Density-Controlled Sampling of Parametric Surfaces Using Adaptive Space-Filling Curves

J. A. Quinn[1], F. C. Langbein[1], R. R. Martin[1], and G. Elber[2]

[1] Cardiff University, UK
{J.A.Quinn,F.C.Langbein,Ralph.Martin}@cs.cardiff.ac.uk
[2] Technion, Israel
Gershon@cs.technion.ac.il

Abstract. Low-discrepancy point distributions exhibit excellent uniformity properties for sampling in applications such as rendering and measurement. We present an algorithm for generating low-discrepancy point distributions on arbitrary parametric surfaces using the idea of converting the 2D sampling problem into a 1D problem by adaptively mapping a space-filling curve onto the surface. The 1D distribution takes into account the parametric mapping by employing a corrective approach similar to histogram equalisation to ensure that it gives a 2D low-discrepancy point distribution on the surface. This also allows for control over the local density of the distribution, e.g. to place points more densely in regions of higher curvature. To allow for parametric distortion, the space-filling curve is generated adaptively to cover the surface evenly. Experiments show that this approach efficiently generates low-discrepancy distributions on arbitrary parametric surfaces and creates nearly as good results as well-known low-discrepancy sampling methods designed for particular surfaces like planes and spheres. However, we also show that machine-precision limitations may require surface reparameterisation in addition to adaptive sampling.

1 Introduction

Many applications in geometric modelling and computer graphics require generation of evenly distributed points on surfaces. An even point distribution is important for two reasons: (i) to avoid aliasing artefacts which might be caused by regularly-spaced samples, and (ii) to provide efficiency in surface calculations—if the distribution is uneven, more samples are needed to guarantee a minimum point density. We discuss and analyse an efficient, practical algorithm to generate such point distributions on parametric surfaces in 3D, based on an approach suggested by [1]. The method also allows the user to control the local point density and further corrects for handling extreme parametric distortion.

Local density control is useful for applications such as *surface triangulation*, where we may want more samples in regions of higher curvature [2]. Existing applications where stochastic/low-discrepancy distributions are already used include radiosity [3] and ray tracing [4]. Another application is *point-based rendering* where surfaces are represented by point sets and are rendered employing

splats (small discs) which can be quickly generated using hardware shaders [5]. This provides a simple and efficient approach to non-photorealistic rendering and rendering of complex or dynamic surfaces [6]. Efficiently distributing, resampling and parameterising suitable point distributions on a surface is essential for interactivity and visible artefact reduction [7]. Stochastic methods are also widely used for measurement and quality control applications, e.g. to compute volume integrals and surface curvature of complex shapes. Other examples are *meshfree finite element analysis* [8, 9], and *re-meshing* of meshes with a known parameterisation [10]. Applications based on low-discrepancy sampling typically require far fewer samples than a random approach, so the effort put into generating these samples will likely become worthwhile when performing repeated surface calculations such as physical simulations or when using dynamic surfaces.

We seek an *even* distribution of points on a surface, perhaps with respect to a specified density function. In particular we seek a *low-discrepancy* point set: discrepancy is a measure of the deviation of a point set from a uniform distribution. It is computed locally for a subset by taking the absolute difference between the ratio of points lying inside the subset and an area measure of the subset (e.g. the surface integral over the density function). The discrepancy of a point set is the supremum of this local discrepancy. A point set is of low-discrepancy if its discrepancy is minimal. Often the subsets considered are restricted to certain types such as rectangles [11, 12]. A point set of low discrepancy covers the surface as evenly as possible and often has the desirable property that there are no large 'holes' in the distribution, while at the same time avoiding aliasing problems [4].

In this paper we describe an approach to generating low-discrepancy point distributions on parameterised surfaces by distributing points along a space-filling curve as suggested by Steigleder and McCool [1]. By generating a *space-filling curve* in the parameter domain, the problem of distributing points in 2D is reduced to sampling a curve appropriately. Sample points are placed along the space-filling curve using an idea similar to *histogram equalisation*. Adaptive generation of the space-filling curve allows us to handle parametric distortions where, e.g., a small area in the parameter domain is mapped onto a large area on the surface. However, for parameterisations in which extremely large areas of the surface are spanned by very small ranges of parameter values, we encounter limitations caused by the machine precision, as explained in Sect. 5.3.

The space-filling curve not only converts the 2D problem to a 1D problem, but also provides the additional benefit of good spatial localisation of the points, in the sense of knowing which points in the output sequence lie near which other points in the sequence [13]. This is clearly of advantage for problems which require, for example, rapid determination of the k-nearest neighbours of a given surface point.

We first review previous work and describe the algorithm. Then we briefly demonstrate the technique in the plane directly following [1], and evaluate the results obtained. Comparing the generated point sets to known low-discrepancy distributions indicates that they are close to the best, known low-discrepancy distributions, with consistently good results for various test shapes used to measure

discrepancy, and varying numbers of points. Most low-discrepancy sequences are optimised for discrepancy measures based on rectangular subsets, and while the approach is not quite as good as some other low-discrepancy methods for rectangular discrepancy, for the other-shaped discrepancy measures tested, it is better. We next demonstrate the effectiveness of the approach for sampling arbitrary parametric surfaces by computing discrepancy on the unit sphere with respect to spherical triangles. This provides strong evidence that this approach is a fast and effective way of low-discrepancy sampling on the sphere. Estimating the discrepancy for general surfaces in $\mathbb{R}^3$ is rather harder as it not only requires the calculation of exact surface areas, but also choice and construction of sampling shapes (such as a triangles) to assess the distribution on each particular surface. However, we give visual examples showing our methods applied to more general surfaces, with and without explicit user-provided density control functions. We then demonstrate how adaptive curve generation helps considerably on shapes with extreme parameterisations, but that it is ultimately not always sufficient, and that surface reparameterisation may also be required.

2 Previous Work

In this section we discuss previous approaches to generating low-discrepancy distributions in the plane and on parametric surfaces. We start by briefly reviewing general work that prompted our research, and then then discuss work more specifically related to our approach.

Niederreiter and Sobol sequences have been theoretically shown to produce an optimal low-discrepancy sequence [14] for rectangular subsets of $2D$ manifolds with discrepancy varying as $O(N^{-1}\log^2 N)$ for N points. They have been shown to be considerably better than random sampling in Monte-Carlo techniques both theoretically [15] and in practice for various geometric problems [16] and have also been applied to problems such as rendering [17]. In order to evaluate the quality of the point distributions generated by the current algorithm, we compared our results to low-discrepancy distributions generated by ACM TOMS Algorithms 738 [18] and 659 [19]. We also compared the results to a random distribution and a jittered distribution on the plane [20]. Generating independent pairs of random numbers gives a random distribution in the unit square [21] with expected discrepancy $O(N^{-1/2})$ [22]. Base-2 Hammersley distributions on the unit sphere were generated using the algorithm described in [23].

Our approach uses an adaptive version of an algorithm suggested by Steigleder and McCool [1] to generate density-controlled stratified samples in n-dimensions and employ them to sample surfaces. In this paper we provide experimental evidence that this approach allows the user to generate high quality, density-controlled distributions on arbitrary surfaces, correcting for parameteric distortions by adaptively generating the curve in the parameter domain. However, if the parametric distortion is too great, reparametrisation of the surface may also be required, as we discuss later. The sample points are produced along the curve using a technique similar to histogram equalisation.

Our algorithm uses a method similar to histogram equalisation to distribute points along a space filling curve lying in a surface to achieve the desired density distribution based on approximating the area of small surface patches (see Sect. 3). Other 1D inversion methods exist, such as the one presented in [24], but they do not generalise to arbitrary surfaces or provide the same utility as a space-filling curve.

Although algorithms such as [18] are available for generating low-discrepancy point sets in n-dimensional parallelpipeds starting from given seeds, we are not aware of methods specifically for generating low-discrepancy point sets on arbitrary surfaces in 3D, and which also allow the user to control the point density via a function. However, methods for generating such point-sets are known for specific surfaces. For example, [25] uses lines between low-discrepancy points on the surface of a sphere to calculate mass properties of objects, and [23] demonstrates the generation of Halton and Hammersley low-discrepancy sequences on the sphere. A similar approach using intersecting lines is used by [26] to generate a point cloud from a mesh. The first two papers use specific methods to generate points on the sphere, and do not readily generalise. The third provides no evidence that the point distribution generated from the mesh is of low discrepancy on the surface. Hartinger [27] describes a generalisation of a quasi-Monte Carlo technique originally proposed by [28] to generate points in the plane with an arbitrarily chosen distribution for computing integrals.

3 Point Distribution Algorithm

In this section we describe how to generate a low-discrepancy point set on a parametric surface S. Given a surface parametrisation $f : [0,1]^2 \to \mathbb{R}^3$ of S with the unit square as normalised parameter domain, a non-negative bounded density function $\delta : S \to \mathbb{R}_0^+$, and a desired number of points N, our algorithm generates a set of points p_l, $l = 1, \ldots, N$, distributed on S according to the density δ. More precisely, we generate a set of parameter values representing these points. We desire that, for each subset A of the surface S, the fraction of points p_l lying inside A should be as close as possible to the ratio between the surface integrals $\int_A \delta \, ds / \int_S \delta \, ds$ to ensure that the point set has low discrepancy (with respect to the desired density). For the special case $\delta \equiv 1$, the points are uniformly distributed with respect to surface area.

The basic idea of Steigleder and McCool's algorithm [1] is to convert the 2D distribution problem into a 1D distribution problem, by placing points along a space-filling curve that covers a square, which for our application is the parameter space. An approximation to a space-filling curve is generated in the unit square, and mapped onto S. This approximation can be described by a sequence of vertices v_l, $l = 0, \ldots, M$, lying on the curve in the surface. We then create a 1D point distribution q_k, $k = 1, \ldots, N$ in $[0,1]$ where the distances between these points indicates some desired initial distribution in the unit square (choices for this distribution will be discussed later, but, for example, we could use equal-distance points). This distribution is then mapped onto the 2D surface using

similar ideas to histogram equalisation: points p_l, $l = 1, \ldots, N$ are distributed on the curve in the surface so that they have approximately the same distances as the q_l in $[0, 1]$ in the plane. However, to ensure that fractional arc-lengths are preserved on the curve in the surface (rather than the parameter domain), we must measure the distances according to the local surface geometry. At the same time, we add a further factor to produce the desired point density function δ on the surface, by further adjusting the distance measure.

As space-filling curves map a 1D line onto a 2D surface, measuring distances as lengths along the curve is ill-defined. Instead, we approximate the area of the surface that is covered by a certain fraction of the space-filling curve, and use this as the distance measure on the curve which indicates how many points should be distributed on a segment of the curve. Each of the points v_l can be associated with a small patch A_l of the surface for which we approximate the ratio $s_l = \int_{A_l} \delta \, ds / \int_S \delta \, ds$. The s_l together with the v_l form a discrete distribution which indicates the desired local density of points. By calculating the cumulative sum of s_l as we move along the curve, we integrate over the discrete distribution indicating how the number of the points we wish to distribute has to be increased. This allows us to estimate the distances between the points on the curve in the surface and distribute them according to the distances between the q_l in $[0, 1]$. The rest of this section presents the main algorithm in more detail, especially concerning measuring distance on the curve. Then we describe various options for generating initial point distributions and the histogram equalisation.

Pseudocode for the overall point distribution algorithm is given in Fig. 1. It consists of the following main components: a space-filling curve generator, Fig. 2, that computes a set of vertices lying on an approximation to a space-filling curve for the unit-square; a generator which computes a 1D sequence of real numbers in the unit interval $[0, 1]$ indicating how the points are to be distributed on the curve; and an equalisation method to distribute points on the surface along the space filling curve by measuring distances along the curve based on the local surface area and the density function. First, the curve is generated, the parameterisation is applied and surface area is calculated along the curve. We then generate a 1D sequence of points, and map them to the curve, according to the integral of the surface area, taking into account a surface density function and the level of recursion of the curve.

First we call a recursive function PopulateCurveTree to generate an approximation of a space-filling curve in the unit square. A tree is generated using this method according to a specified curve type C (different space-filling curves can be used in principle, such as the *Hilbert Curve*, the *Hilbert II Curve*, etc.). We use a tree to facilitate the subdivision of the unit square, and after all subdivision, the final curve vertices are generated and stored in the leaf nodes.

The unit square is subdivided into a number of cells centred around each curve vertex (dependant on the curve). The method AssessSubdivision applies the supplied parameterisation f to each cell, and calculates the distances from the centre point to the corners and half-way points along each side. The maximum distance is taken, and compared to the supplied cell size a. If the

Algorithm DistributePoints (f, δ, r, N, C)
Input: f—parametrisation $f : [0,1]^2 \to \mathbb{R}^3$ for surface S
δ—density function $\delta : S \to \mathbb{R}_0^+$
r—maximum recursion depth for space-filing curve approximation
a—parameter for the minimum required area associated with a curve vertex v
N—number of points that should be distributed on S
C—type of space-filling curve
Output: A low-discrepancy point distribution on S according to δ, given as a list of parameter values for f

1. $n \leftarrow$ GenerateRootNode(C)
2. $T \leftarrow$ PopulateCurveTree (n, r, a, 0)
3. $[v_1, \ldots, v_M] \leftarrow$ Output(T)
4. $S_0 \leftarrow 0$
5. **for** $l \leftarrow 1, \ldots, M$ **do**
6. $\quad S_l \leftarrow S_{l-1}$+Density ($f$, δ, v_l)
7. $[q_1, \ldots, q_N] \leftarrow$Create1DPoints($N$)
8. $[p_1, \ldots, p_N] \leftarrow$Equalise($[q_1, \ldots, q_N]$, $[v_1, \ldots, v_M]$, $[S_1, \ldots, S_M]$)
9. **return** $[p_1, \ldots, p_N]$

Fig. 1. Point Distribution Algorithm

value is greater than a and the current depth g is less than the recursion cap r, a true result is returned and GenerateChildren is called on the node. r should be set according to machine precision, or as a limit on the maximum memory requirements of the process. GenerateChildren subdivides the cell, generating the number of children required according to the space-filling curve C, and applies the appropriate permutation. Permutations represent an ordering of the vertices in a given cell, and are calculated according to a set of simple rules based on the ordering of the parent cell and which child the current cell is.

If no subdivision is required, a final index, representing the cell's position along the curve is computed from the permutation and stored. When all subdivision is complete, a method Output is called which scales the cell indices to the maximum level of curve recursion used and hashes the values stored in the leaf nodes of the tree to a set of vertices v_l, $l = 1, \ldots, M$ indicating the line segments (see Fig. 1, line 3). In order to accurately estimate the distances between points on the curve in the surface for the distribution, M has to be sufficiently large (see Sect. 5.3). A suitable value for a can be chosen as half the required maximum distance between vertices of the space-filling curve on the surface. Appropriate choice of space-filling curve is discussed later in this section.

Next we compute the desired cumulative density of the point distribution by mapping each v_l onto the surface and estimating the ratio $s_l = \int_{A_l} \delta \, ds / \int_S \delta \, ds$ for a small surface patch A_l, using a method explained shortly. The s_l are approximated by the function Density (Fig. 1, line 6). For each point $f(v_l)$ on the surface we get a cumulative density $S_l = \sum_{k=1}^{l} s_l$, $l = 1, \ldots, M$ (Fig. 1, lines 4–6). As we have an approximation of a space filling curve, we have $S_M \approx \int_S \delta \, ds$,

Algorithm PopulateCurveTree (n, r, a, g)
Input: n—root node for the specified curve
r—max space-filling curve recursion depth
a—parameter for the minimum required area associated with a curve vertex v
g—current depth of recursion
Output: A space filling curve defined by n, in a tree structure

1. $b \leftarrow$ AssessSubdivision(a)
2. **if** b is true and $g < r$
3. $[c_1, \ldots, c_N] \leftarrow$GenerateChildren($n$)
4. **for** $c \in [c_1, \ldots, c_N]$ **do**
5. PopulateCurveTree(c, r, a, $g + 1$)
6. **else**
7. Curve Index Stored in Node

Fig. 2. Curve Generation Algorithm

if we choose suitable A_l. This means that the overall surface integral of δ is a scaling factor that can be ignored for distributing points as we know the number of points we intend to distribute. Hence, to approximate s_l we only approximate $\int_{A_l} \delta \, ds$.

We have considered several different methods to approximate this integral: (a) compute the area of the triangle $f(v_{l-1})$, $f(v_l)$, $f(v_{l+1})$ and multiply this area by the value of the density function at the centroid of this triangle; (b) use a small square centred at $f(v_l)$ instead of a triangle, constructed from extra vertices generated in the parameter domain with a size equal to a line segment of the curve; and (c) use $\delta(f(v_l))\sqrt{\det f_I(v_l)}$ where f_I is the first fundamental form of the surface, as $ds = \sqrt{\det f_I}\, du\, dv$. In general all three approaches work well, although (b) and (c) produced slightly superior results to (a), as demonstrated in Sect. 5.1. This may be partly due to the fact that for (a) the triangles overlap. For the examples reported in this paper we demonstrate results for (a) and (c). However, note that, ideally, the first fundamental form of the surface should be used and calculated automatically [29]. Also note, that for approach (c), we have to scale the results using: $A'_l = A_l(w^{k_{max}-k_{current}})$, where $k_{\max}$ is the maximum and $k_{current}$ the current depth of the space-filling curve recursion and w is the number of vertices in a curve 'cell' (i.e. the number of curve vertices where $k = 1$). This enables us to normalise the area value for the non-uniform curve.

After we have computed the cumulative density distribution S_l over the v_l, we generate a set of 1D points q_k, $k = 1, \ldots, N$ in $[0, 1]$ to provide some chosen initial distribution in the unit square (as explained later) by mapping these points on the space-filling curve in the parameter domain (Fig. 1, line 7). The S_l over the v_l describe how to find arc-lengths along the curve: the distance between $f(v_l)$ and $f(v_{l+1})$ on the curve in the surface is given by $S_{l+1} - S_l$ (the desired density). Thus, in the next step we map the distances between the q_k according to the the cumulative density distribution S_l over the vertices $f(v_l)$,

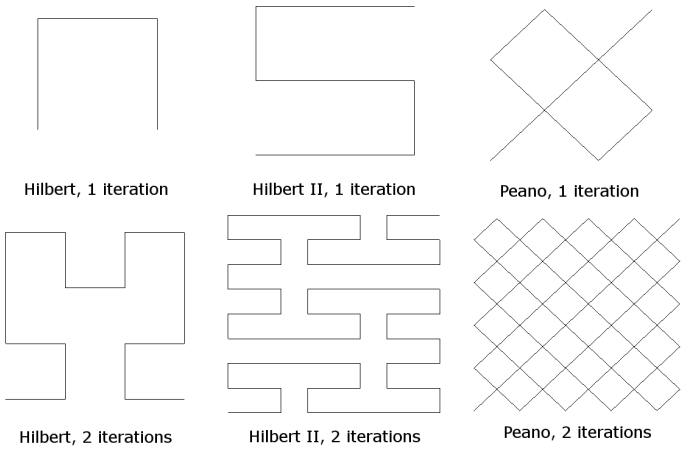

Fig. 3. The first two iterations of the Hilbert, Hilbert II and Peano curves

$l = 1, \ldots, M$ on the curve in the surface via an equalisation approach (Fig. 1, line 8): a subset $p_l, l = 1, \ldots, N$ of the v_l is chosen such that the distances between the p_l correspond to the distances between the q_l.

We now briefly describe in greater detail the space-filling curve approximations used in the first step of our algorithm. Various space-filling curves were investigated, including the Peano, Hilbert and Hilbert II curves [30] (see Fig. 3). Our adaptive sampling algorithm was used to generate these curves. However, as results in Sect. 4 demonstrate, the Hilbert curve is the space-filling curve of choice. If a uniform Hilbert curve is required, for use with very simple parameterisations, the algorithm described by Butz [31] and expanded by Lawder [32] can be used: it provides a very efficient way of generating points on the Hilbert curve using bit operations. In [1], *Jittered equally-spaced points* in the unit interval $[0, 1]$ were mapped onto the Hilbert curve. We generalised this technique and tested various methods for generating the 1D point sequences in the unit interval, mapped as fractional arc-lengths: *1D low-discrepancy sequence*, *Equally-spaced points*, *Jittered equally-spaced points* and *Randomly spaced points*. We expected the Jittered and Uniform points to demonstrate the best results, although one drawback to placing evenly spaced points on the curve is that if the number of points is commensurate with the number of vertices on the curve, we may get aliasing artefacts. Jittering the points removes the possibility of this occurring. For this reason, and because of the results discussed in Sect. 4, the third method is the method of choice for our final algorithm.

4 Experiments in the Plane

We performed some initial experiments in the plane to determine whether the choice of space-filling curve had a significant effect on the point distributions

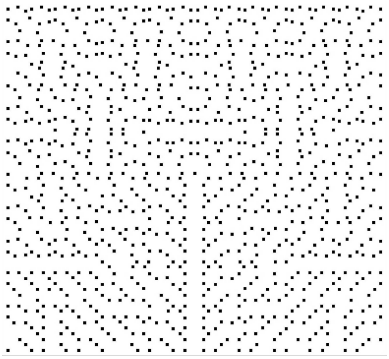

Fig. 4. Hilbert Curve 'Gaps'

produced—even though Steigleder and McCool[1] suggest the use of a Hilbert curve, other space filling curves can also be used. These expand on the experimental results in [1] which showed how the star discrepancy scales for point sets of varying size, by comparing the point distributions generated in the unit square with the reference point distributions listed in Sect. 2, determining the variation of discrepancy with number of sample points in each case. The star discrepancy is the most commonly employed measure due to its simplicity; it computes the discrepancy of a point set in the unit square based on axis-aligned rectangular subsets with one corner fixed to the origin. We also investigated generalisations of the star discrepancy, more specifically using triangles and quarter-circles, as in many applications, portions of surfaces of interest need not simply be rectangular.

4.1 Evaluation of Space-filling Curves

Various choices exist for the space-filling curve. To determine the impact of this choice on the quality of the point distribution, we initially assessed the generated point sets both visually, and using experimentally measured discrepancy. Visually we aimed to ascertain any obvious regularities which might cause aliasing problems, or obvious gaps indicating high discrepancy—the human eye is very good at recognising patterns and holes in data.

Hilbert curve: The Hilbert curve showed large vertical and horizontal gaps in the distribution (see Fig. 4) when evenly-spaced or low-discrepancy 1D sequences were used. Aliasing could become a serious problem with such large holes in the distribution. Note in the example shown, the left half of the curve is identical to the right half with a big gap in the middle. Such gaps could cause visual and numerical problems. With a jittered 1D sequence, the curve did not exhibit these problems.

Hilbert II + Peano: Both the Hilbert II and Peano curves produced high quality results visually and in terms of measured discrepancy. Note that the

Hilbert II curve does not have an axis of symmetry, unlike the Hilbert curve, and as a result does not suffer from the same aliasing problem.

The Hilbert II curve gave promising visual results as well as numerical discrepancy results and was hence chosen for further evaluation. Although the Hilbert curve performed poorly in the visual evaluation for certain 1D sequences, its numerical discrepancy results were also good and scale well according to [1]. Furthermore, existing algorithms exist to generate it quickly, and previous results suggest that the Hilbert curve has the best *locality* properties (see Sect. 2). Hence, it was also chosen for further evaluation.

Although the Peano curve gave promising preliminary results, we discarded it for two reasons: firstly, its natural orientation is not aligned with the sides of the reference square, making its construction more complex to implement. Secondly, its constructor is self-intersecting: multiple vertices lie on the same point. Points in the square therefore may be covered more than once by the curve, increasing the likelihood of clumping. The converse is true for the other space-filling curves, as shown by the following argument. As the curve completely traverses the unit square and comes within a definite maximum distance of every point of it, if points are placed evenly along the curve, clumping cannot occur provided that any one segment of the curve has a minimum distance from other segments of the curve (apart, of course, from the ones preceding and following it). Hence, if no clumping can exist on the line, for a particular $1D$ sequence, the quality of the distribution would appear to be entirely due to the structure of the curve. This explains how space-filling curves, although always completely filling a space, can, and do, distribute the same sequence of points differently.

From now on, we will refer to the Hilbert curve sampled using a jittered 1D sequence as Hilbert-Jittered.

4.2 Evaluation of Numerical Discrepancy

Following the preliminary tests in the previous section, we investigated in detail the discrepancy properties of 2D point distributions generated by our implementation using the Hilbert and Hilbert II curves, sampled using random, evenly-spaced, jittered and low-discrepancy 1D point sequences. We found that the Jittered-Hilbert approach demonstrated the most consistent results, and so we only compare these with 2D point distributions produced at random, and using Niederreiter's method, Sobol's method and jittering. Note that when testing on the plane, even with adaptive curve generation, the curve would in fact be uniform, as the area is constant.

All of the tests were performed for point sets of size $N = 2^l$ and $N = 2^l + 2^{l-1}$, for $l = 0, \ldots, m$. Setting m to 19 gives 40 distribution sizes varying from 2 points to 1572864 points, resulting in a logarithmic range of data. Note that the number of vertices on the curve v_l generated from the recursive approximation depth k and the sizes of N were non-commensurate. However, testing with a uniform curve (a curve with a constant recursive depth) showed that even with the largest N and k set to 12 for the Hilbert curve and 8 for the Hilbert II curve, the curves

1D Sequence	*Rectangular*	*Circular*	*Triangular*
2D Random	-0.49	-0.5	-0.49
Sobol	-0.90	-0.70	-0.58
Niederreiter	-0.90	-0.69	-0.57
2D Jittered	-0.75	-0.74	-0.72
Hilbert Jittered	-0.73	-0.73	-0.72

Table 1. Gradients of least-squares best-fit discrepancy lines for distributions on the plane

consist of enough vertices such that the distance along the curve between each point placed is large enough to achieve the upper bound of the discrepancy of the set for each geometric subset.

Results are shown using graphs displaying discrepancy versus the logarithm of the number of points. Although theoretical discrepancy results are characterised by a power law times a logarithmic factor, the logarithmic factor is hard to determine experimentally due to its minor numerical effect. As can be seen in our graphs, on a double-logarithmic scale the experimentally determined discrepancy can be well approximated by a straight line. Thus, computing the slope of this line gives us an adequate way of comparing the behaviour of the different approaches.

For a random sequence the expected slope of a least-squares fitting line is $-1/2$. Clearly, we hope our point distributions to scale better than a random distribution. For N points in the unit square, the expected best relative error in area which can be achieved is $O(N^{-1}\log^2 N)$ [16], giving a lower bound of approximately -1 for the slope of the best-fit straight line, which we hope the method should approach as closely as possible.

Figure 5 shows scaling of rectangular discrepancy for the 2D random, Niederreiter, Sobol, 2D jittered and Hilbert-jittered distributions. It is clear that the slopes for the Niederreiter and Sobol distributions are the steepest, outperforming the other distributions. The Hilbert-Jittered and two-dimensional jittered distributions performed very similarly; not as well as the Niederreiter and Sobol sequences, but closer to them than to the random sequences. Figure 6 makes a similar comparison using a circular discrepancy measure. The Niederreiter, Sobol, Hilbert-jittered and 2D Jittered sequences perform similarly. Figure 7 makes a further comparison using the triangular discrepancy measure. In this case, the 2D jittered and Hilbert-jittered sequences outperform the Niederreiter and Sobol sequences, which perform closer to the random sequence. The random sequence is worst, as expected, throughout all three tests. The slopes of the best-fit straight lines in each case are listed in Table 1.

4.3 Discussion

When considering rectangular discrepancy, the Niederreiter and Sobol sequences perform better than the Hilbert space-filling curve. We also note that the distribution generated performs similar to the 2D jittering technique as might be

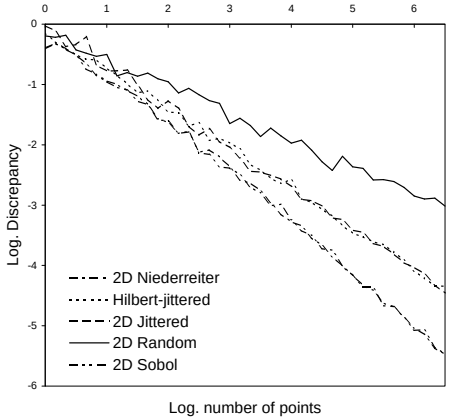

Fig. 5. Rectangular discrepancy compari-

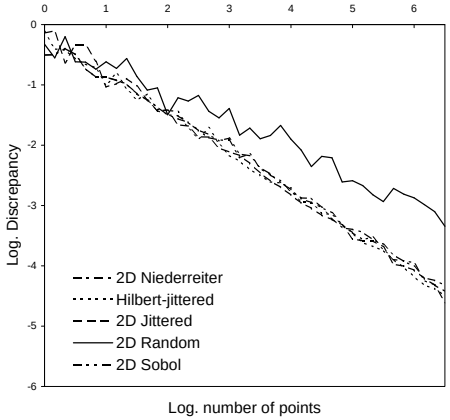

Fig. 6. Circular discrepancy comparison

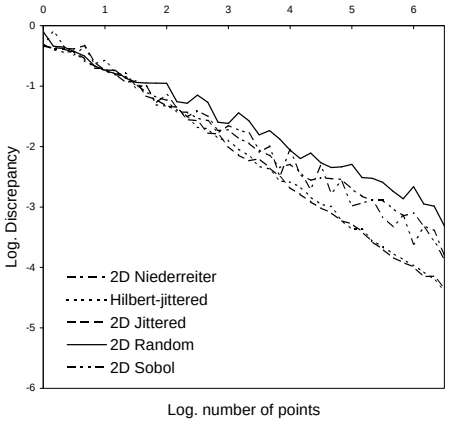

Fig. 7. Triangular discrepancy comparison

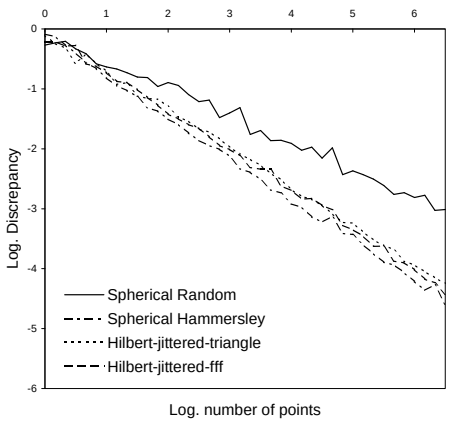

Fig. 8. Spherical discrepancy comparison

expected, probably due to the grid-like structure of the curves. When considering circular and triangular discrepancy, however, the picture is quite different. The Niederreiter and Sobol sequences perform considerably worse in these tests, especially the triangle test where they perform only slightly better than the random sequence, suggesting little robustness. The two best performing distributions for these two tests were the jittered 2D and Hilbert-jittered curve distributions. It appears that the Sobol and Niederreiter methods are *too* specialised to rectangular discrepancy, and the jittered methods are better for all-round usage. The curve method is also consistently superior to using a random distribution.

In [1] the star discrepancy is briefly investigated. Expanding on this, our experiments strongly suggest that the space-filling curve approach gives distributions in the plane with a low-discrepancy behaviour. We also demonstrate the

robustness of the technique when testing the star-discrepancy with different geometric subsets. While we do not have a rigourous theoretical proof for this, [1] suggests that the approach is very similar to regular stratified sampling, only with irregularly shaped strata and, hence, has the same discrepancy.

5 Experiments with Surfaces in 3D

In this section, we test point generation on 3D parametric surfaces. We first compare the point distributions generated on the unit sphere with certain reference point distributions listed in Sect. 2. To do so we employ the spherical discrepancy measure described using spherical triangles. We compare results using the Hilbert-jittered space-filling curve approach with the spherical-Hammersley and spherical-random 2D point distributions. We end this section by demonstrating sampling of various surfaces generated by our algorithm using uniform and non-uniform density. Finally we consider the utility of adaptive space-filling curve generation, and its limitations, and show how reparametrisation may also be required.

5.1 Numerical Experiments

To assess the quality of techniques for generating point distributions on surfaces or volumes in 3D, their properties with respect to measuring areas or volumes are often employed. Computing a measure on surfaces equivalent to the star discrepancy is non-trivial and alternative approaches such as techniques described in [33] may be more suitable to compare the quality of the distributions. Hence, in the following we only compare numerical results for point distributions on the unit sphere.

We generated points on the sphere using the spherical Hammersley distribution, a spherical random distribution and our method based on Steigleder and McCool's algorithm. We assessed how the spherical discrepancy varied with point set size, for point sets of size $N = 2^l$ and $N = 2^l + 2^{l-1}$ for $l = 0, \ldots, 19$. Sect. 3 gives three techniques for approximating the local surface area at a point on the Hilbert curve, based on triangles, rectangles and infinitesimal area elements using the first fundamental form of the surface: $dA = \sqrt{EG - F^2}\, du\, dv$. We show results of variation in discrepancy with number of points, as before, for the triangular and first fundamental form methods, denoted by Hilbert-jittered-triangle and Hilbert-jittered-fff, to demonstrate the difference in quality of distribution between the least and most exact area approximation methods.

When the adaptive curve generation algorithm is used on the sphere, there is little variation in curve recursion depth across the surface of the sphere, resulting in a nearly uniform curve recursion level. However, for the purpose of these tests, we disabled the adaptive sampling, leaving us with a completely uniform curve recursion so that results are consistent.

Figure 8 shows the variation of discrepancy with point set size on a logarithmic scale. The spherical Hammersley sequence performs the best by a small margin, very closely followed by the Hilbert-jittered-fff and Hilbert-jittered-triangle

Sequence	*Spherical*
Random	-0.49
Hammersley	-0.75
Hilbert-jittered-fff	-0.73
Hilbert-jittered-triangle	-0.71

Table 2. Gradient of least squares fit line of distributions on the sphere

methods, while the random distribution performs considerably worse than the other methods.The gradients of the best fit lines of the various distributions are given in Table 2, where we can see that the Hammersley sequence performs in a similar way to the Niederreiter and Sobol sequences in the plane when measuring discrepancy using quarter-circles. The gradient for the Hilbert-jittered-fff approach is similar to that of the spherical Hammersley sequence, closely followed by the Hilbert-jittered-triangle approach. The spherical random sequence, however, performs poorly.

From the results, we can see that the best fit lines for the spherical Hammersley point set have very slightly better slopes than our approach. The random distribution on the sphere, however, performed considerably worse than the other three approaches; similar to the 2D random distribution. Hence, using the Hilbert-jittered distribution and our surface equalisation technique, we can produce a low-discrepancy distribution on the unit sphere comparable to other low-discrepancy sequences that have been specifically designed to work with a sphere. Our algorithm, however, works for any parameterised surface and attempts to correct for severe stretching of the parameter domain. In addition it allows the user to adjust the point distributions with a density-function and maintains a consistent localised ordering of points.

5.2 Visual Demonstrations

We provide visual results to show the method being applied to sample various surfaces, both to produce a uniform density, and a controlled density of points. We show points distributed on the unit square, on a Monkey Saddle, an Eight Surface and a Whitney Umbrella.

Figure 9 shows three images of $3,000$ points distributed in the unit square. The image on the left shows a uniform unit density δ. The middle image shows $\delta(u, v) = u$ and the image on the right shows $\delta(u, v) = u + v$, where u and v are coordinates in the unit square.

Figures 10, 11 and 12 show three images of the Monkey Saddle, the Eight Surface and half of the Whitney Umbrella respectively using the Hilbert-jittered-fff approach. Each figure shows a parametric mesh visualisation of the surface, two images demonstrating point distributions generated using our approach, and the adaptive Hilbert curve mapped onto the surface. Each middle image shows points with uniform density whilst the right-hand images show a distribution density proportional to the Gaussian curvature of the respective surface. The

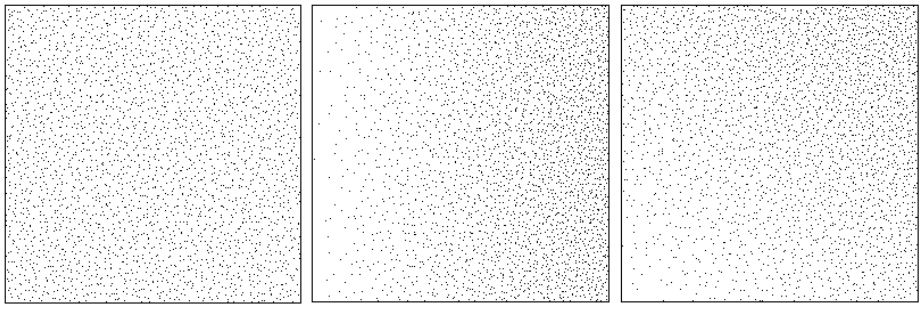

Fig. 9. Distribution in the unit square: uniform density; density $= u$; density $= u + v$

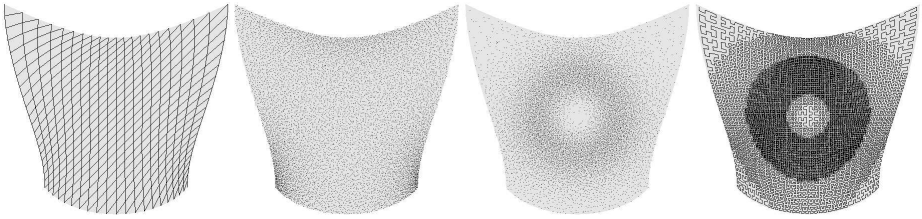

Fig. 10. The Monkey Saddle: parametric mesh; uniform density; curvature controlled density; adaptive Hilbert curve according to curvature

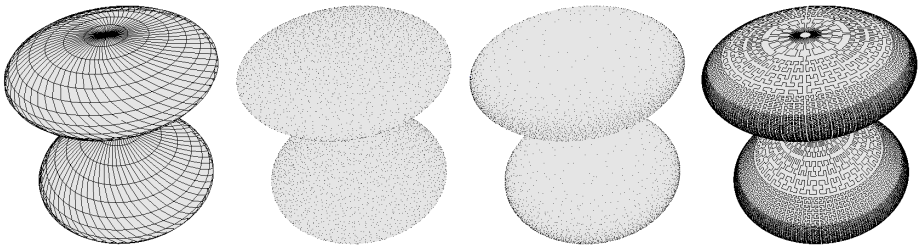

Fig. 11. The Eight Surface: parametric mesh; uniform density; curvature controlled density; adaptive Hilbert curve according to curvature

Monkey Saddle is sampled with $3,000$ points, the Eight surface with $10,000$ points and the Whitney Umbrella with $6,000$ points.

Note that even on the images with uniform density, some areas appear darker than others. This is a visualisation problem, rather than a fault in the distributions, depending on the angle of viewing of the surface. Due to the severity of this problem on the Whitney Umbrella near the central singularity, only half of the surface has been drawn. Also note that the images of the adaptive curves were simplified in terms of recursive depth to make it easier to see the curve.

Our Java implementation on a 3Ghz Intel Pentium 4 with 1GB RAM takes about a second to generate the images shown above, with the curve generated

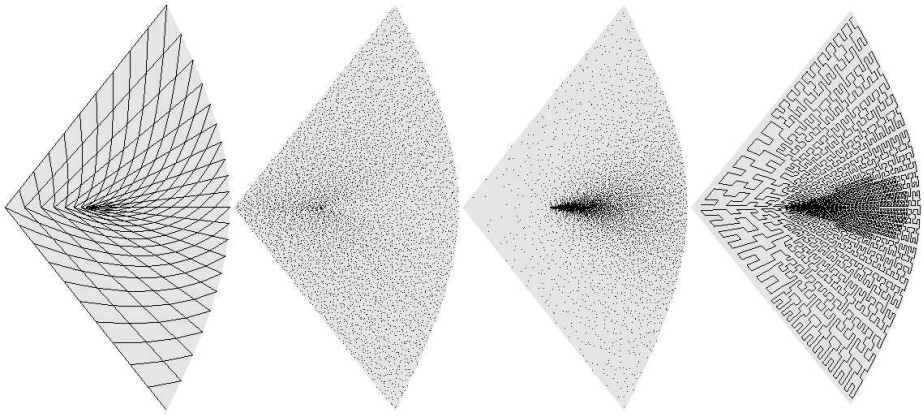

Fig. 12. Half of the Whitney Umbrella: parametric mesh; uniform density; curvature controlled density; adaptive Hilbert curve according to curvature

adaptively according to surface curvature. For these examples, the recursive curve depth used was: $8 < k < 15$. In this context, where the surface can be parameterised straightforwardly, using the adaptive curve generation technique simply allows us to have far fewer curve vertices in order to sample at the required density. The extra computation involved in deciding whether to subdivide the curve at every branch in the tree, however, increases the algorithm complexity, and thus runtime. However, if we have a non-uniform density (as in the examples shown in this section), runtime actually scales much better using the adaptive method as far less curve has to be generated in areas with a low density. The adaptive method also requires considerably less memory in this type of situation.

Showing that the approach given produces the same quality of distribution on any parametrically described surface is hard, but the visual results obtained above are plausible. Unlike earlier surface point distribution algorithms, this method can sample arbitrary surfaces, whilst providing direct sampling density control.

One limitation of our results is the reliance on the approximation of discrepancy as the only real quality measure of a distribution. This is accentuated by the number of test cases used: the plane and the sphere. However, as briefly discussed in Sect. 1, measuring the discrepancy on general surfaces robustly is a hard problem. We also demonstrate our results visually, but one can only garner so much from this. Essentially, it would be desirable to have another quantifiable metric by which we could compare results for various distributions tailored specifically for rendering and engineering applications. There are various options for application specific measures, and while none are perhaps quite as generically applicable as discrepancy, for the specific applications we intend to explore in the future, they might be of considerable use. Another possibility is Mitchell's [34] *Blue Noise* criterium: the sampling spectrum should be noisy and

have few spikes, with a deficiency of low-frequency energy. The fulfilment of this criterium might be measured for the distributions.

5.3 Limitations of Adaptive Curve Generation

In this section, we discuss the limitations of adaptive curve generation technique. Figure 13 shows the Hilbert space-filling curve mapped onto the surface of a superellipsoid $x(u,v) = (cos^{1/3}u\ cos^{1/3}v)$, $y(u,v) = (cos^{1/3}u\ sin^{1/3}v)$, $z(u,v) = sin^{1/3}u)$ with and without adaptive curve generation. The non-adaptive curve consists of approximately $262,000$ vertices, whilst the adaptive curve consists of only $91,000$. It is clear that although there are still gaps between curve vertices when using the adaptive technique, they are far smaller than when using a uniformly generated curve.

This example also shows that there is a maximum density that can be reached for a given parametrisation due to machine precision limits. When generating the Hilbert curve we map 1D points to 2D points via bit indices using 2 algorithm. Thus, a 1D point represented by b bits, is mapped onto a 2D point where for each of its coordinates we only have $b/2$ bits. Thus, two distinct, computed points in the parameter domain differ in at last one coordinate by at least $e = 1/(2^{b/2}-1)$. Hence, to reach a minimum density of 1 point per surface area a, squares of the size e^2 in the parameter domain should be mapped to areas on the surface of at most size a. Or if we estimate the area in our algorithm via the first fundamental form, the norm of the first fundamental form should be smaller than a/e^2. Note that this ignores any numerical problems in evaluating the parametrisation.

To handle parameterisations which do not allow us to reach a certain density, an alternative to using multi-precision arithmetic is to reparameterise the surface locally or globally using polynomials. E.g. for the superquadric above, we can reparameterise it piecewise with a simple power law. In the interval $[0, \pi/2]$ we use $u = u'^3\pi/2$ to replace $\sin^{1/3}(u)$ with $\sin^{1/3}(u'^3\pi/2)$ for $u' \in [0,1]$ and similar for the other intervals and for the cosine.

Figure 14 shows one corner of the superquadric shown in Fig. 13, with three curves mapped onto the surface. The image on the left shows a uniform Hilbert curve of depth $k = 7$. The middle image shows an adaptive Hilbert curve with $6 < k < 14$, and the image on the right shows a cubed reparameterisation of a uniform Hilbert curve of depth $k = 7$.

Generating the space-filling curve adaptively allows us to achieve localised higher densities on surfaces for a similar or lower memory requirement. The approach also allows us to correct for extreme parameterisations. However, Fig. 14 shows the limitations of this for an extreme parameterisation. To improve this situation for specific parameterisations, an alternative to using multi-precision arithmetic is to reparameterise the surface, either locally or globally. We have demonstrated that in the case of the superquadric, we can reparameterise it using a global power law (a simple polynomial fitted to part of the parameter to alter the distribution to better adapt to the sine or cosine terms). Future work will involve automating this reparameterisation approach, possibly on a per-cell basis during subdivision to take advantage of adaptive technique.

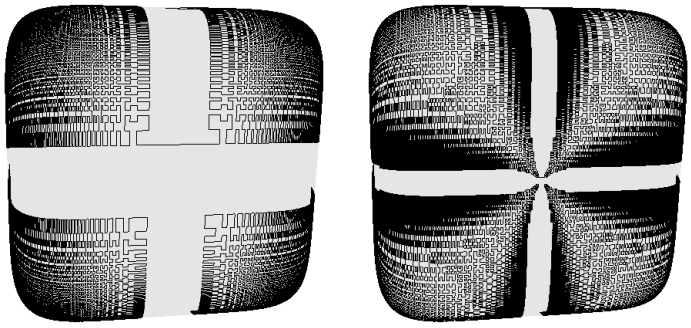

Fig. 13. A superellipsoid: without (262,000 vertices) and with (91,000 vertices) adaptive sampling

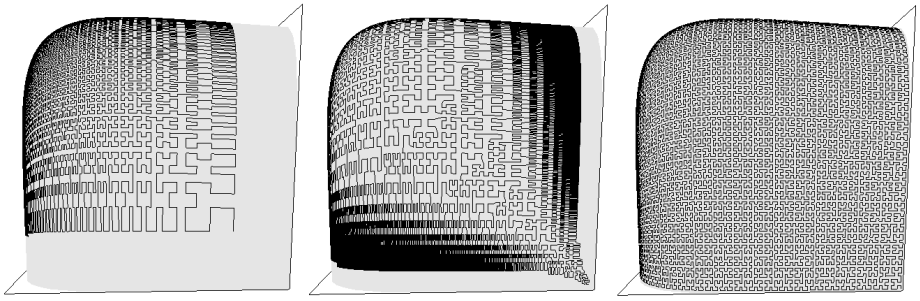

Fig. 14. One corner of a superellipsoid: normal distribution, uniform curve (left); normal distribution, adaptive curve (middle); cubed distribution, uniform curve (right)

6 Conclusions

We have discussed the generation of density-controlled low-discrepancy distributions on arbitrary parametric surfaces employing a method due to Steigleder and McCool [1]. It allows us to map a 1D point sequence onto a surface via an adaptively generated space-filling curve. We have examined various choices for the 1D point distribution and space-filling curve. We found that both evenly-spaced and low-discrepancy 1D point sequences perform well when used with Hilbert and Hilbert II curves, where the Hilbert II curve is superior to the Hilbert curve. However, using a jittered evenly-spaced 1D point sequence in combination with adaptive generation of the Hilbert curve produces good point distributions in a short time for general surfaces and density functions.

Comparing the approach with other known low-discrepancy sequences in 2D and for special surfaces in 3D showed that specialised methods may perform better for some particular surface or discrepancy measure. However, the presented method is only slightly worse the specific distributions and outperforms these distributions if considered for general use. Overall this approach robustly generates low-discrepancy point distributions on surfaces, although reparame-

terisation may be needed to overcome limitations of the method arising from machine precision limits. We note that the use of the space-filling curve provides many inherent advantages, such as locality and connectivity.

One area of future work involves applying our method to a particular application and comparing how well it solves the problem to other approaches. Such problems include the computation of surface integrals of geometric models using the Crofton formula [25], and other multivariate integration problems such as a Monte Carlo approach to computational fluid dynamics [35] and finite element analysis [8, 9]. We may also test the approach with a ray-tracing or radiosity simulator to visually asses the distributions. These results would allow us to gauge the quality of various distributions in real-world problems.

One limitation of the method given is its reliance on a fully generated and stored adaptive Hilbert curve. We believe the method could be accelerated greatly if it were implemented to run directly on a GPU. [36] describes an algorithm to generate L-system subdivision curves using 32-bit precision pixel shaders. This technique maps well to programmable GPUs, and if further parallelised, could considerably increase the speed of space-filling curve generation.

References

1. Steigleder, M., McCool, M.: Generalized stratified sampling using the Hilbert curve. Journal of Graphics Tools: JGT **8**(3) (2003) 41–47
2. Bern, M., Eppstein, D.: Mesh Generation and Optimal Triangulation. In Hwang, F.K., Du, D.Z., eds.: Computing in Euclidean Geometry. World Scientific (1992)
3. Keller, A.: Instant radiosity. In: SIGGRAPH. (1997) 49–56
4. Cook, R.L.: Stochastic sampling in computer graphics. ACM Trans. Graphics **5**(1) (1986) 51–72
5. Rusinkiewicz, S., Levoy, M.: QSplat: A multiresolution point rendering system for large meshes. In Akeley, K., ed.: Proc. ACM SIGGRAPH Comput. Graph. (2000) 343–352
6. Kobbelt, L., Botsch, M.: A survey of pointbased techniques in computer graphics. Computers and Graphics **28**(6) (2004) 801–814
7. Zwicker, M., Pauly, M., Knoll, O., Gross, M.: Pointshop 3D: An interactive system for point-based surface editing. In Hughes, J., ed.: Proc. ACM SIGGRAPH. (2002) 322–329
8. Zagajac, J.: A fast method for estimating discrete field values in early engineering design. In: Proc. 3rd ACM Symp. Solid Modeling and Applications. (1995) 420–430
9. Shapiro, V.A., Tsukanov, I.G.: Meshfree simulation of deforming domains. Computer-Aided Design **31**(7) (1999) 459–471
10. Floater, M.S., Reimers, M.: Meshless parameterization and surface reconstruction. Computer Aided Geometric Design **18**(2) (2001) 77–92
11. Dobkin, D.P., Eppstein, D.: Computing the discrepancy. In Proc. 9th ACM Symp. Computational Geometry (1993) 47–52
12. Zeremba, S.: The mathematical basis of monte carlo and quasi-monte carlo methods. SIAM Review **10**(3) (1968) 303–314
13. Gotsman, C., Lindenbaum, M.: On the metric properties of discrete space-filling curves. IEEE Trans. Image Processing **5**(5) (1996) 794–797

14. Niederreiter, H.: Random number generation and quasi-monte carlo methods. SIAM Review (1992)
15. Niederreiter, H.: Quasi-monte carlo methods and pseudo-random numbers. Bull. AMS **84** (1978) 957–1041
16. Davies, T.J.G., Martin, R.R., Bowyer, A.: Computing volume properties using low-discrepancy sequences. In: Geometric Modelling. (1999) 55–72
17. Keller, A.: The fast calculation of form factors using low discrepancy sequences. In Purgathofer, W., ed.: 12th Spring Conference on Computer Graphics, Comenius University, Bratislava, Slovakia (1996) 195–204
18. Bratley, P., Fox, B.L., Niederreiter, H.: Algorithm 738: Programs to generate Niederreiter's low-discrepancy sequences. j-TOMS **20**(4) (1994) 494–495
19. Bratley, P., Fox, B.L.: Algorithm 659: Implementing Sobol's quasirandom sequence generator. j-TOMS **14**(1) (1988) 88–100
20. Shirley, P.: Discrepancy as a quality measure for sample distributions. In: Eurographics '91. Elsevier Science Publishers, Amsterdam (1991) 183–94
21. Halton, J.H.: A retrospective and prospective survey of the monte carlo method. SIAM Review **12** (1970) 1–63
22. Kocis, L., Whiten, W.J.: Computational investigations of low-discrepancy sequences. ACM Trans. Math. Softw **23**(2) (1997) 266–294
23. Wong, T.T., Luk, W.S., Heng, P.A.: Sampling with hammersley and halton points. J. Graph. Tools **2**(2) (1997) 9–24
24. Secord, A., Heidrich, W., Streit, L.: Fast primitive distribution for illustration. In: Rendering Techniques. (2002) 215–226
25. Li, X., Wang, W., Martin, R.R., Bowyer, A.: Using low-discrepancy sequences and the crofton formula to compute surface areas of geometric models. Computer-Aided Design **35**(9) (2003) 771–782
26. Rovira, J., Wonka, P., Castro, F., Sbert, M.: Point sampling with uniformly distributed lines. Eurographics Symp. Point-Based Graphics (2005) 109–118
27. Hartinger, J., Kainhofer, R.: Non-uniform low-discrepancy sequence generation and integration of singular integrands. In H. Niederreiter, D.T., ed.: Proc.MC2QMC2004, Springer-Verlag, Berlin (2005)
28. Hlawka, E., Mück, R. In: A Transformation of Equidistributed Sequences. Academic Press, New York (1972) 371–388
29. Elber, G.: Free Form Surface Analysis using a Hybrid of Symbolic and Numeric Computation. PhD thesis, Dept. of Computer Science, University of Utah (1992)
30. Mandelbrot, B.: The fractal geometry of nature. Freeman, San Francisco (1982)
31. Butz, A.: Alternative algorithm for hilbert's space-filling curve. IEEE Trans. Computers **Short Notes** (1971) 424–426
32. Lawder, J.: Calculation of mappings between one and n-dimensional values using the hilbert space-filling curve. Technical Report JL1/00 (2000)
33. Cui, J., Freeden, W.: Equidistribution on the sphere. SIAM J. Sci. Comput. **18**(2) (1997) 595–609
34. Mitchell, D.R.: Generating antialiased images at low sampling densities. In Stone, M.C., ed.: SIGGRAPH '87 Conference Proceedings (Anaheim, CA, July 27–31, 1987), Computer Graphics, Volume 21, Number 4 (1987) 65–72
35. Alexander, F.J., Garcia, A.L.: The direct simulation monte carlo method. Comput. Phys. **11**(6) (1997) 588–593
36. Mech, R., Prusinkiewicz, P.: Generating subdivision curves with L-systems on a GPU. In: SIGGRAPH. (2003)