Eurographics Symposium on Point-Based Graphics (2005)
M. Pauly, M. Zwicker (Editors)

Point Sampling with Uniformly Distributed Lines

J. Rovira,[1] P. Wonka,[2] F. Castro,[1] and M. Sbert[1]

[1] Institut d'Informàtica i Aplicacions, Universitat de Girona, Spain
[2] Arizona State University, USA

Abstract

In this paper we address the problem of extracting representative point samples from polygonal models. The goal of such a sampling algorithm is to find points that are evenly distributed. We propose star-discrepancy as a measure for sampling quality and propose new sampling methods based on global line distributions. We investigate several line generation algorithms including an efficient hardware-based sampling method. Our method contributes to the area of point-based graphics by extracting points that are more evenly distributed than by sampling with current algorithms.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1. Introduction

In recent years, several algorithms for point-based computer graphics have been developed [27]. These algorithms address multiple aspects of point-based models, such as acquisition, modeling, rendering, compression, storage, smoothing, and water-marking. To extend the existing powerful tool set of point-based computer graphics, this paper contributes an algorithm for generating point samples from polygonal models. The point samples generated by our algorithm can then be used by modeling and rendering tools.

Traditionally, a strong argument given to support point-based computer graphics is the simplicity of the representation. Rusinkiewicz et al. [28] for example, gave a complete framework including sampling, level-of-detail rendering, and compression that was conceptually easy to understand and easy to implement. Dachsbacher et al.[25] showed impressive results for rendering point-based models by creating data structures optimized for hardware rendering.

The goal of this paper is to continue this tradition of simplicity and robustness. Therefore, as input to our algorithm we consider a soup of triangles with unknown connectivity and topology. Our method works in the presence of smooth, manifold surfaces, but it also has to consider a set of disconnected triangles, such as leafs of a tree (as will be demonstrated in section 6).

Existing point sampling methods are directly related to rendering. Therefore, the sampling algorithms are often tailored to a specific rendering method. For example, many splatting methods emphasize overlapping splats in screen space rather than regularity of the sample distribution [28]. The general outline, of many algorithms is to sample a point set and then post-process the point set according to quality criteria imposed by the rendering method. Important representatives are the sampling with layered depth cubes and the following three-to-one reduction [22], or the sampling of geometry into an octree and the displacement of samples along the surface normal [23]. Similarly, level-of-detail algorithms for point sets emphasize specific rendering methods [24, 29].

The nature of our algorithms is fundamentally different, because we set out by analyzing the sample distribution as such, rather than the optimization for one specific algorithm. Additionally, we want to create a framework that is applicable to general models such as vegetation. In this context, we cannot assume smooth, continuous surfaces. We also want to constrain the samples to lie on the actual polygons.

The main contribution of this paper is a theoretical analysis of point sampling and the introduction of several meth-

ods to create regularly distributed sampling points, including a fast hardware assisted sampling implementation. We believe that the proposed sampling algorithms, will be an important complement to existing algorithms in a point-based computer graphics toolbox.

2. Overview

We set out to build a theoretical framework that compares different sampling strategies. The first important question that we will address is how to compare the quality of sampling methods and how to measure uniformity. In section 3 we will address this question and propose discrepancy as a measure for sampling uniformity. In section 5 we will review several sampling algorithms based on integral geometry and show their evaluation and results in section 6.

3. Discrepancy as a measure of regularity

The discrepancy [11, 20, 16] can be viewed as a quantitative measure for the deviation of a finite set of d-dimensional points from a totally even (regular) distribution (or, in other words, as a measure of the irregularity of the distribution). There are several formulations of the discrepancy. One of them is star-discrepancy, which is defined with respect to the family of subsets of I^d. Given a set of points $C = x_1, \ldots, x_n$ of I^d, we can define their star-discrepancy in the following way

$$Disc^*(C) = \underset{A \subset I^d}{max} \left| \frac{n(A)}{n} - V(A) \right| \tag{1}$$

where A is any d-dimensional cube in I^d that contains the origin, $n(A)$ is the number of points that belong to cube A and $V(A)$ is a normalized measure of the size of cube A. That is, the star-discrepancy is the maximum difference between the relative number of points of a cube containing the origin and its relative size.

Thus, the lower the discrepancy of a set of points, the more regular their distribution. In general, sets of points generated using pseudo-random generators (the ones provided by the programming languages) have a higher discrepancy than the ones generated using quasi-Monte Carlo sequences (see [11]). In fact, quasi-Monte Carlo sequences have been specially designed to minimize the discrepancy (this is the reason they are also called low discrepancy sequences). Quasi-Monte Carlo sequences were introduced in Computer Graphics by Alexander Keller [19].

It can be shown [11, 20] that a set of N points generated using pseudo-random values has a star-discrepancy $O(\sqrt{\frac{loglogN}{N}})$, whereas if we consider quasi-Monte Carlo sequences the star-discrepancy behaves as $O(\frac{(logN)^d}{N})$, where d is the dimension. Note that the discrepancy grows as the dimension grows, but, since $N^{-1} < N^{-1/2}$, there is always a value of N from which on this discrepancy is lower than the one for pseudo-random generation.

4. Intersection with random lines

The basic tool to study intersections with random lines is Integral Geometry [10]. Integral geometry allows us to study and measure sets of lines, for example how many lines intersect a convex body, how many intersect a surface, etc. Integral Geometry defines a uniform density of lines that is homogeneous and isotropic (invariant under rotations and translations). An embedded body K is intersected by these lines. The moments of the chord lengths have been well studied for convex bodies.

The measure of the lines crossing a convex body K is given by [9, 10]:

$$\int_{K \cap G} dG = \frac{\pi}{2} A \tag{2}$$

where G represents the uniform lines, dG its measure, and A holds for the area of K.

For a general non-convex body K equality (2) generalizes to

$$\int_{K \cap G} n_G dG = \pi A \tag{3}$$

where n_G is the number of intersections of line G with object K.

The resulting intersections with K are uniformly distributed over the area of the surface. This can be seen in the following way. Consider a differential surface from K, dA. According to (2) the measure of lines crossing it is πdA (we can consider it as a cylinder with height 0 and base dA). But this is also the measure of the number of intersections.

Thus, we can sample points on the surface of a body using in a uniform way by using uniformly distributed lines. These lines can be obtained in several ways (see section 5.1) The most simple case is enclosing the body in a 3D sphere and selecting random pairs of points on the surface. (see Fig.1a). The number of intersected points on area dA follows a binomial distribution given by $Bin(N_r, \beta)$, where $\beta = \frac{2dA}{A_s}$ is the probability of a line crossing dA, A_s is the area of the surrounding sphere, and N_r is the number of lines cast. This distribution can be approximated by a Poisson distribution with mean $N_r\beta$. In the case of generating lines from a general convex bounding box [15] (see section 5.4) A_s is substituted by the area of the box.

Lines can also be obtained using bundles of parallel lines (see section 5.3, Fig.1b,2). The average number of intersections is now given by $\frac{dAN}{2\Delta}$ where N is the number of bundles

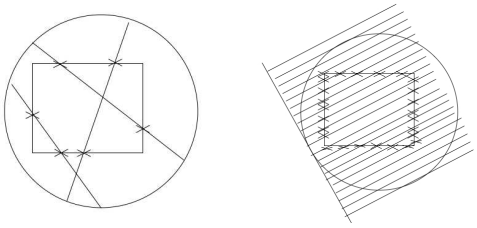


Figure 1: *Points are generated from the intersections of lines (left) and bundles of parallel lines (right).*

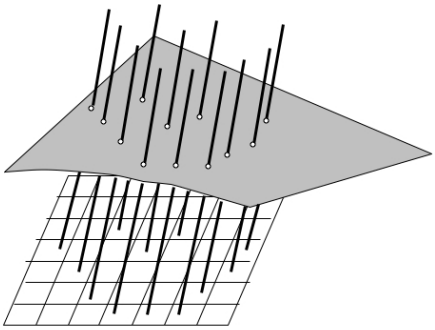


Figure 2: *Generating points from intersections with a bundle of parallel lines.*

and Δ is the area of the cell of the bundle (see section 5.3). The number of intersected points on area dA can be thus described by a Poisson distribution with mean $\frac{dAN}{2\Delta}$.

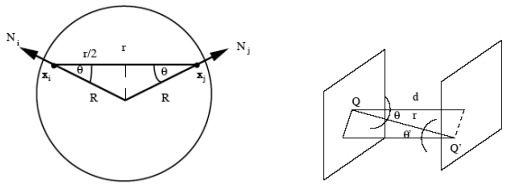
5. Line density generation

We show here different alternatives to generate a global line density (where global has the same meaning as uniform). The first 3 methods involve sampling points uniformly distributed on the surface of a sphere. Next we describe the algorithm to generate such points on sphere with center c and radius r

Generate 2 values ξ_1,ξ_2 uniformly distributed in $[0,1)$
$cos\theta = 1 - 2 * \xi_1$
$sin\theta = \sqrt{1-(cos\theta)^2}$
$\varphi = 2 * \pi * \xi_2$
$vDir = (sin\theta * \sin\varphi,\ cos\theta\ , sin\theta * \cos\varphi)$
$udPoint = c + r * vDir$

5.1. Two random points on a sphere surface

In [10] it is shown that the density of global lines intersecting a convex body K (that is, density of chords) is given by $\frac{\cos\theta\cos\theta'}{r^2} d\sigma d\sigma'$, where θ, θ' are the angles of the intersecting line with the normals in the intersected points, $d\sigma, d\sigma'$ are area differentials in the same points, and r is the length of the chord. This density, for a sphere, becomes simply (except a constant factor) $d\sigma d\sigma'$ (see figure 3a).



(a) (b)

Figure 3: *(a) Geometry for two points on sphere line generation. (b) Geometry for incorrect line generation.*

This means that taking pairs of random points in the sphere surface we obtain a global uniform density of lines. But observe that this is only valid for a sphere, this is, taking pairs of points on the surface of a convex body does not result in a uniform density (except of course for the sphere). To see how one can deviate from the uniform density, let us consider taking pairs of points on opposite faces of an orthogonal prism [6]. The correct density should be proportional to $\frac{\cos\theta\cos\theta'}{r^2} d\sigma d\sigma'$, the one taken is proportional to $d\sigma d\sigma'$. The ratio of the densities is proportional to $\frac{\cos\theta\cos\theta'}{r^2}$. Now, observe figure 3b, from this figure $\cos\theta = \cos\theta' = \frac{d}{r}$, thus the ratio is proportional to $\frac{1}{r^4}$. This means for instance that for twice the distance we cast 2^4 more lines that necessary. In other words, much more lines are cast in oblique directions than in orthogonal ones.

Observe also that the sphere density is equivalent to taking a single point on the sphere and a direction from this point weighted according to the cosine of the angle θ between the radius at this point (this is the same to say the normal to the tangent plane) and the direction. Thus, taking simply a uniformly distributed direction does not result in a uniform density of lines.

The sphere density is described in [13], and was first used for Radiosity in [12] and in IBR in [1]. Interestingly, this uniform density generation has not a counterpart in 2D. This is, taking pairs of points uniformly distributed on a circumference does not provide a uniform density within the circumference [13].

5.2. Random direction and point in main circle

This is the generation that appears in classic IG books [9, 10, 13] and it has been used in IBR by Camahort et al. [1]. (See figure 4c). It is obviously equivalent to selecting a tangent plane (thus a point in the sphere), and a point in the projection of the sphere on the plane.

5.3. Bundles of parallel lines

If one limits randomness in the previously defined density to selecting the tangent plane and taking the points on the plane on a regular grid (see figure 4d), one obtains bundles of parallel lines. This density (see figure 4d) has been used in the Radiosity context in [15, 14, 8, 17], and in IBR in [1, 7]. A point to remark with this density is that the expected number of lines crossing a planar polygon with area A given N bundles of lines is $\frac{AN}{2\Delta}$, where Δ is the area of the grid cell. Another point is that the grid cell can be any parallelogram. Also, to avoid aliasing, the origin of the grid can be jittered [15]. The advantage of using bundles of parallel lines is that fast projection algorithms as the z-buffer can be used.

5.4. Lines from the walls of a convex bounding box

Using the fact that the differential of solid angle around $d\sigma$ can be written as $d\omega = \frac{\cos\theta'}{r^2} d\sigma'$, we can transform the chord density into $\cos\theta d\omega$. This means taking a random point on the surface of the convex bounding box and a cosine weighted direction (see 4b). This density generation was first described in [15]. It is useful as we can use the same bounding box of the scene to generate the lines. Observe that this is not the same as taking pairs of uniformly distributed random points on the bounding box surface, which, as seen above, is incorrect.

6. Results

6.1. Sampling points evenly distributed on a polygon

Our experiments are intended to study the distribution of the points sampled on a polygon† by means of embedding it in a uniform density of lines, as described above, and computing the intersections between this polygon and the lines. Given sets of points obtained in this way, we have computed their star-discrepancies and compared their values corresponding to different line generations, including pseudo-random number generators and quasi-Monte Carlo sequences. On the other hand, lines have been generated using two different techniques, both involving a bounding sphere:

- Sampling pairs of points on the surface of the sphere (see 5.1).

† For practical purposes we use a square, but the results are valid for any planar or non-planar shape.

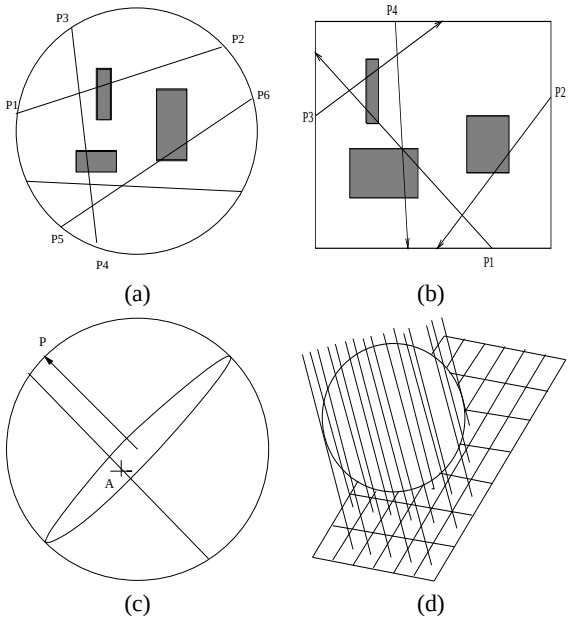


Figure 4: *Different ways to simulate uniform global lines. (a) Two random points on the surface of the sphere. (b)* Local *lines are cast from the walls of convex bounding box. (c) Random direction defining a disc by the bounding sphere center, and random point in this disc. (d) Tangent plane by a random point on the sphere surface and bundle of parallel lines perpendicular to this plane.*

- Sampling tangent planes (to the sphere) and generating a bundle of parallel lines from each plane (using a grid) (see 5.3).

We have to remark that, since the uniform density of lines is known to be invariant under rotations and translations, the quality of the results (that is, of the point sets) is independent on the position of the polygon.

We have also compared the sets of points mentioned above with sets of points sampled directly on the polygon using pseudo-random numbers and quasi-Monte Carlo sequences. Next we present the results obtained in our experiments.

NOTE: In the case of point sets generated using pseudo-random number generators (the ones used in Monte Carlo integration), the values of the star-discrepancy have been computed by averaging the results obtained in several independent tests.

6.1.1. Points sampled directly on the polygon

We have sampled N points directly on the polygon using pseudo-random number generators and different quasi-Monte Carlo sequences like Halton, scrambled Halton, Sobol, Weyl and Hammersley ones (for a description on these quasi-Monte Carlo sequences, see [11, 20, 4]). Note

that, for each point, 2 values in the interval $[0,1)$ are required.

Fig. 5 shows $N = 1000$ points generated using these techniques. Star-discrepancies have been computed for each of these 6 sets of points, and values obtained using quasi-Monte Carlo sequences have been near one order of magnitude lower than using pseudo-random generators. The Hammersley sequence has performed specially well.

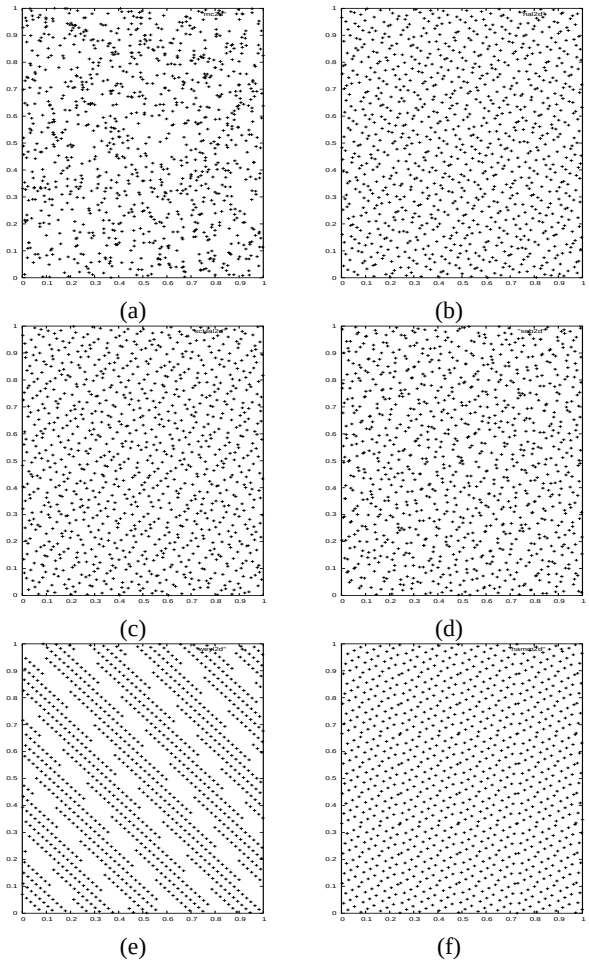


Figure 5: *1000 2D points sampled directly on the polygon using: (a) Pseudo-random numbers. (b) Halton sequence. (c) Scrambled Halton sequence. (d) Sobol sequence. (e) Weyl sequence. (f) Hammersley sequence.*

We have experimented with different values of N for each of the generations. In the graph in Fig. 6 we represent N (x-axis) vs. star-discrepancy (y-axis). Values of the discrepancy corresponding to pseudo-random generation are in all cases more than one order of magnitude higher than the ones obtained using quasi-Monte Carlo sequences, and so those values have been removed from the graph to avoid scale distortion. Comparing the different quasi-Monte Carlo sequences, the Hammersley sequence behaves the best according to star-discrepancy, followed by scrambled Halton and classical Halton ones. Finally Sobol and Weyl sequences behave the worst in this experiment.

Referring to asymptotical behavior, we note that star-discrepancy of the 2D-point sets seems to follow the expected $O(\frac{log^2 N}{N})$ for the quasi-Monte Carlo generation in dimension 2, but this is not true for pseudo-random (Monte Carlo) generation, where star-discrepancy decreases slowly (its expected behavior is $O(\sqrt{\frac{loglogN}{N}})$).

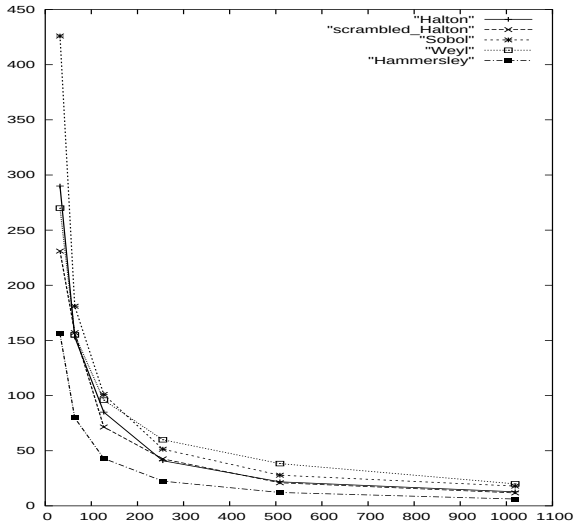


Figure 6: *Number of points N times* 10^{-3} *(x-axis) vs. star-discrepancy times* 10^6 *(y-axis) for different point sets generated directly on the polygon. The curve corresponding to pseudo-random generation has been removed from the graph to avoid scale distortion, since these values are more than one order of magnitude higher than the rest.*

6.1.2. Points sampled using lines obtained from pairs of points on a bounding sphere

Now the polygon has been bounded by a sphere. A uniform density of lines has been generated in the sphere by means of sampling pairs of points on its surface (see 5.1). For each line that intersects the polygon, the intersection point has been considered, obtaining in this way the point set. Note that in this case we deal with 3D-points, but since such points belong to a polygon, they can be easily mapped on a 2D space. Note also that 4 values in the interval $[0,1)$ are required for each line. Thus the required dimension is 4 instead of 2.

The same pseudo-random generation and quasi-Monte Carlo sequences experimented in the previous section have been used here. Sets of 1000 2D points have been generated. Referring to star-discrepancies of these point sets and comparing them with the ones obtained using direct generation

of the points (see section 6.1), the main fact observed is that, while the discrepancy obtained using pseudo-random numbers is very similar to the one obtained in 6.1 (with this generator), the discrepancy obtained using quasi-Monte Carlo sequences is clearly bigger (about 3 times) than the one obtained in 6.1. This involves a reduction of the advantage (referred to discrepancy) of quasi-Monte Carlo sequences in front of pseudo-random generation when point sets are obtained not by direct sampling on the polygon but intersecting it against sets of uniformly distributed lines. Hammersley and Halton sequences behave the best in this experiment.

We have tested different values of N for each of the generations. In the graph in Fig. 7 we represent number of points N (x-axis) vs. star-discrepancy (y-axis). A higher discrepancy for pseudo-random generated sets is observed in almost all cases, but the differences respect to quasi-Monte Carlo generation have been reduced regarding to the ones obtained in section 6.1.1. We note also that Weyl sequence behaves clearly worse than the rest of quasi-Monte Carlo sequences (that present a similar behavior) in this experiment.

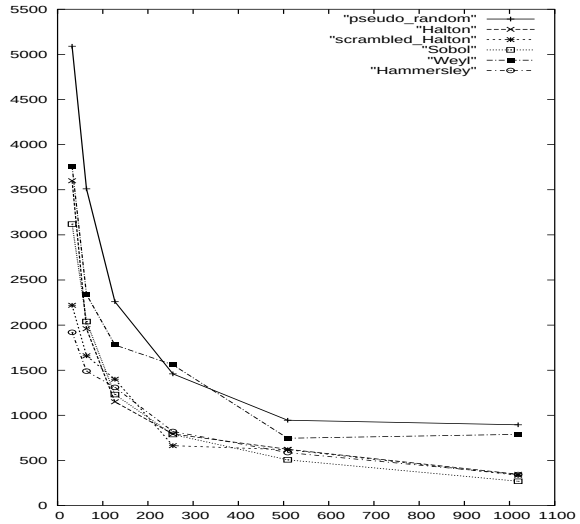


Figure 7: *Number of points N times* 10^{-3} *(x-axis) vs. star-discrepancy times* 10^6 *(y-axis) for different point sets generated by intersecting the polygon against sets of uniformly distributed lines generated in a bounding sphere by sampling pairs of points on its surface.*

Finally, and regarding to the asymptotical behavior, we note that here the star-discrepancies of the 2D-point sets generated using quasi-Monte Carlo sequences seem to follow the expected $O(\frac{log^4 N}{N})$ for the quasi-Monte Carlo generation in dimension 4, and thus the discrepancy decreases slower than directly generating the points on the polygon (note that in pseudo-random generation the discrepancy behavior does not depend on the dimension).

6.1.3. Points sampled using bundles of parallel lines generated from tangent planes

This technique (see section 5.3) also involves generating lines and computing their intersections with the polygon. We also need a bounding sphere. The randomness relies on sampling, for each of the N bundles, a tangent plane (whose normal vector constitutes the direction of all the lines in the bundle), and also an origin point for a regular $n \times n$ grid on the plane (whose cells constitute the origin of the n^2 lines in the bundle). This involves sampling (for each bundle) a point on the sphere surface that sets the tangent plane, and a point on the plane that sets the origin of the regular grid. That is, 4 values in the interval $[0,1)$ are required for each bundle of lines.

A first question to be solved when dealing with this generation is the relation between N, the number of bundles (planes) and n, the number of linear subdivisions in the grid. From some previous tests, we have considered $N = n$ a good relationship, and all the experiments presented in this section are based on it. Note that with this relationship the total number of lines used is N^3. Further experiments have shown to be optimal the relationship $N = \frac{3}{4}n$.

The first experiment, like in the previous sections, consists of generating approximately 1000 points on the polygon using pseudo-random values and the quasi-Monte Carlo sequences used previously. Star-discrepancies are in general lower than the ones obtained in the previous section. Fig. 8 compares the $2D$ projections of such point sets (for pseudo-random generators and the Halton sequence) with the ones obtained when using lines from pairs of points on a bounding sphere (see 6.1.2). The rest of quasi-Monte Carlo sequences behave in a similar way.

In Fig. 9 we present a graph in which x-axis represents the number of sampled points and y-axis represents the star-discrepancy. From observing this graph, the first conclusion is that all the generations, including pseudo-random numbers, present similar values for the star-discrepancy. Thus, there is not any significant difference between pseudo-random numbers and quasi-Monte Carlo sequences when using bundles of parallel lines. This behavior probably corresponds to the lower influence of the random factor in this generation, due to the use of the regular grid.

We have also represented in this graph the values of the star-discrepancy obtained using direct generation of the points in the polygon (see section 6.1.1) and pseudo-random numbers, in order to compare them with the corresponding values using bundles of parallel lines. The interesting point is that the discrepancy appears to be clearly lower (up to 3 times) in this last case. This means that for pseudo-random generators (the ones provided by the computers) is more appropriate using the bundles of lines technique than directly sampling the points in the polygon. We can visually confirm this behavior by comparing Fig. 5 (a) and Fig.8 (a): this last

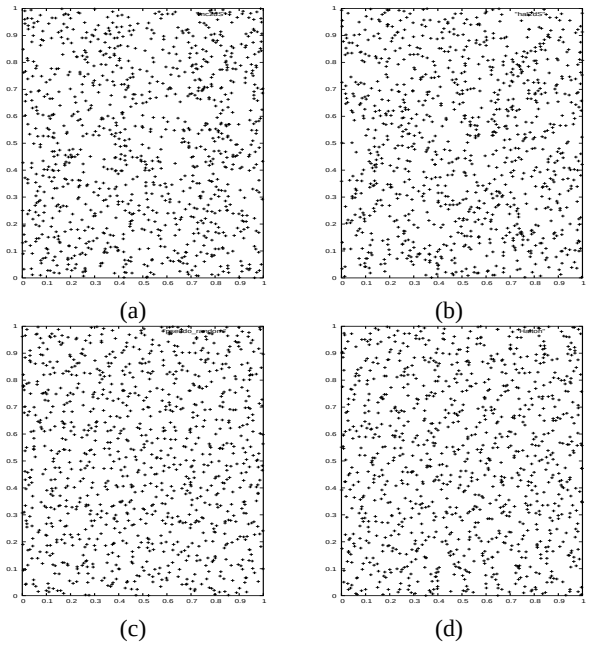


Figure 8: *(a) and (b) show 1000 2D points obtained from the intersection of the polygon against lines from pairs of points sampled on a bounding sphere using pseudo-random numbers in (a) and the Halton sequence in (b). (c) and (d) also show 1000 2D points obtained now from the intersection of the polygon against bundles of parallel lines.* 20 *bundles have been used in this experiment, and tangent planes and origin points for the corresponding grids have been obtained using pseudo-random numbers in (c) and the Halton sequence in (d). Note the higher regularity of the distribution when using bundles of parallel lines.*

set of points appears to be more evenly distributed than the first one.

On the other hand, there is a noticeable reduction of the star-discrepancy when using this generation respect to the values obtained when generating the lines from pairs of points sampled on the surface of the bounding sphere (see previous section). Star-discrepancy is reduced to approximately one half with quasi-Monte Carlo sequences and to approximately a third part when using pseudo-random values. These results show this generation technique to be superior than the technique involving sampling pairs of points on the sphere (at least regarding to the discrepancy of the obtained point sets). This fact is shown, for the case of Halton sequences, in Fig. 10.

Regarding to the asymptotical behavior, we note that here, like in the previous section, the star-discrepancy values corresponding to quasi-Monte Carlo sequences seem to follow $O(\frac{log^4 N}{N})$ as expected for dimension 4.

Systematic sampling for tangent planes

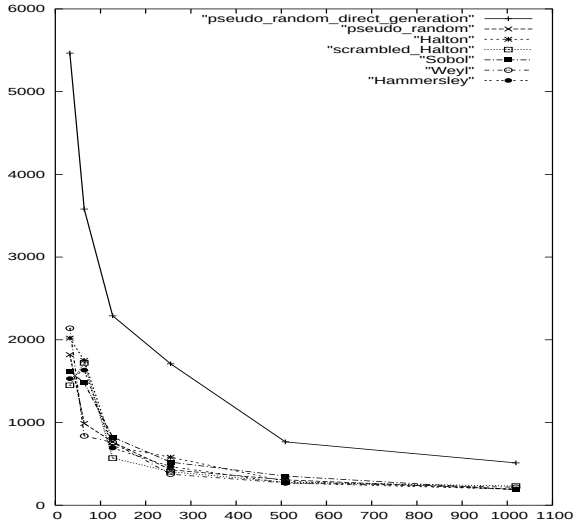


Figure 9: *Number of points times* 10^{-3} *(x-axis) vs. star-discrepancy times* 10^6 *(y-axis) for different point sets generated by intersecting the polygon against bundles of parallel lines generated using tangent planes and regular grids. We have also represented the discrepancy for the point sets obtained using pseudo-random values and direct generation on the polygon.*

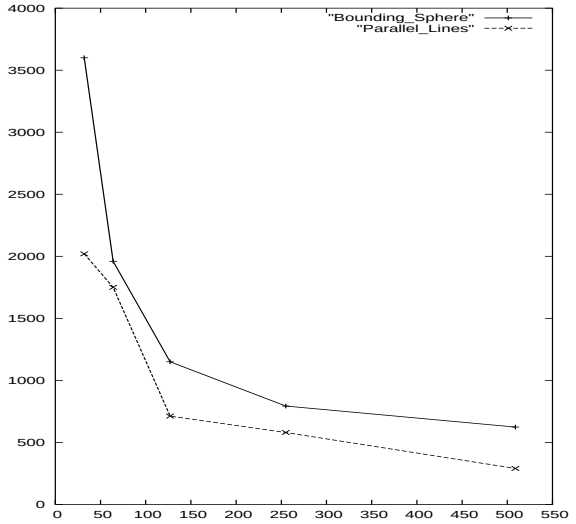


Figure 10: *Number of points times* 10^{-3} *(x-axis) vs. star-discrepancy times* 10^6 *(y-axis). Generation of lines using pairs of points on a bounding sphere is compared with the tangent planes technique, which offers better results. Halton sequences have been used in both cases.*

We have tested the generation of the tangent planes using systematic sampling (that is, using a regular grid mapped on the sphere to obtain the points that set the tangent planes). The results obtained do not improve the ones obtained without this kind of sampling. Thus, this strategy has been discarded.

Use of only 3 *axis-aligned planes*

Another experiment has been done in the sense of using only 3 axis-aligned planes. In this case, the regularity of the distribution of the points strongly depends on the position of the polygon respect to such planes, that is, depends on the direction of the normal to the polygon (the more similar this direction to the direction of one of the 3 planes, the worse the results). This makes this strategy to be unsuitable for a complex environment in which the polygons are supposed to have any direction.

6.2. Hardware assisted sampling

We have implemented a tool to sample meshes using bundles of lines with the depth-peeling algorithm ([26]). The algorithm works as follows:

For each direction of extraction (generated with Monte Carlo), the first step is to obtain the intersections of the mesh with the bundles of lines. Depth-peeling obtains a set of layers from a scene by using a front depth buffer updated along multiple render passes. These layers effectively contain the intersections with the bundle of lines along the viewing direction.

The second step of the algorithm is to reconstruct the 3D positions and additional information (color, normal, etc.) of the samples on each layer, by using both the rendered buffer and the depth buffer of each layer. This second step is performed in the CPU and involves processing all the pixels on each layer and generating the point samples.

The tool is very fast and works at interactive rates for the models used in this paper.

6.2.1. Implementation results

We have tested our algorithm with two models. The first model, Venus, consists of 43.357 polygons and the second model, Tree, contains 32.196. Our tool allows to select the number of directions and the resolution of the bundles. The tests have been done on a Pentium-M 1.5 notebook with a GeForce FX 5650.

In figure 11 (right) we show the results of sampling the Venus model with 12 bundles. The time to obtain the point cloud is 0.78 seconds. For comparison purposes, in figure 11 (left) we show the result with sampling with only 3 mutually orthogonal directions (as in [22]), but containing the same number of rays than the 12 bundles (using denser bundles). The time to obtain this point cloud is 0.22 seconds. Observe the much better point distribution in the right image obtained at not much higher cost.

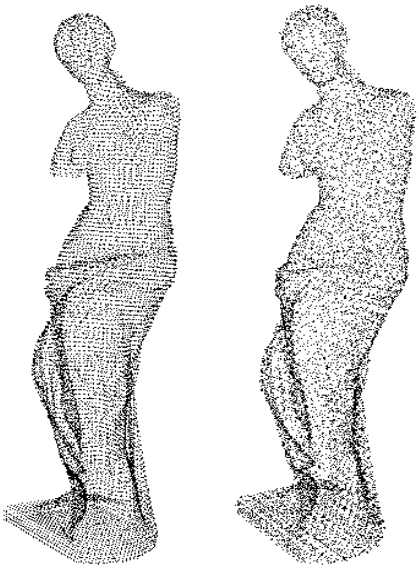


Figure 11: *Point clouds obtained using 3-axis projection (left) and using Monte Carlo to determine the directions (right). Both contain around 12K points.*

Another benefit of using hardware rendering to obtain the ray intersections is the possibility to use texturing to apply transparency masks to the polygons. With this algorithm you can sample only the used area of each polygon without a noticeable extra cost. This is useful for plant models where the detail of the shape of the leaves is added with a partially transparent texture. See figure 12 for an illustration of this idea.

Figure 12: *This figure illustrates another benefit of hardware-based sampling. Textured polygons (left) can be sampled according to an alpha-texture and samples are only created only at opaque surface points automatically.*

6.2.2. Algorithm efficiency notes

The effect of the mesh size on the efficiency of our algorithm is lower than in most CPU-based algorithms. Instead, the processing of the layers done in CPU takes most of the time. A possible optimization would be to encode the world position in the color buffer, as it would make unnecessary to use the depth buffer to re-transform from camera-space to world-space in CPU. This is only possible if rendering to floating-point buffers is supported.

Another difference with CPU-based algorithms is that extracting information from a sample is limited to the capacity of the render buffers. For this reason, to extract all the information like the color, the normal, etc., more than one buffer might be needed and thus, more than one rendering pass for each layer and for each direction. CPU-based algorithms can extract all the information when computing the ray-mesh intersections, but they are still slower.

6.2.3. Hardware limitations

There are some sources of error on our hardware-based algorithm.

The first one comes from the depth-peeling precision. The depth of each pixel in the front depth buffer is encoded with limited numerical precision, and as a consequence, inadequate peeling can happen for surfaces which are very close. This limitation can be almost eliminated by using floating-point precision buffers supported in the current generation of hardware.

The second source of error comes from the resolution used to extract the layers. When reconstructing the world position of the samples using the render buffer and depth buffer, each pixel actually represents a whole parallelepiped in the space. Setting the position in its center is an approximation that can give large errors if the render resolution (density of the bundles) is low. Encoding the world position in the color buffer instead as explained in the previous section overcomes this problem. Current hardware maximum rendering resolutions is around 4096 pixels, for this reason denser line bundles would require splitting the process and doing multiple depth-peeling extraction for each direction.

7. Conclusions and future work

We have studied the distribution of point sets obtained by intersecting polygons against a set of lines uniformly distributed around the polygons. This procedure allows to easily generate such point sets on a large polygonal model using the same set of global lines instead of generating each point set individually in each polygon.

We are interested in generating sets of points that are the most regularly distributed on the polygons. As a measure of the regularity of the point sets we have used the star-discrepancy: the lower the star-discrepancy, the more regularly distributed the points. This value tends to decrement as the number of points (and so the number of lines) grows. Thus, given a certain number of lines, we aim for obtaining the lowest values of the star-discrepancy.

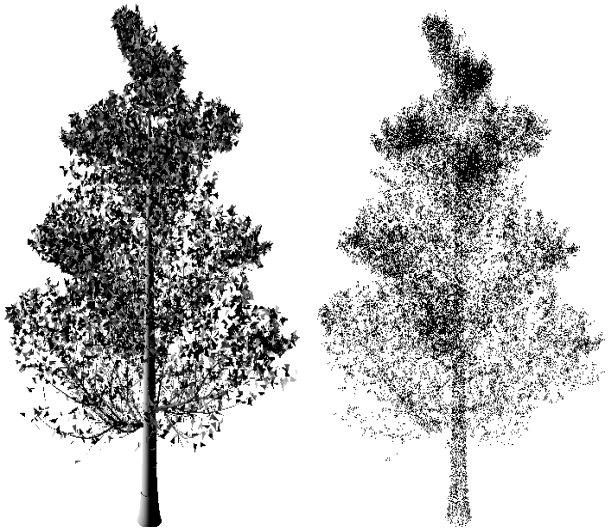


Figure 13: *Tree mesh model (left) and a 44K point cloud extracted with our algorithm (right) in 3.41 seconds.*

According to our experiments, the discrepancy values obtained using bundles of parallel lines are noticeably lower than the ones obtained with lines generated from pairs of points sampled on a bounding sphere (see Fig. 10). Regarding to the generation of the $[0,1)$ values, there are no important differences between pseudo-random numbers and quasi-Monte Carlo sequences when using bundles of parallel lines (some quasi-Monte Carlo sequences seem to be just a bit superior than pseudo-random numbers). Conversely, when generating the lines from pairs of points on the sphere, quasi-Monte Carlo sequences happen to be clearly superior than pseudo-random numbers.

Another interesting result is that, unlike quasi-Monte Carlo sequences, pseudo-random numbers behave clearly better (that is, produce point sets with lower discrepancy) when generating the points using bundles of parallel lines than when directly sampling the points on the polygons (see section 6.1.3).

Finally, we have implemented the bundles of parallel lines on a hardware-based tool that is able to extract the point set of a polygonal model at interactive rates.

As a future work we plan to investigate the use of systematic and adaptive sampling of directions to further improve the discrepancy.

Acknowledgments

This project has been funded in part with grant number TIN2004-07451-C03-01 from the Spanish Government, and GameTools Project (number IST-2-004363).

References

[1] Emilio Camahort, Apostolos Lerios and Donald Fussell "Uniformly Sampled Light Fields", *Rendering Techniques'98*, pp.117–130, Springer–New-York, 1998.

[2] Francesc Castro, Roel Martínez, Mateu Sbert "Quasi Monte Carlo and extended first shot improvement to the multi-path method for radiosity", *Proceedings of SCCG'98*, Budmerice, Slovakia, April 1998.

[3] Francesc Castro, Mateu Sbert, "Application of quasi-Monte Carlo sampling to the multi-path method for radiosity", 3rd International Conference on Monte Carlo and quasi-Monte Carlo Methods in Scientific Computing, Claremont, USA, June 1998 (to appear in Springer series Lecture Notes in Computational Science and Engineering, Springer Berlin, 1998).

[4] Francesc Castro, Mateu Sbert, "Quasi-Monte Carlo techniques in Multipath radiosity", Homenatge a L.Santaló. Ed. Universitat de Girona.

[5] M.Feixas, E.Acebo, P.Bekaert, M.Sbert, "An Information Theory Framework for the analysis of scene complexity", Eurographics'99, Milan, 1999.

[6] Marc Levoy and Pat Hanrahan, "Light Field Rendering", *Proceedings of Siggraph 96*, pp.31–42, LA, August 1996

[7] Dani Lischinski and Ari Rappoport, "Image-Based Rendering for Non-Diffuse Synthetic Scenes", *Rendering Techniques'98*, Springer–New-York, 1998.

[8] Laszlo Neumann, "Monte Carlo Radiosity", *Computing*, 55(1):23–42, 1995

[9] J.Rey-Pastor and Luis Santaló Sors, *Geometría Integral*, Espasa-Calpe, Madrid, 1951

[10] Luis A. Santaló. *Integral Geometry and Geometric Probability*. Addison-Wesley, New York, 1976.

[11] Harald Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. Capital City Press, 1992.

[12] Mateu Sbert, "An Integral Geometry Based Method for Fast Form-Factor Computation", *Computer Graphics Forum*, Vol 12 N. 3, pp.409–420 , 1993, (Eurographics'93)

[13] H.Solomon, *Geometric Probability*, volume 28 of *CBMS -NSF Regional conference series in applied mathematics*. SIAM, Philadelphia, PA, 1978.

[14] Mateu Sbert, Xavier Pueyo, Laszlo Neumann, and Werner Purgathofer, "Global multipath Monte Carlo algorithms for radiosity", *The Visual Computer*, 12(2):47–61, 1996.

[15] Mateu Sbert. *The use of global random directions to compute radiosity. Global Monte Carlo methods*. Ph.D. thesis, Universitat Politècnica de Catalunya, Barcelona, March 1997. Available in *http://ima.udg.es/~ mateu*

[16] Francesc Castro *Efficient Techniques in Global Line Radiosity*. Ph.D. thesis, Universitat Politècnica de Catalunya, Barcelona, Desember 2002. Available in *http://ima.udg.es/~ castro*

[17] Laszlo Szirmay-Kalos and Werner Purgathofer, "Global ray-bundle tracing with hardware acceleration", *Rendering Techniques'98*, pp.247–258, Springer–New-York, 1998.

[18] Frederic Cazals and Mateu Sbert, "Some Integral Geometry tools to estimate the complexity of 3D scenes", *INRIA Research Report*, n.3204. July 1997.

[19] Alexander Keller, "Quasi-Monte Carlo Radiosity", *Proceedings of Eurographics Workshop on Rendering*, pp.102-111. 1996.

[20] Alexander Keller, "Quasi-Monte Carlo Methods for Photorealistic Image Synthesis", *Ph.D. thesis*. University of Kaiserlautert. 1997.

[21] Greg Turk, "Re-Tiling Polygonal Surfaces", *Computer Graphics (ACM SIGGRAPH 92 Conference Proceedings)*, Vol. 26 N.2. pp.55-64, 1992

[22] Pfister, H., Zwicker, M., van Baar, J. and Gross, M., "Surfels: Surface Elements as Rendering Primitives", *ACM SIGGRAPH 2000 Conference Proceedings*, pp. 335-342, 2000.

[23] Mario Botsch, Andreas Wiratanaya and Leif Kobbelt, "Efficient high quality rendering of point sampled geometry", *Proceedings of the 13th Workshop on Rendering*, pp. 53-64. 2002.

[24] Jianhua Wu and Leif Kobbelt, "Optimized Sub-Sampling of Point Sets for Surface Splatting", *Computer Graphics Forum*, 23(3), pp. 53-64. 2004.

[25] Carsten Dachbacher and Marc Stamminger, "Rendering Procedural Terrain by Geometry Image Warping", *Rendering Techniques*, pp. 103-110. 2004.

[26] Cass Everitt, "Interactive order-independent transparency", *Technical report, NVIDIA Corporation, May 2001.*, 2001

[27] Marc Alexa, Markus Gross, Mark Pauly, Hanspeter Pfister, Marc Stamminger, and Matthias Zwicker, "Point-Based Computer Graphics", *SIGGRAPH 2004 Course Notes*, 2004

[28] Szymon Rusinkiewicz and Marc Levoy , "QSplat: A Multiresolution Point Rendering System for Large Meshes", *ACM SIGGRAPH 2000 Conference Proceedings*, 2000

[29] Mark Pauly, Markus Gross, Leif Kobbelt "Efficient Simplification of Point-Sampled Surfaces", *IEEE Visualization 2002*, 2000