In 4th Discrete Geometry in Computer Imagery Conference , 1994 , Grenoble , pp65-76

An investigation of Discrete Ray-Tracing

José Delfosse[1,2] , William Terrence Hewitt[3] , Michel Mériaux[1]

LIFL[1]-URA 369 CNRS
Université de Lille 1
59655 Villeneuve d'Ascq FRANCE
{delfosse,meriaux}@lifl.fr
Tel : +33 20-43-47-20
Fax : + 33 20-43-65-66

Department of Computer Science[2]
University of Manchester
M13 9PL , Manchester , UK
delfossj@cs.man.ac.uk
Tel : +44 61-275-6048
Fax : +44 61-275-6040

Computer Graphics Unit[3]
Manchester Computing Centre
M13 9PL , Manchester , UK
w.t.hewitt@mcc.ac.uk
Tel : +44 61-275-6095
Fax : + 44 61-275-6040

Abstract : *We investigate in this paper the problem of discrete ray-tracing. The incorrect intersection phenomena are shown. The need of a specific discretization is discussed according to these problems. We propose a restricted use of a semi-discrete ray-tracer which avoid some problems that arose with discrete ray-tracing.*

1. Introduction

The Ray-Tracing technique introduced by Whitted [Whit80] is a powerful rendering method to modelized very high specular effects with diffuse approximation. Moreover, this technique is also used in radiosity [Gora84][Nish85] in order to mix the best diffuse interreflection from radiosity method with the specular form of Ray-Tracing [Sill89]. Ray-Tracing has become a standard rendering method in the Computer Graphics Community.

However, this recursive technique leads the rendering time to be very expensive. The number of fired rays grow dramatically with the number of pixel, depth of recursivity and number of objects. In the past ten years, researchers have proposed many solutions to improve the computation time of such a method. These methods attempt to reduce the intersection function which is 75 to 90% of the rendering time [Whit80].

Some of the methods rely on the 2D coherency to reduce the calculation such as [Akim91] [Carl89] and [Koic90]. On the other hands, spatial coherency has been investigated to take account of the position of a ray among the objects. The spatial coherency attempts to perform only the intersection between a ray and the objects close to it. We could quote the macro-regions [Devi89], Octree [Glas84] and SEADS [Fuji86]. Others techniques use a ray coherency such as [Arvo87] and [Aman84].Of course, Ray-Tracing has become a classical problem of parallelization. Some studies have been done in this field with [Bado90][Lefe92] and [Keat94].

In this paper, we investigate a spatial coherency proposed by Yagel *et al* [Yage92]. It is in fact a maximum use of a regular subdivision of space proposed by Fujimoto *et al* [Fuji86] (SEADS or also known by Accelerated Ray-Tracing System (ARTS)). The main focus is to reach a very high level of rendering time as showed by Degrande [Degr94]. A lot of topological problems arise due to the discrete geometry. These problems have not been highlighted in [Yage92] but are very important for the final result of the pictures.

2. Discrete Ray-Tracer (DRT)

We present here the *"pure"* Discrete Ray-Tracing as proposed by [Vida92] and [Yage92]. Throughout this paper the acronym DRT means Discrete Ray-Tracing in its pure form.

2.1 The philosophy

Figure 1 means at once the principle. Giving a discrete cube in which the scene is discretized, discrete rays are fired from the eye through each pixel center of the picture in this discrete space so as to get the closest intersection. DRT is therefore carried out in two phases. First of all, all the continuous object are discretized in their voxel form. It is known as the voxelization phase. The rendering phase is then performed by sending discrete rays.

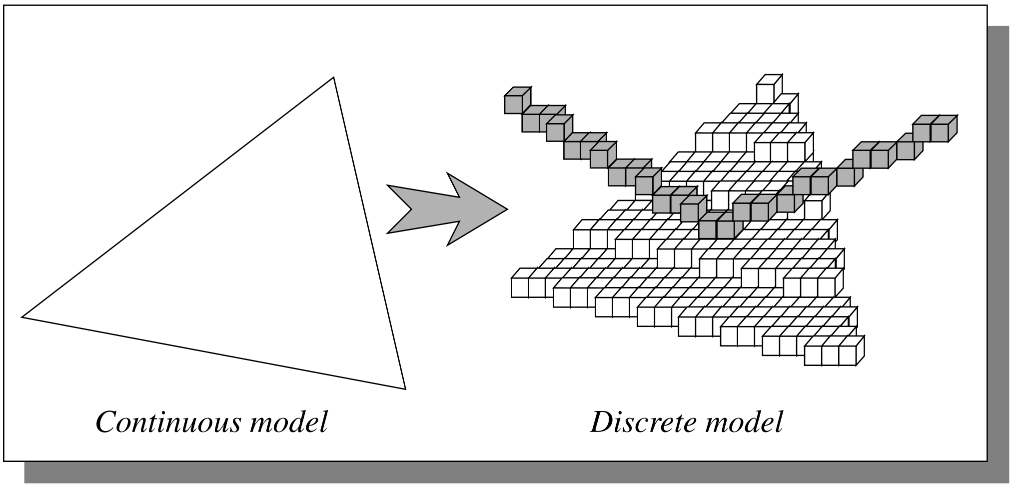

Figure 1 - *Discrete ray tracing*

Since all the operations would have to be completely performed with integer arithmetic, the method *should* work quickly. This philosophy is also *"insensitive"* to the complexity of the scene since there could be an infinite number of objects alongside the ray which will not interfere with its progress. The intersection function of ray-tracing is therefore reduced to a single test of the voxel.

It is important to understand the difference between DRT [Yage92] and ARTS [Fuji86]. The latter subdivides the real space in regular voxel (not necessarely cubic). This voxel must be seen as a real bounding box in which several objects could lie. Hence, ARTS stores in every voxel a list of objects which intersect this one. A parametric ray moves voxel by voxel [Aman87] in the 3D grid and checks the intersection with all the objects in each voxel pierced by it. As against ARTS, DRT is a maximum subdivion technique where a voxel belongs to only one geometrical object. In a matter of fact, it is an integer 3D point and not a bounding box. We will compare anyway these two similar techniques in the result section.

2.2 Choice of a line connectivity

We need to choose an algorithm that realize the walk process throughout the discrete cube. As far as the authors are aware, there are only algorithmic definitions of such a problem. Some definitions are shown in [Vida92] and are an extension of Bresenham's 2D line [Bres65]. The algorithms used in ARTS are from [Aman87][Fuji86] or [Clea88]. Cohen *et al* [Cohe91] give three algorithms according to the desired connectivity.

In order to cross the empty space quickly, Yagel *et al* [Yage92] propose a halo technique. The objects are surrounded by a halo (a flag) which inform the rays that they are close to a surface. As a 6-connected line generates more voxel than a 26-connected one, the ray crosses the cube in a 26-connected form and swaps in a 6-connected line when it hits a halo. It avoids any missed intersection since a 6-connected line is more accurate.

As the ray is 6-connected close to the surface, the wall-pass-through effect (hole in the surface due to the difference of connectivity between the ray and the surface) cannot happens with 18 or 6-connected surface [Vida92]. Therefore, the rasterization of surface must be done in a 18 or 6-connected form.

2.3 Incorrect shadows and illuminations

One of the first problem the reader could notice is the incorrect shadow. Figure 2 (left) highlights the problem. A discrete ray hits a discrete sphere in one voxel. In order to know if this voxel sees a light source, a discrete shadow-ray is fired towards it. Due to the sampling, this ray might re-hit the sphere it was leaving. As figure 2 shows, some voxel should see the light source but do not. It gives plenty of degradations in the final result.

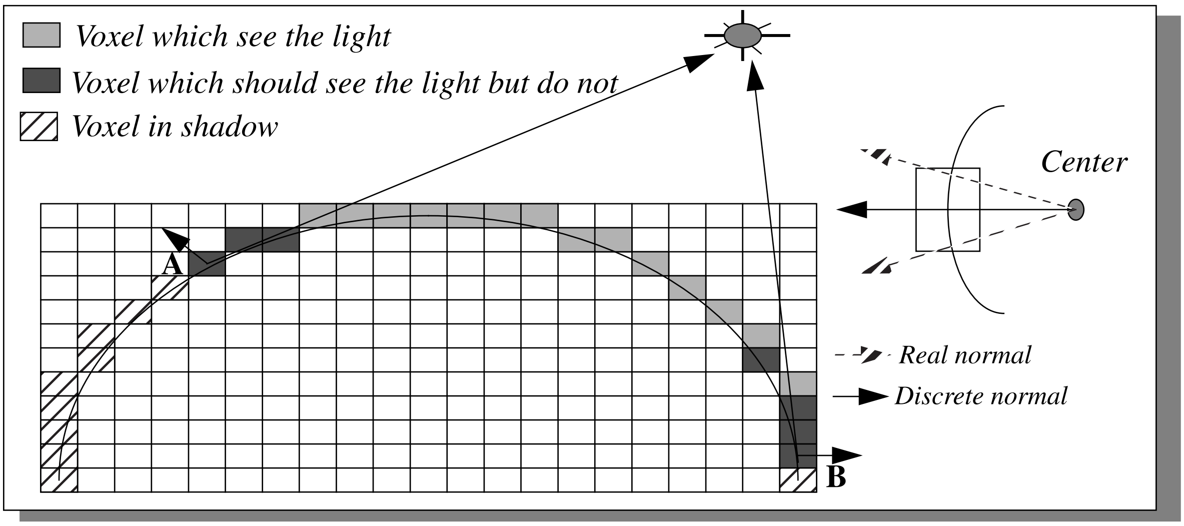

Figure 2 - *Incorrect shadow and normal*

To deal with this problems, the authors use a simple test. If the dot product between the normal to the sphere (or polygon) and the vector towards the source is positive (*N.L>0*), it means that the voxel *can* see the light-source. A discrete ray is therefore fired towards the light source so as to know whether there is an occluder or not. It ***ignores*** all the voxel which belong to the sphere (each voxel has an object number). If the dot product is negative, the voxel cannot see the light-source and a shadow ray is no longer necessary. This simple test fixes the problem explained before. Note there is a problem with the transparent ray which could miss the inside border of the sphere ignoring the voxel of this one.

To compute the normal to the sphere, the difference between the voxel coordinate with the centre of the sphere is performed. It gives therefore an approximation of the real normal since the discrete normal always crosses the centre of the voxel and does not vary. The smaller the voxel is the smaller the error is. Figure 2 (right) shows the problem. As some neihgbour pixels of the picture might genererates close discrete rays, these rays might hit the same voxel. The normal used to compute the light contribution for these pixels [Phon75] will be the same and will give the same colour. The picture will be composed of continuous color plate easily noticeable by the observer. Section 2.5 will take up this phenomenon.

It is also possible to realize a pre-processing shadow. During the voxelization phase, each non empty voxel stores whether it sees the light-sources or not. It requires to fire the shadow-rays for each of these voxels. During the rendering phase, the shadow information will be in each voxel whatever the position of the observer is. It accelerates a lot the rendering phase although the pre-processing is slowed down (voxelization + shadow-ray).

2.4 Incorrrect intersections

Near parallel rays give a bad approximation of the real intersection. Figure 3 shows the problems. The discrete ray hits the discrete polygon in *D* while the real one hits in *I*. From voxel *D*, a shadow ray *Sd* reaches the light-source while from voxel *I*, the shadow ray *Si* cannot see it because of the presence of an occluder. Therefore, the final point on the screen will not be dark but highlighted due to the error. Yagel *et al* [Yage92] say to perform a real intersection test to find the gooddest voxel.

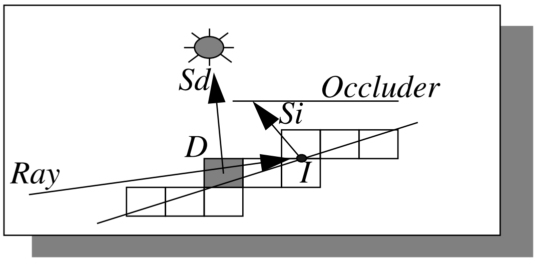

Figure 3- *Incorrect intersection*

2.5 Cube size / Picture size

In order to reduce the problem of aliasing , Yagel *et al* [Yage92] adviced to use a good resolution of cube according to the resolution of the picture. The footprint of a cube voxel must be equal to the picture pixel one. Clearly, we compute a picture n^2 on a cube n^3. Figure 4 shows the variation of information for a varying size picture on a constant size cube.

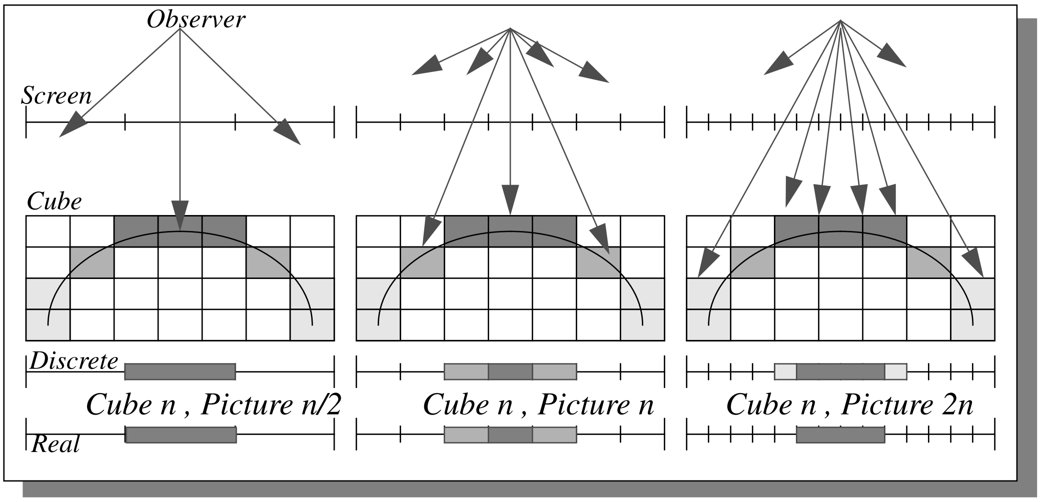

Figure 4 - *The resolution dependancy*

Rays are fired from the observer. They cross the center of each pixel (one-dimension view).

They hit a voxel in the discrete cube and give colour information to the pixel. The final discrete result is put below the cube. The reader will compare with the real result below the discrete one.

The right figure has too much pixel for the resolution of the cube. Clearly, there is not enough information for each ray. Several close pixels hit the same voxel. Left picture has too much information for the resolution of the screen. In that case, there is apparently no problem. A very important question is : What is the cube size m^3 which give a good quality for a picture of size n^2 ? It is hard to answer.

The authors use the same assumptions of [Yage92]. Note in that case, the problem which arose at the end of section 2.3 about the normal approximation (Figure 2) is reduced. Indeed, the error is very small as against the situation of the right picture in Figure 4.

2.6 How to use DRT

To summarize the use of DRT, here are the principal informations :
* Halo technique (26/6 connected line)
* Dot product test for shadow ray and ignoring rays
* Size cube n^3 for a size of picture n^2
* pre-processing shadow
* 6 or 18-connected surface required

3. Semi-Discrete Ray-Tracer (SDRT)

The problems showed in the last section leads to perform a real analytical test as adviced by [Yage92]. However, the authors show it is an incorrect choice and cannot solve the problems of incorrect intersection. Throughout this paper, the acronym SDRT means Semi Discrete Ray Tracing.

3.1 Analytical intersection test

Discrete rays may have an incorrect intersection due to the sampling. In order to avoid them, [Yage92] proposes to check the intersection between the real ray and the real object. Clearly, when a discrete ray hits a voxel, it checks this intersection with a classical analytical test with the real object. If it is good, the real intersection is kept as the intersection point and we compute the *real* normal. If it is not, it carries on until a new discrete intersection

It is apparently a solution which avoid a part of the problems. This approach is very close to the ARTS technique. As we perform a real intersection test, it is no longer a “pure” discrete ray tracer. That is why the authors renames this methods by SDRT.

3.2 The common voxel

Figure 5 shows that as soon as there is objects in contact or in intersection, we need to know in a voxel whether there are several object or not. There is an order during the voxelization phase. If the sphere S_1 has been voxelized before the sphere S_2 then voxel B belong to sphere S_2. Supposes a ray which hits sphere S_2 in A. The real intersection test is false an it carries on ignoring voxel of sphere S_2 as well as voxel B. Finally, the discrete ray crosses sphere S_1. Object intersections lead to get the same problem.

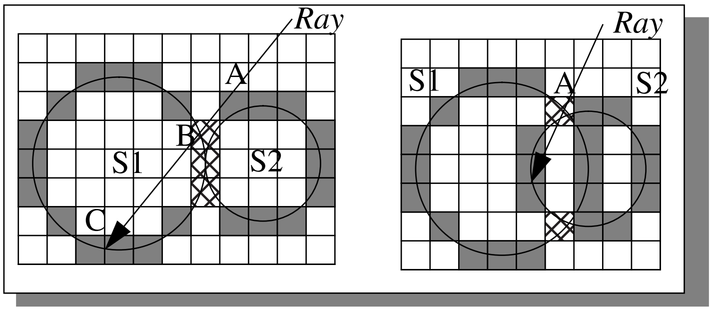

Figure 5 - *Common voxel*

It would be possible to draw several objects which cross the same voxel. In that case, how many objects we should store in a voxel ? The definition of a common voxel which store the objects present in one voxel is necessary to avoid degradations in the final pictures (see plate color).

However, if we need to store all the objects present in a voxel and check every object in it, there is no longer any difference between ARTS and SDRT. The assumption made to use a real intersection test lead us to realize a technique similar to ARTS. SDRT cannot be more efficient than ARTS in this situation. It is therefore useless to carry on in that way.

3.3 Incorrect real intersection test

The common voxel is not the only problem. We need in fact to declare the common objects.Figure 6 highlights the problem. A ray (ray *B*) close to a polygon (polygon 2) may not hit this one on the edge (voxel *B*). The real intersection could be incorrect since the voxel hit is not a common voxel. We need therefore to know in the voxels close to the edge (voxel B) of both polygons that there is another polygon (polygon 1) over there.

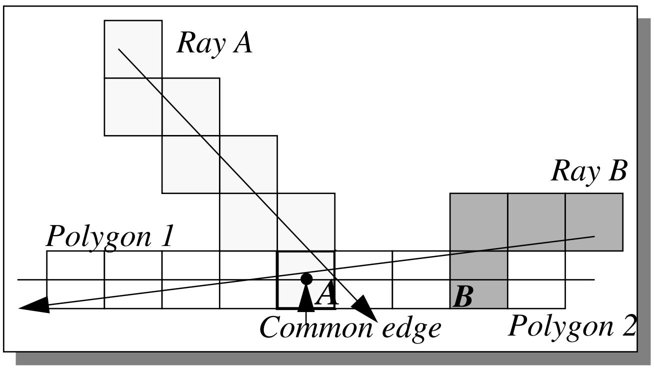

Figure 6 - *Incorrect real intersection*

Clearly, we need to define a common object as against a common voxel. However, this solution cannot solve all the problem. What happens for polygons in intersection without common edges. The only solution which avoid all the problems of SDRT using a real intersection test is in fact the ARTS techniques.

3.4 A restricted use

The authors propose here a restricted use of SDRT which can be seen as a happy medium between DRT,SDRT and ARTS.

As many scenes are composed with polyhedron structure, it is possible to use SDRT with the common polygon definition. A room is composed by triangular polygons that have three neighbours and therefore three common edges. During the voxelization of one polygon, if we are discretizing the same edge of another polygon, the two polygons store in their information section the number of their neighbour. Figure 7 shows an example. Polygon (1) has neighbours {2,3,4}.

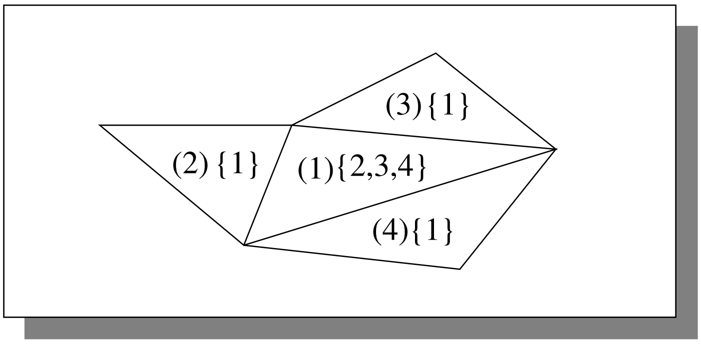

Figure 7 - *Polyhedron structure*

When a discrete ray hits a voxel, it checks the real intersection with the polygon. Most of the time , this intersection is correct. If it is wrong, it means that we are in a situation like figure 6. There are only three possibilities. The discrete ray checks the intersection with neighbour #1. If it is correct, light contribution is performed at this point with the good polygon. If it is incorrect again, it checks with neighbour #2 and so on until all the neighbours have been checked. We are sure of one of these is correct in the case of figure 6. Polygon (1) might have more than three neighbours on the polygon points. If all the intersection with the neighbours #1,#2 and #3 are incorrect then we keep the information of Polygon (1) as the good intersection (The polygon must have at least 3 neighbours). No error are noticeable using a cube n^3 for a picture n^2.The authors call such a technique RSDRT as Restricted Semi Discrete Ray Tracing.

Therefore, if we ensure a polyhedron structure and a nice polygon voxelization, we get the same quality than a real ray-tracer. RSDRT works only on specific scene like room, polygonalized object which are common in computer graphics. The advantage of RSDRT as against ARTS is there is only a maximum of four intersections performed for the ray close to the edges. The better case is one intersection in the middle of a polygon. The number of polygon per voxel in ARTS is depending on the subdivision level and the distribution of the scene.

4. Results

We obtained the results described here by running the ray-tracers on a HP 9000 Model 735/ CRX48Z, Processor RISC PA7100 with a clock speed of 99 MHz, 124 MIPS, 40 MFLOPS (147 SpecMark89) and 112 Mega-bytes of memory.

We compared both techniques with Accelerated Ray Tracing System (ARTS)[Fuji86] since the theoretical acceleration is quite the same (Section 2.1). We use the SPD package proposed by [Haines87] and especially the well-known Fractal Mountains and Sphere Flake scenes.

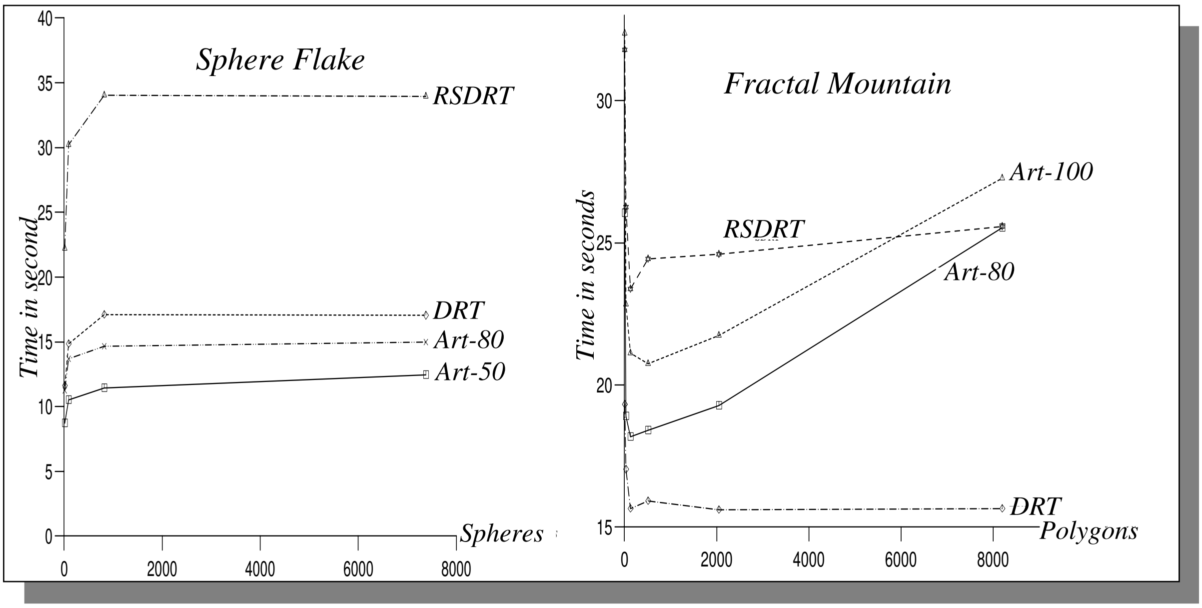

Figure 8 - *Rendering time*

Figure 8 shows that for SphereFlake, RSDRT and DRT are not efficient when compared with ARTS. The cost of intersection time between a ray and a sphere is not counteracted by the discrete speed memory test. However, with Fractal Mountain (only polygons), the slope of the graphs are softer than with ARTS.

The average number of polygon per voxel with Art-80 or Art-100 was greater than 4. RSDRT computed only a maximum of four intersection for the rays close to the edges (Polyhedron structure). It is why RSDRT is faster than ARTS. If we had increase the grid subdivision with ARTS, the average number would have been the same and the walk process throughout the grid would have been longer. The rendering time for ARTS would have been worst.

5. Conclusion

The authors have presented an investigation of discrete ray-tracing. The problems of a direct implementation have been shown.

DRT works within conditions that attempt to reduce the aliasing problem. Whilst the quality is not "good" , the rendering time are very interesting and more investigation about specific hardware would have to be done with such a technique.

SDRT in ist pure form cannot work without incorrect real intersection. The authors have shown it is not sufficient to use a real intersection test to solve the problem. Moreover, a “correct” SDRT tend towards the ARTS technique. Only a restricted use of SDRT may work faster than ARTS for specific scenes. More comparisons between ARTS and RSDRT will have to be done.

Discrete line intersection is surely the most important thing to be done in the future work as well as efficient rasterization of polygon. Indeed, in order to keep the integrity of DRT, it would be interesting to decide the incorrect intersection in a discrete form and "fast".

6. Acknowledgment

The authors would like to thank the staff members of the Computer Graphics Unit (CGU) at the Manchester Computing Centre as well as the Computer Graphics Unit at LIFL.The following persons helped to the realization and clarity of this paper: Samuel Degrande (LIFL), Alan Knowles (University of Manchester) and Eric Andres (University of Strasbourg). This work would not have been possible without the financial support of CNRS.

7. References

[Akim91] AKIMOTO Takaaki,MASE Kenji,SUENAGA Yasuhito,"*Pixel Selected Ray-Tracing*",IEEE Computer Graphics an Applications,July 1991,pp14-22

[Aman84] AMANATIDES J., "*Ray-tracing with cones*", Computer graphics 18(3), SIGGRAPH'84, July 1984, pp129-135

[Aman87] AMANATIDES John & WOO Andrew, "*A fast voxel traversal algorithm for Ray-Tracing"*, EUROGRAPHICS'87, pp 3-10

[Arvo87] J.ARVO, "*Fast ray tracing by ray classification*" SIGGGRAPH July 1987,Vol.21/Numb.3, pp55-64

[Bado90] BADOUEL D., "*Schemas d'éxécution pour les machines parallèles a mémoire distribuée, Une etude de cas : le lancer de rayon*", PhD Thesis 1990, University of Rennes, FRANCE (in french)

[Bres65] J.E BRESENHAM,"*Algorithm for computer Control of a Digital plotter*" IBM System journal, Vol.4/Numb.1, 1965, pp25-30

[Carl89] Carlo H. Sequin, Eliot K. Smyrl, "*Parameterized Ray-Tracing*", SIGGRAPH, July 1989, pp307-314

[Clea88] CLEARY J.G.,WYWILL G., "*Analysis of an algrithm for fast ray-tracing using uniform space subdivision*" The visual Computer, Vol.4, 1988, pp65-83

[Cohe91] COHEN Daniel,KAUFMAN Arie, "*Scan-Conversion algorithms for linear and Quadratic Objects*", IEEE Computer Graphics and Applications, pp280-301, 1991

[Degr94] Samuel Degrande, "*Définition d'une machine céllulaire pour la mise en oeuvre d'un lancer de rayon massivement parallèle*", PhD Thesis, University of Lille 1,July 1994 (in french)

[Devi89] DEVILLERS Olivier, "*The Macro-regions : an efficient space subdivision structure for Ray-tracing*",EUROGRAPHICS'89, pp27-38

[Fuji86] FUJIMOTO Akira,TANAKA Takayuki,IWATA Kansei,"*ARTS: Accelerated Ray-Tracing System*",IEEE Computer Graphics & Applications,April 1986,pp16-26

[Glas84] A.S GLASSNER, "*Space subdivision for fast ray-tracing*" IEEE Computer Graphics&Applications,July 1984,pp14-22

[Gora84] Goral, C.M, K.E Torrance, D.P Greenberg and B.Battaile,"*Modelling the interaction of light-source between diffuse surface*",SIGGRAPH 84, pp213-22

[Hain87] HAINES Eric, "*A proposal for standard Graphics environnments*" ,IEEE

Computer Graphics & Applications,November 1987, pp3-5

[Keat94] M.J KEATES, R.J HUBBOLD,"*Accelerated ray tracing on the KSR1 virtual shared-memory parallel computer*",Technical report 94-2-2, University of Manchester, UK,Department of Computer Science.

[Koic90] Koichi Murakami, Katsuhiko Hirota, "*Incremental ray tracing*", Eurographics workshop on photosimulation,realism and physics in computer graphics, Conference proceedings,June 11-13 1990,Rennes, France

[Lefe92] LEFER Wilfrid, "*Etude de la parallelisation de l'algorithme du lancer de rayon en synthese d'images*" ,PhD Thesis. 1992, University of Caen, FRANCE (in french)

[Nish85] Nishita, T., and E.Nakamae, "*Continuous tone representation of three-dimensional object taking account of shadows and interreflection*", SIGGRAPH 85, pp23-30

[Phon75] PHONG Bui Phong, "*Illumination for Computer generated pictures*" , Communicationsof the ACM, June 1975,Vol.18/Numb.6, pp311-317

[Sill89] SILLION Francois and PUECH Claude, "*A general two-pass method integrating specular and diffuse reflection*",Computer Graphics, SIGGRAPH 89,Vol 23 No 3, pp335-344,August 1989

[Vida92] VIDAL Bruno, "*Vers un lancer de rayons discret*",PhD Thesis 1992, University of Lille, FRANCE (in french)

[Yage92] YAGEL Roni,COHEN Daniel,KAUFMAN Arie, "*Discrete Ray-Tracing*" , IEEE Computer Graphics & Application, September 1992, pp19-28

[Whit80] WHITTED Turner,"*An improved Illumination model for shaded display*" Communications of the ACM, June 1980,Vol.23/Numb.6,pp343-349