 1993.

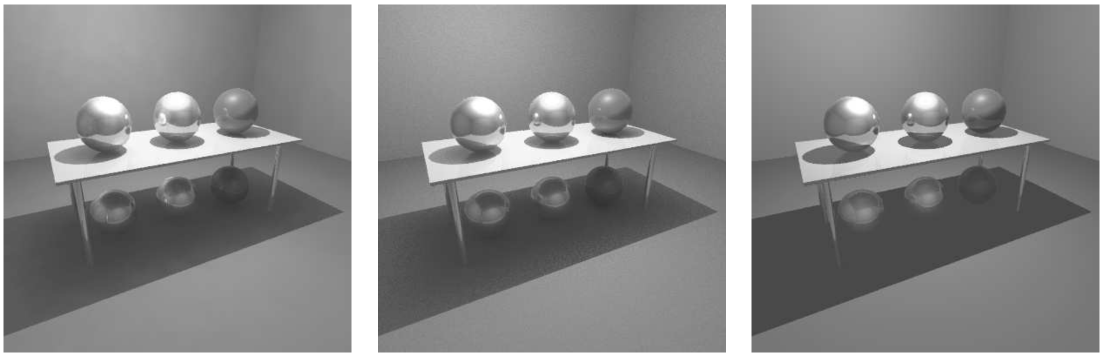
[Sch94] C. Schlick, *An inexpensive brdf model for physically-based rendering*, Computer Graphics Forum **13** (1994), no. 3, 149–162.

[War94] G. J. Ward, *The radiance lighting simulation and rendering system*, Computer Graphics (1994), 459–472.

[WH92] G. J. Ward and P. S. Heckbert, *Irradiance gradients*, Third Eurographics Workshop on Rendering, Bristol, England (1992), 85–98.

[Whi80] T. Whitted, *An improved illumination model for shaded display*, Communications of the ACM **23** (1980), 343–349.

[WRC88] G. J. Ward, F. M. Rubinstein, and R. D. Clear, *A ray tracing solution for diffuse interreflection*, Computer Graphics **22** (1988), no. 4, 85–92.



Monte Carlo : ≈ 3 h | 4200 $\overrightarrow{ILV}$: ≈ 10 mn | Ambient Term: ≈ 2 mn

Figure 4: Some comparisons between 3 methods.

Figure 5: An indoor simulation.