The Discrete Tube: A Spatial Acceleration Technique for Efficient Diffraction Computation

Lilian Aveneau, Eric Andres, Michel Mériaux

IRCOM SIC, UMR 6615 CNRS
Boulevard 3 - Téléport 2 - BP 179
86960 Futuroscope Cedex, France
E-mail: {aveneau,andres,meriaux}@sic.sp2mi.univ-poitiers.fr

Abstract. Keller's Geometrical Theory of Diffraction [8] allows to render scenes with dihedron diffraction account. The Diffraction algorithm presented in [2] is too slow, since its complexity is linear with respect to the number of dihedra. In order to accelerate it, we propose to reduce the complexity with a discrete based algorithm. Considering that diffraction mainly occurs inside the n-first FRESNEL's ellipsoids [11], we can limit the diffraction computation to dihedra inside such ellipsoids. For efficiency we propose to use an ellipsoid approximation, the discrete tube. We describe two different algorithms for computing such a discrete tube. Their results are discussed, and show an important acceleration compared to the previous method.

Keywords: Discrete Algorithm – Rendering techniques – Diffraction – GTD – Ray-Tracing

1 Introduction

Diffraction is the electromagnetic phenomenon which explains the wave scattering in object vicinity [3]. At visible wavelength this effect is generally of low importance and is usually neglected. Some physical simulations however imply to compute the wave propagation as accurately as possible, *e.g.* in avionic system, or in architectural design.

Since for high wavelengths diffraction is of great importance, an other demand comes from wireless phone providers : to ensure good communications at every location in their covering areas, they need to put many transceivers in cities. In order to reduce their number, they have to compute the transmission quality between a fixed transmitter and a mobile one (a wireless phone). Their goal is to minimise the fixed transceiver number while keeping a good city coverage.

In order to render image synthesis scenes with diffraction account, we have proposed [2] a solution based on KELLER's Geometrical Theory of Diffraction [8]. This solution allows to compute energy transport with diffraction account not only at the camera like in NAKAMAE's method [10], but also at each scene points. In image synthesis this solution allows for instance to compute shadows with more precision. Nervertheless our solution required an important computation

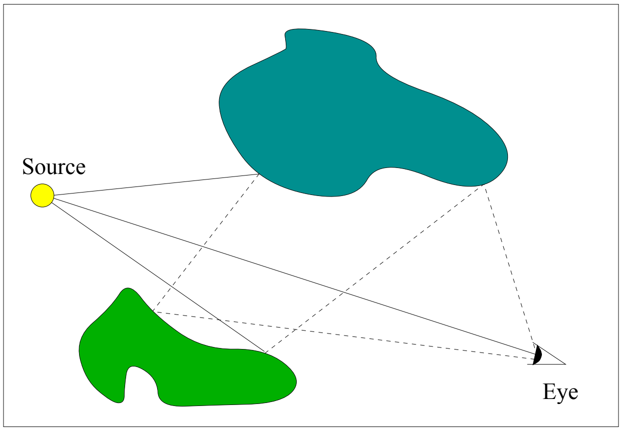

Fig. 1. *Ray classification : direct rays are continuous, reflected rays are dotted*

time : at each point, computation complexity is linear with respect to the number of dihedra. Rendering complex scenes implies high computation time. Since the diffraction mainly occurs in the n-first Fresnel's ellipsoids, we propose to use a regular grid and limit dihedron edge diffraction to a discrete tube which is the parallelepipedic cover of an ellipsoid. We choose to use such a tube since the computation of a discrete ellipsoid is much more complex and appears to be superfluous.

After describing two algorithms that compute such discrete tubes, we present some results and discuss the acceleration factors we have obtained.

2 The problem

2.1 Diffraction and Ray-Tracing

The *Ray-Tracing* algorithm [12] is based on classical optics which is a geometric theory where light is propagated in vacuum or in an homogeneous medium along straight lines called *rays*. The foundation of this theory is Fermat's principle which assumes that light path lengths are extrema. A ray classification can be made by separating direct rays, which are straight lines between the source points and the receiver points, and reflected rays which are straight lines between the reflection points and the receiver points. Thus light paths can be composed of a single direct ray, or a direct and one or more reflected rays (cf. fig. 1).

Classical optics does not allow to compute a realistic light propagation. Indeed the diffraction phenomenon, which explains light scattering in an object vicinity, cannot be rendered this way. Other theories, like Kirchhoff's [3], or Lorentz's microscopic one [7], allow to include such phenomena into simulations. They imply however the implementation of a new algorithm class, that produces results at a cost of a very large computation time. Our experiences have shown that they have a very bad convergence (so computation time is exceedingly large) and need high floating point precision. Therefore they can not be used directly in image synthesis because of their complexity.

We have chosen an other approach in order to produce realistic images. Indeed, the introduction of the diffraction phenomenon in the ray-tracing algorithm [2] is based on KELLER's Geometrical Theory of Diffraction (or GTD) [8] [9]. This theory extends the Geometrical Optics with a new class of rays, the diffracted rays, which are emitted by points localized on the object outline called the diffraction points. The foundation of this model is the extension of FERMAT's principle. It assumes that light paths, which can be composed of a direct ray, or a direct and a combination of reflected rays and diffracted rays, still have extremum lengths. With a direct application of the extended FERMAT's principle, such paths can be found by minimising the light paths for simple geometrical shapes, like dihedra [2].

With this theory, the diffraction treatment can be added into the ray-tracing algorithm [12]. More precisely, we can compute the direct diffracted illumination at a point by finding all the diffracted paths (with only diffraction points, no reflection point) between a source point and a receiver one. This computation is done in the WHITTED's *shade* [12][6] function, the goal of which is to evaluate the radiance received at a given point.

2.2 Complexity of the Diffraction Algorithm

The Diffraction Algorithm [2] can be written as :

```
1 For all ordered combination of n dihedron edges do
2     Compute the corresponding light path
3     If this path exist
4         Compute the carried radiance
5     EndIf
6 EndFor
```

Fig. 2. *The diffraction algorithm*

The main advantage of such a solution is to be faster than a basic Monte-Carlo approach based on KIRCHHOFF's Theory [3]. However, the use of KELLER's theory implies large computation even with low complexity scenes. The profiling of our diffraction algorithm shows that we spend an important part of the time in the search of diffracted paths. As for WHITTED's ray-tracing algorithm, the complexity is directly related to the number of scene objects. This appears obvious if we consider the first line of the diffraction algorithm (fig. 2). In order to produce realistic images with diffraction account, we have to accelerate this algorithm.

The first idea is to use a similar acceleration method than for the Ray-Tracing algorithm, *i.e.* an algorithm based on a spatial structure (*e.g.* octree, BSP, regular grid ...). Such an algorithm works for the classical Ray-Tracing because the light transport between two reflection points is a straight line. In the case of diffraction computations, the light transport between two points is more complex. Therefore this method can not be used directly in our diffraction algorithm.

Our idea is to search only for the diffraction paths that are inside an ellipsoid. Indeed with the HUYGENS-FRESNEL construction [3], it can be shown that the main part of the electromagnetic energy is propagated into the n-first FRESNEL's ellipsoids (95% in the first one). Moreover, it is easy to see that the GTD formulation keeps these properties. In our diffraction algorithm, we propose to limit the computations between a point source and a receiver point to paths that are strictly inside such an ellipsoid. This can be done since we have the following restriction : it is only necessary to consider path parts with no reflection point ; indeed, in this case such a path can transport a very important radiance whereas at least one of these points are not in the first ellipsoids.

2.3 Towards a discrete solution

We have seen that we can limit diffraction computation to dihedra that are inside an ellipsoid. This implies a small error on the received radiance, but we assume that, if we take for instance the ten first FRESNEL's ellipsoids, this error will be invisible for the human eye. The naive method for such a limitation is based on the analytical finding of the intersection of a dihedron and such an ellipsoid. There are two reasons that make it a bad solution. Firstly it implies to work with all the dihedra, so the complexity will be roughly the same than for the old Diffraction Algorithm. Secondly the cost of the analytical intersection between a line and an ellipsoid is close to the cost of the algorithm which computes and verifies the existence of a diffracted path. Our goal is to work with as few dihedra as possible in order to reduce the diffraction treatment complexity.

We thus propose to store all the dihedra into a regular grid. This may allow us to efficiently find those that are important in the computation of the light propagation between two points. By finding the intersection of the ellipsoid and the regular grid, we will reduce the number of dihedra involved in the diffraction transport. Thus, we will get an acceleration for our diffraction algorithm.

Our problem can be sumed up as the ability to compute the intersection of an ellipsoid and a regular grid as fast as possible. This problem is equivalent to finding the most efficient discretization of an ellipsoid. Because our goal is to produce an acceleration for the diffraction algorithm, instead of computing a discrete ellipsoid, we propose to compute a discrete parallelepipedic cover that we call the “discrete tube”. This is simpler to compute and thus faster, and it is easy to see that this does not mean more errors in the diffracted radiance computation since we may only get more dihedra, and thus more precision than with the actual ellipsoid.

3 The discrete tube

In this section we present two algorithms that allow us to compute such a tube. Both their complexities are quite similar, but the second one is slightly faster. However a modified version of the first algorithm could be used for the acceleration of beam tracing [5]. These algorithms are based on the efficient computation of a 3D six-connected line [4].

3.1 First algorithm : Discrete Line Scanning (DLS)

The main idea here is to construct discrete rectangles which have to be, firstly, centered on each voxel of a 6-connected discrete line and, secondly, orthogonal to this line. If we do that for all the voxels of a 6-connected discrete line segment, we obtain a discrete tube. For this construction we have to define the discrete plane that will become the support of such rectangles. As we will see later, we choose to use the naive plane, which is a particular case of the discrete analytical plane [1], for an efficient rectangle construction :

Definition 1. *(3D Discrete analytical plane [1]) A 3D discrete analytical plane $P = P_3(\alpha_0, \dots, \alpha_3, \omega)$ is the set of points $X = (x_1, x_2, x_3)^T \in \mathbb{Z}^3$ that verify this equation :*

$$0 \le \alpha_0 + \sum_{i=1}^{3} \alpha_i x_i < \omega,$$

where $\omega \in \mathbb{N}^*$ is called the arithmetical thickness, $(\alpha_1, \alpha_2, \alpha_3) \in \mathbb{Z}^3$ are called the coefficients, $\alpha_0 \in \mathbb{Z}$ is called the translation constant of the plane, with $gcd(\alpha_1, \alpha_2, \alpha_3) = 1$.

Definition 2. *(Naive plane [1]) A 3D plane is called a 3D naive plane if $\omega = \max_{i=1\dots3}(|\alpha_i|)$.*

Such a naive plane has an interesting property that will ensure we do not miss any discrete tube voxel. Indeed, a proposition from E. Andres states that the discrete 3D planes (and so the 3D naive planes) tile the space $\mathbb{Z}^3$:

Proposition 1. *(Tiling of $\mathbb{Z}^3$ [1])*

$$\mathbb{Z}^3 = \biguplus_{t=-\infty}^{+\infty} P_3(\alpha_0 + t\omega, \alpha_1, \alpha_2, \alpha_3, \omega).$$

Moreover, since 3D planes do not have 6-tunnels, by scanning the voxels of the 6-connected line, we ensure that we do not miss some planes, and so some discrete tube voxels.

All these theoretical bases allow us to write our algorithm. After an initialisation stage, an incremental part does the scanning of the discrete 3D line between the transmitter point E and the receiver point R.

Initialisation stage In the stage, we first have to initialize the computation of the line segment between the points E and R. This is equivalent to the well-known algorithm for drawing such a discrete line [4].

Next we continue with the preparation of the computation of the successive parallel naive planes P_i. We thus determine the first voxel $V_0 = (x_1^0, x_2^0, x_3^0)^T$ that contains the point E.
P_0 is the naive plane perpendicular to the direction $\overrightarrow{ER}$ which includes V_0 : its coefficients $(\alpha_1, \alpha_2, \alpha_3)$ are in a first step set to the discrete coordinates (a, b, c) of the vector $\overrightarrow{ER}$; in a second step they are reduced in such a way that $\gcd(\alpha_1, \alpha_2, \alpha_3) = 1$. The translation coefficient of P_0 is easy to compute since the voxel V_0 belong to P_0 ; indeed we have $\alpha_0^0 = -\sum_{i=1}^{3} \alpha_i x_i$. The arithmetical thickness ω of P_0 is computed according to the definition 2.

Incremental stage This stage is based on the 3D 6-connected discrete line construction.

For each voxel V_i of our discrete line $\mathcal{L}$, we compute the naive plane P_i which, firstly, contains V_i, and, secondly, is parallel to the first naive plane P_0 defined above[1]. This plane has the same coefficients than P_0. It differs only by its translation coefficient α_0^i (see proposition 1). We have :

$$\forall\, i > 0, \quad \alpha_0^i \;=\; \alpha_0^{i-1} + \delta^i \cdot \omega$$

with

$$\delta^i \;=\; \begin{cases} 1 \text{ if } V_i \;\notin\; P_{i-1} \\ 0 \text{ otherwise.} \end{cases}$$

If the intersection of the naive planes P_i and P_{i+1} is null, we have to compute the current rectangle. Assuming that the largest coefficient of our naive planes is α_1, the rectangle projection on the (y,z)-plane allows us to compute it easily[2]. The rectangle is centered on the voxel V_i. We compute the projected rectangle width as :

$$\text{width} \;=\; \left\lceil \frac{\mathrm{d_R} \cdot X_{\overrightarrow{ER}}}{||\overrightarrow{ER}||} \right\rceil$$

where d_R is the discrete tube diameter, $X_{\overrightarrow{ER}}$ is the x-coordinate of the $\mathbb{R}^3$-vector $\overrightarrow{ER}$, and the braces denote the *ceiling* function[3]. Since the rectangle belongs to a naive plane, and since the arithmetical thickness ω is equal to α_1, each pixel of our projected rectangle leads to its corresponding 3D voxel in a non-ambiguous manner. We have :

[1] It is important to notice that two consecutive naive plane P_i and P_{i+1} based on two consecutive voxels can be identical. So the recursive algorithm has to work both on the discrete line voxel V_i and on the successive naive planes term t defined in proposition 1.

[2] Construction is similar with α_2 or α_3 equal to $\max(|\alpha_1|, |\alpha_2|, |\alpha_3|)$.

[3] The ceiling function $\mathbb{R} \rightarrow \mathbb{Z}$ returns $\lceil x \rceil \;= \inf(n \in \mathbb{N} \mid n >= x)$

$$x = \left\lfloor -\frac{\alpha_2 \cdot y + \alpha_3 \cdot z + \omega}{\alpha_1} \right\rfloor$$

where the braces denote the *floor* function[4]. We get all the x-coordinates with a double loop on the y and z coordinates.

It is important to notice that this method for computing such rectangles works well since we choose to use naive planes. With a greater arithmetical thickness, *e.g.* for standard planes, the previous method cannot be used since for some couples (y,z) there are more than one solution for x.

3.2 Second algorithm : Discrete Rectangle Scanning (DRS)

The second discrete tube generation algorithm is a variation of the previous one. The main idea here is to extrude the first rectangle centered at V_0 along the 6-connected discrete line [4] that passes through the transmitter point E and the receiver point R.

Initialisation stage It begins as for the DLS algorithm by the definition of the naive plane P_0. We then compute the 6-connected discrete line segment (OP) which is the translation of the discrete line segment (ER) to the coordinates system origin (*i.e.* $\overrightarrow{OP} = \overrightarrow{ER}$). Next we store[5] the result in a voxel list $\mathcal{L}$. Such a voxel list can be considered as a displacement list.

Incremental stage For each voxel of the naive plane P_0 we compute the voxels corresponding to the translated line $\mathcal{L}$.

The incremental stage is really easy to implement : this explains that this second algorithm is a little more efficient than the previous one. An important question must be asked : did we miss, with this procedure, some voxels of the discrete tube ? In fact we do not miss any voxel, since we use a 6-connected line and a naive plane. Indeed the intersection of a naive plane and a 6-connected line is at least one voxel [1]. Our algorithm can be considered as a rectangle extrusion. On the contrary, some voxels are identified twice or more. Fortunately it is not a problem : the regular grid mapping of dihedron edges and the diffraction treatment imply to mark the edges that are inside a discrete tube. Indeed an edge can be included in two or more discrete tube voxels, but needs only to be treated once in the diffraction algorithm.

[4] The floor fonction $\mathbb{R} \rightarrow \mathbb{Z}$ returns $\lfloor x \rfloor = \sup(n \in \mathbb{N} \mid n <= x)$

[5] A possible way for optimizing an algorithm is to minimize the dynamic memory allocation. Fortunately it can be noticed that the regular grid size is fixed. Thus it is possible to allocate once and for all this voxel list memory if we consider the greatest 6-connected discrete line : the maximum number of voxels for such a line is smaller or equal to $s_x + s_y + s_z$, where (s_x, s_y, s_z) are the side lengths of the grid.

Algorithm	Total time	Diffraction Computation	Acceleration
Classical	6h 57'	6h 41'	-
DLS	51'	35'	11.5
DRS	42'	26'	15.5

Table 1. *Comparison between the three Diffraction Algorithms for the interior scene*

Algorithm	Total time	Diffraction Computation	Acceleration
Classical	12h 22'	12h 18'	-
DLS	3' 33"	1"	44300
DRS	3' 33"	1"	44300

Table 2. *Comparison between the three Diffraction Algorithms for the car scene*

4 Discussion

These two algorithms have been implemented in our Ray-Tracing software. For the discrete tubes diameter d_r, we use a function of the ellipsoid diameter, which depends on the shortest distance $d(E, R)$ between the transmitter point E and the receiver point R, the wavelength λ of the propagated wave (we take the maximum of the light wavelengths, *i.e.* $\lambda = \max_i(\lambda_i) \simeq 780 \cdot 10^{-9} m$), and the power Φ radiated at E :

$$\frac{\mathrm{d_R}}{2} = 1 + \min\left(1, \frac{\left\lfloor \sqrt{n \cdot \Phi \cdot \lambda \cdot d(E,R)} \right\rfloor}{s}\right),$$

where s is the grid size. We chose such a solution in order to ensure the fact that we do not miss any continuous tube voxels.

Our experiences show that with n = 1 we obtain a good realism. In order to ensure to take almost 100 percent of the diffracted radiance in all cases, we chose to take n = 10 in our implementation (so the ten first ellipsoids). As our images show (*cf.* figure 3), this approximation is good since the human eye can not see any difference[6] between the first one which is computed with the classical diffraction algorithm, and the second one which is computed with our spatial acceleration (DRS and DLS algorithms produce the same results).

The *raison d'être* of these two algorithms is to get an acceleration of the diffraction treatment for a polygonal scene. To study if such an acceleration is obtained, we tested it on different images. The table 1 shows the results for a simple scene, the interior scene (see figure 3). In this case the acceleration factors for our two algorithms are greater than 10. They are relatively small since, firstly, they are only 346 dihedra, and, secondly, the discrete tube diameters

[6] In fact this difference is below 1 percents

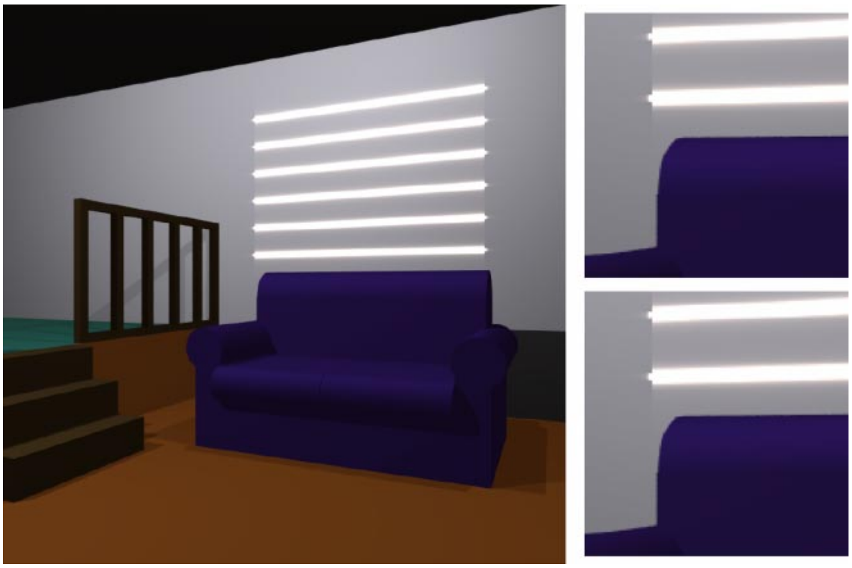

Fig. 3. *The interior scene rendered with (top-right) and without (bottom-right) grid acceleration. We can observe on the wall diffraction by the sun through a shutter. We can observe on the two right images that there are no difference between the previous and the new result.*

are relatively large. Indeed the diffraction is due to the sun, which has a great radiance power ; moreover the distances between the transmitter points and the receiver points are very large since the room lies on earth.

An other test scene (see figure 4) is composed of a car in a closed room. The light transmitter is a lamp inside the room, of relatively poor power (60 Watts). The scene contains 10476 polygons and 15710 dihedra. As the table 2 shows, our algorithms produce an impressive acceleration. The reason of such an acceleration is very simple : the power emitted, the distances between the transmitters and the receivers, and the light power are small ; so the tube diameters are very small too ; therefore we avoid most of the dihedron edges at each diffraction computation because the car is a very complex object, composed of more than ten thousand polygons.

In fact when a scene is composed of many polygons, a regular grid and the discrete tube algorithm lead to a really important acceleration factor.

In order to choose between the two proposed algorithms in an implementation, we need to discuss on which is the fastest one. The figure 5 shows the comparisons of these two algorithms with the interior scene (see figure 3). The x-axis denotes the grid size. It varies between 1 to 100. An attentive reading of this figure shows that the DRS algorithm becomes faster than the DLS one when the grid size becomes larger than 35. Therefore the authors prefer to use the DRS algorithm for its efficiency. In terms of implementation time, this al-

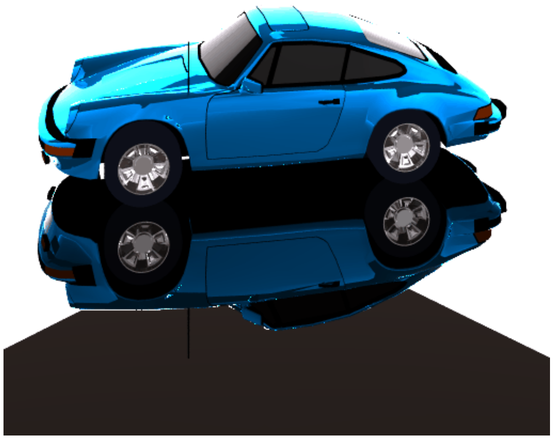

Fig. 4. *The car scene, composed of 10476 polygons which lead to 15710 dihedra.*

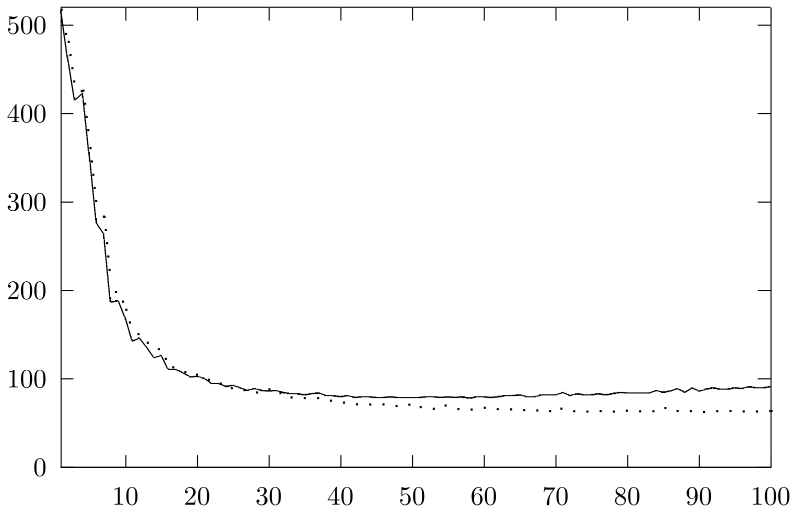

Fig. 5. *Computation times for the interior scene for various grid dimension. Image size is* 512 × 512. *One ray per pixel. The x-axis denotes the grid size. The y-axis denotes the computation time, in seconds. The continuous lines denotes the DLS algorithm. The dotted lines denotes the DRS algorithm.*

gorithm seems also to be the better choice : it is simpler to implement than the DLS one. The problem of the DRS algorithm behaviour is that we can not use an infinit grid size since computer memories have limits.

5 Conclusion

Diffraction is an electromagnetic phenomenon usually ignored in image synthesis. People considered that it was too complex for a low visual importance. Nevertheless some situations imply to render this phenomenon. For that purpose, we have proposed a solution in [2], that we call the Diffraction Algorithm. This method has however a major problem : it is very computation time consuming.

In this paper we present two new algorithms that make diffraction in image synthesis available at a reasonnable computation time cost. They are based on discrete considerations, and their results are quite similar to the classical Diffraction Algorithm. With respect to this last one, they show an acceleration factor that can be really large when diffraction is of low importance in the scene. As a conclusion, it is possible now to include the diffraction treatment into a Ray-Tracing based software : when the diffraction phenomenon is visible in the scene, like for instance in the interior scene, our two algorithms produce good results at a reasonnable cost with respect to the Ray-Tracing cost ; on the other hand when the diffraction account do not induce any more realism, fortunately our new algorithms induce a negligible computation time.

Even though we recommend the use of the DRS algorithm, the DLS one can be extended to produce an acceleration of the pyramidal beam tracing [5]. In order to do this, and like for a classical Ray-Tracing method, we need to find the intersection of a pyramidal beam with the first scene object it meets. The classical method is to test it for all the scene objects. As D. Ghazanfarpour has shown in [5], a better solution is to compute a discrete beam. His method is not based on the discretization of such a beam, but on a grid manipulation that handles too many voxels, and so too many object intersection tests. Authors think that the DLS algorithm can be modified in order to compute such a discrete beam. An important advantage of such a construction would be to detect when an object completely occludes the beam : in these cases it is not interesting to make intersection tests with objects behind the occluder. Moreover this solution leads to less voxel manipulations, so it may be faster.

References

1. Eric Andres, Raj Acharya, and Claudio Sibata. Discrete analytical hyperplanes. *Graphical Models And Image Processing,* 59(5):302–309, September 1997.
2. Lilian Aveneau and Michel Mériaux. Phénomènes ondulatoires en synthèse d'images. *Revue internationale de CFAO et d'informatique graphique*, 12(4):405–425, 1997.
3. M. Born and E. Wolf. *Principles of Optics.* Pergammon Press, New York, 6th edition, 1980.

4. Daniel Cohen. Voxel traversal along a 3D line. In Paul Heckbert, editor, *Graphics Gems IV*, pages 366–369. Academic Press, Boston, 1994.
5. Djamchid Ghazanfarpour and Jean-Marc Hasenfratz. A beam tracing with precise antialiasing for polyhedral scenes. *Computer & Graphics*, 22(1), 1998.
6. Andrew Glassner. *An introduction to Ray Tracing.* Academic Press, 1989.
7. H. Haken, *Light*, volume 1: Waves, Photons & Atoms. North-Holland, 1986.
8. Joseph B. Keller, Geometrical theory of diffraction. *Journal of the Optical Society of America*, 52(2):116–130, February 1962.
9. Robert G. Kouyoumjian and Prabhakar H. Pathak. A uniform geometrical theory of diffraction for an edge in a perfectly conducting surface. *Proceedings of the IEEE*, 62(11):1449–1461, November 1974.
10. Eihachiro Nakamae, Kazufumi Kaneda, Takashi Okamoto, and Tomoyuki Nishita. A lighting model aiming at drive simulators. *Computer Grahics*, 24(4):395–404, August 1990.
11. Radolphe Vauzelle. *Un modèle de diffraction en 3D dans le* 1^{er} *ellipsoïde de Fresnel.* PhD thesis, Université de Poitiers, 1994.
12. Turner Whitted. An improved illumination model for shaded display. *Communication of the ACM*, 23(6):343–349, June 1980.