Eurographics Symposium on Point-Based Graphics (2007)
M. Botsch, R. Pajarola (Editors)

Low-Discrepancy Point Sampling of Meshes for Rendering

J. A. Quinn, F. C. Langbein and R. R. Martin

School of Computer Science, Cardiff University, UK

Abstract

A novel point sampling framework for polygonal meshes is presented, based on sampling a mesh according to a density-controlled low-discrepancy distribution. The local sampling density can be controlled by a density functional defined by the user, e.g. to preserve local features, or to achieve desired data reduction rates. To sample the mesh, it is cut into a disc topology, and a parametrisation is generated. The parameterised mesh is sampled using a Hilbert curve in the parameter domain, which is adapted to parametric distortions and mapped onto the mesh. 1D sample points along the Hilbert curve are then generated, correcting for parametric distortion and a user-specified local density, to give a density-controlled low-discrepancy sampling of the mesh. After a pre-processing step, the sampling density can be adjusted in real-time. Experiments show that this approach can quickly resample existing meshes with low discrepancy samples. The effectiveness and speed of the approach are demonstrated by applying it to viewpoint dependent rendering, level of detail representation, and interactive remeshing.

1. Introduction

Point based representations of surfaces are becoming a viable alternative to polygon meshes, especially for complex, smooth or dynamic surfaces [KB04]. However, meshes are the most common method for rendering and storing models. Triangle meshes may be irregular due to limitations of the capture process, and may e.g. describe large flat areas using a small number of polygons. Vertices of such meshes are not suitable for point based rendering. There is a need to resample such meshes to produce point sets more useful for point based graphics, e.g. with more regular distributions that preserve shape well. A secondary objective may be to adjust the sampling density to meet user criteria, e.g. the local point sampling density is proportional to surface curvature.

This paper thus does not focus on rendering of the points, but on the sample distribution. Nevertheless, the sampling must take into account all requirements of the rendering stage in order to be useful. The criteria that we suggest are as follows: the points should be evenly distributed, usually with respect to some function of curvature. The points must not be, however, arranged in a regular pattern in order to avoid artifacts and aliasing problems [Zwi03]. The point sampling must be invariant to direction, that is, given a uniform, isotropic density, the distance between points should be consistent in all directions.

We propose a technique, utilising previous work on sampling parametric surfaces with space-filling curves [QLME06], to resample arbitrary polygonal meshes. The emphasis is on the quality of the new point distribution, with the ability to control and adjust the sampling density at interactive rates. We generate a point distribution on the surface that has low discrepancy with respect to a user-specified density function. Previous work has shown that low-discrepancy sampling of a manifold is a very efficient method of capturing geometry with a minimum number of samples [RWCS05]. Low-discrepancy distributions also generally ensure an even distribution without regular grid-like patterns, and thus such sampling produces fewer artifacts caused by aliasing [Coo86]. This can be important in such applications as computing volume properties [DMB99], and rendering [Kel97]. In Sect. 4.2, we also demonstrate that our jittered sampling produces a better surface coverage than other techniques for common point-based rendering approaches.

We now briefly overview our technique (for details see Sect. 3). Taking a triangle mesh as input, we first cut it (if necessary) to give a disc topology, and parameterise it in $[0,1]^2$. We then adaptively generate a finite approximation to a Hilbert curve in this parameter domain, such that when its vertices are mapped back to the surface mesh, the mesh is covered uniformly according to a user-specified density (see Sect. 3.2). The space-filling curve allows us to treat the 2D problem of evenly distributing points on the surface as a 1D problem. We start by generating a 1D distribution to sample the Hilbert curve; for reasons discussed in [QLME06], we use jittered samples. To achieve the desired density of points on the surface, we approximate the local area distortion caused by the parametrisation, and take into account the user specified density function. We use a technique similar to histogram equalisation from image processing to produce the desired point distribution on the mesh.

There is no universally applicable method to assess the quality of a resampled surface. Various assessment criteria are employed in the literature, e.g. discrepancy [RWCS05], minimum angle statistics [SG03], the blue-noise criterion [AMD02] and the Hausdorff distance: for measuring the accuracy of remeshed surfaces [CRS98] and point sampled surfaces [WZK05]. Even visual assessment can be useful, especially if the final application is rendering. For (quasi) Monte-Carlo integration, it has been demonstrated that estimates of the area or volume of an object using low-discrepancy samples converges far faster than when using random sampling [LWMB03]. An experimental technique used by [RWCS05] involves sampling a unit square polygon divided into two triangles, and calculating the star discrepancy for a varying number of points. The technique demonstrates the discrepancy on the plane, but does not address the discrepancy of the surface as a whole.

In Sect. 4, we employ an approximate discrepancy measure on the entire mesh to demonstrate that the discrepancy in point sets produced by our approach drops as we increase the number of points in a similar fashion to low-discrepancy sequences previously tested on parametric surfaces [QLME06, CF97]. This verifies that the mesh sample distributions exhibit low discrepancy properties, i.e. they cover the surface area well and do not suffer from aliasing associated with regular sampling. We also assess the coverage of points in the unit square, comparing low-discrepancy and random distributions to our technique.

Although the pre-processing step of generating the Hilbert curve in $[0,1]^2$ is somewhat time-consuming (a few seconds), generating a new set of point samples on the mesh is extremely fast (less than 10 milliseconds for all meshes tested), allowing us to resample the surface at interactive rates in response to user required changes of the density, e.g. as the camera zooms in or out. If the density changes significantly, we must subdivide the curve locally (see Sect. 3.2) in a background process.

Experimental results are given to demonstrate the distributions produced using our technique, and to compare them to other approaches. To demonstrate the flexibility of our technique and advantages of using the Hilbert curve as the underlying technique for resampling, we show that it can be easily applied to viewpoint and level of detail based rendering 4.3, and remeshing 4.4. For the former, we give timing and visual results, and demonstrate silhouette enhancement. We compare the quality of meshes produced by our remeshing implementation to other competitive techniques by measuring the Hausdorff distance using Metro [CRS98], showing results competitive with one of the most popular remeshing techniques.

In the following, we first discuss prior work (Sect. 2). We then present our algorithms (Sect. 3), followed by experimental results (Sect. 4) and conclusions (Sect. 5).

2. Related Work

In this section, we review work related to point sampling meshes and low-discrepancy distributions.

An overview of point based graphics is given in [AGP*04]. This includes work on levels-of-detail (LOD) and viewpoint dependent rendering (VDR), neighbourhood computation, especially K-nearest neighbours and triangulation, surface simplification, especially iterative methods and particle simulation, and error metrics such as the Hausdorff distance. Kobbelt and Botsch [KB04] also give a broad review of point based techniques, with a concise overview of different representations and rendering techniques. Of particular interest is the discussion about neighbourhood calculations, point sampling and level of detail (LOD). Zwicker [Zwi03] also surveys the field thoroughly; he briefly looks at methods to avoid aliasing problems in sampling, such as the blue noise criterion, also discussed with regard to remeshing in [AMD02], and non-spectral properties, namely discrepancy.

Rovira et al. [RWCS05] describe how to sample an input mesh by intersecting uniformly distributed lines with the model, projected from bounding box tangent planes. They numerically assess their technique by sampling a unit square polygon divided into two triangles, and measure the star discrepancy of the distribution. They show results that appear to be two orders of magnitude worse than sampling points directly on the polygon with known low-discrepancy distributions (such as Hammersley and Sobol). They also do not consider the discrepancy of the sampling over the whole mesh, and cannot control sampling density. Our approach, however, provides a low-discrepancy distribution over the whole mesh, whilst also having full control over the sampling density. The distributions generated by our algorithm are superior to common low-discrepancy distributions for point based graphics applications, as we show later.

Wu et al [WK04] suggest a sampling technique, and generalise this to LOD splatting [WZK05], employing a progressive error-based decimation technique for point based models, taking into account splat geometry. They demonstrate good results, with low errors, taking about 30 seconds to generate a single model for a large mesh. They cannot, however, control the density of the distribution, and only consider the quality of the distribution in terms of the surface approximation error. Also, whilst the time required to generate a single level of detail is similar to our technique, after this processing time, our approach allows interactive-rate control of the level of detail with arbitrary granularity, according to a specified density, with a guaranteed sampling quality.

We now discuss some of the methods that we utilise in our work. The input mesh, if not already having disc topology, must be cut; we use the algorithm in [SAL06]. Our methods are applicable to arbitrary polygonal meshes, but for simplicity we only consider triangle meshes here. Thus, we use Floater's [Flo97] parameterisation technique, as we desire triangle shapes to be preserved to avoid anisotropic stretch. Because we adaptively sample the parameterised mesh, large area ratios between triangles on the surface and in the parameter domain do not affect the quality of our results (although they may have a minor impact on speed). Floater's technique is also fast. We generate an adaptive Hilbert curve in $[0,1]^2$, based on the algorithm described in [QLME06], according to the input mesh. In [SM03], it is demonstrated that a jittered sampling of the Hilbert curve can produce a generalised stratified sampling. In [QLME06], it is demonstrated that this technique can be further generalised to parametric surfaces, using an approach similar to histogram equalisation to evenly distribute the samples on the surface. When sampling the curve, we recommend a 1D jittered, or stochastic sampling of the Hilbert curve to produce a low-discrepancy distribution. However, other, completely deterministic sampling techniques may also be used such as evenly spaced points or the Niederreiter sequence. Such sequences can be used to guarantee that points on the coarser level models are a subset of the points of finer levels. We also utilise ideas from [KV03]; each sample is stored as a 'differential point', providing discrete differential surface properties such as principal curvatures and surface normals, allowing the most efficient use of sample points.

Due to the absence of topology in point based surface representations, the literature often notes the importance of neighbourhood calculations, e.g. for normal computation [DLS05]. Work such as [SSV07], focusing on huge point sets, further shows the importance of efficient neighbourhood calculations. Our approach uses the Hilbert curve as the underlying construct for the point distributions which also implies a topology: a Hilbert curve is related to a particular quad-tree decomposition of $[0,1]^2$. This inherently provides fast neighbourhood operations [KF94], allowing us to recompute surface properties very quickly if required.

We now briefly consider some of the work related to the applications of our method. [KB04] and [AGP*04] discuss LOD as an important topic; we also consider VDR [BLM04]. For point based graphics, when performing LOD calculations, a key advantage is that removing a point, unlike for meshes, does not modify any topology. Due to the speed at which we can resample the Hilbert curve, once pre-processing is complete , we can easily resample the surface at interactive rates as the distance between the model and the camera changes: we can achieve an arbitrarily fine-grained LOD technique after one-time initial pre-processing. The sampling of the curve can also be completely deterministic, thus minimising the effect of visual *popping*, discussed in [LWC*02].

With regard to VDR, we do not consider image space culling or occlusion, but an object space analysis of the scene. Thus, we consider what part of the scene is visible to the camera, and then simply sample that part of the scene at the desired density. There is considerable computational overhead involved, and we demonstrate it mainly to show how fast our technique is in practice. A situation where this might be useful could be if the user required a fixed scene complexity regardless of viewing angle. Our approach can also be used to increase the sampling density around silhouettes, providing the illusion of a higher quality model, especially useful for masking the mesh-complexity of a simple model with a high-resolution texture.

For an introduction to requirements of and approaches to remeshing techniques, see Alliez et al. [AGGA05]. An important criterion highlighted is that of fidelity: at a given resolution, a newly generated mesh should approximate the original mesh accurately with as few triangles as possible. Previous work [LWMB03] indicates that a low-discrepancy sampling of a surface may fulfil this requirement very well for a set of points. However, we note that it is not necessarily true that a triangulated low-discrepancy distribution has the same quality. Minimising the number of triangles may not always be the most important requirement—for example, finite element methods are more likely to require a regular mesh of equally sized, almost equilateral triangles. Most remeshing techniques require some form of parameterisation, either global [AMD02] or local [SG03]. However, techniques not requiring parameterisation also exist [ACSD*03, BK01].

Building on the basic technique described in [QLME06], the contributions of this paper are to provide a robust framework tailored for point based graphics applications, allowing the user to generate high quality low-discrepancy distributions from large, arbitrary genus, polygonal meshes very quickly. We generalise our previous work to arbitrary mesh sampling, including GPU accelerated mapping from the parameter domain to the mesh, high quality mesh area computation, mesh discrepancy calculations, and most importantly, demonstration of applications, including a working system for real-time arbitrarily fine-grain localised LOD, viewpoint dependent rendering and a remeshing application to demonstrate the flexibility of our framework.

3. Sampling using Space-Filling Curves

This section describes our approach to resampling a mesh. We first briefly overview the complete algorithm and then provide details on Hilbert curve generation, mapping the Hilbert curve onto the mesh, and the sampling process along the Hilbert curve on the mesh.

3.1. Overview

Our algorithm takes an arbitrary mesh M_s and a user-defined density function δ as input and computes a set of points sampling that mesh. $\delta : M_s \longrightarrow \mathbb{R}_0^+$ is a user-defined, positive function indicating the desired sampling density for the construction of the new distribution; δ might be constant, or proportional to surface curvature, for example. As M_s can be of arbitrary genus it is first cut into disc topology using the algorithm described in [SAL06]. Then a mesh parametrisation algorithm [Flo97] is used to generate a mapping $f : M_p \subset [0,1]^2 \rightarrow \mathbb{R}^3$ from a parameter mesh $M_p \subset [0,1]^2$ to the surface mesh M_s. As mentioned earlier, the choices of parameterisation and cutting algorithms have little effect on the final sampling due to the adaptive nature of the Hilbert curve.

Reducing the dimensionality often simplifies a computational problem. In our case, we do so by sampling the mesh in the parameter domain using a space-filling curve: we generate an adaptive curve in $[0,1]^2$ which covers M_p. We use a recursive Hilbert curve construction [QLME06] locally expanded to different recursion depths, determined by the area scale factor between M_p and M_s given by the parametrisation, and the density function δ. The parametrisation f maps the Hilbert curve onto the surface M_s. We then place sample points $\{p_l : l = 1, \ldots, N\}$ along the curve on M_s based on how the integral $\int \delta ds$ increases along the curve, using an approach similar to histogram equalisation. Note that this integral corresponds to a surface integral, as the Hilbert curve in the limit covers the complete surface. The output distribution is a low-discrepancy set of points sampling the surface [QLME06], meaning that for each subset A of the original mesh M_s, the ratio of the number of sample points lying inside A to the overall number of points N should approximate the ratio of the surface integrals $\int_A \delta ds / \int_{M_s} \delta ds$.

3.2. Generating the Hilbert Curve

For a general introduction to space-filling curves see [QLME06]. A Hilbert curve can be interpreted as mapping $H : [0,1] \longrightarrow [0,1]^2$. We compute an adaptive approximation of this curve using a quad-tree approach, where each vertex of the curve represents a quad (square) in $[0,1]^2$. This can be done rapidly to provide a set of Hilbert curve vertices $\{h_l : l = 1, \ldots, L\} \subset [0,1]^2$ expressed by 2D co-ordinates, which uniquely correspond to 1D co-ordinates via H. If a parameterisation technique similar to that in [GY03] were used, the parameter domain could be sampled uniformly, avoiding the need for an adaptive curve. This would allow use of the ideas in [But68] to generate the Hilbert curve. Otherwise, an adaptive curve is needed, i.e. a curve for which the level of recursive construction is not constant (see Fig. 1). Note that although a change in curve density will affect the exact placement of points, it does not affect the quality of the distribution. The recursion depth of the curve is approximately bounded below by some constant ω times the ratio $\int_A \delta ds / \int_{M_s} \delta ds$, for each subset A of the mesh M_s. In practice, we find that $\omega \geq 10$ is sufficient to provide the required density of space-filling curve vertices on the surface mesh M_s with respect to the density function δ, regardless of parametric stretch or distortion. Simply put, we must locally maintain a minimum density of Hilbert curve vertices, h_l so that we can place our final point distribution with sufficient accuracy.

To calculate how many Hilbert curve vertices h_l are required locally, using the bijective mapping between triangles in M_p and M_s, we cannot simply increase the depth of the Hilbert curve solely based on the area of each triangle in M_s (a small triangle in M_p may map to a very large triangle on M_s). Hence, first we define the number of point samples we need in a given surface triangle as $N_\triangle = \int_\triangle \delta dA / \int_{M_s} \delta dA$. We thus require $v_\triangle$ Hilbert vertices inside a surface triangle with $v_\triangle \geq \omega N_\triangle$. Given an initial construction of our Hilbert curve to some preset recursion depth, we then approximate the required number of vertices that should be contained within a quad at any given level. In practice, we require that $v_{\text{quad}} \geq v_\triangle / (A_p / A_{\text{quad}})$, where v_{quad} is the number of vertices in each quad of the quad tree, A_p is the area of the triangle $f^{-1}(\triangle) \in [0,1]^2$, and A_{quad} is the area of the Hilbert quad. These values must be calculated independently for each quad.

Subsequent to this pre-processing phase of our algorithm, if the density changes greatly, so that there are insufficient pre-computed Hilbert curve vertices in one or more quads, the recursive depth of the curve must be increased to compensate. In this case, we initially approximate the distribution with the maximum accuracy possible with the current Hilbert curve vertices. We then create a new low-priority thread, which subdivides the curve locally as required, and resamples the distribution. The time it takes to subdivide the curve is difficult to meaningfully quantify. For a dramatic change in density, involving nearly all quads, we treat δ as becoming $r\delta$, where r is some constant, and we increase the number of points in each quad by a factor r, resulting in a global increase in recursion depth of the Hilbert curve–this is much faster then recomputing areas for the entire surface. Thus, for small changes we recompute areas locally, and for large changes, globally increase the density as described.

As described in [SM03], as we construct the curve, each 1D co-ordinate is mapped to a 2D coordinate, thus halving the precision available in the unit interval when using limited precision arithmetic. This must be monitored, and in exceptional circumstances, reparameterisation may be required [QLME06].

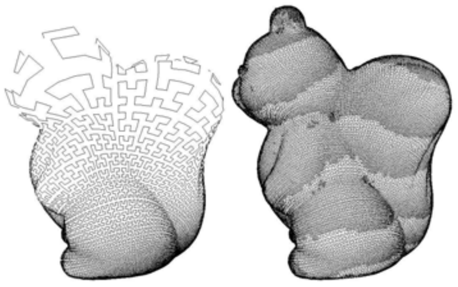
Figure 1: *Uniform (left) and Adaptive (right) Hilbert Curve*

3.3. Mapping the Hilbert Curve onto the Surface

Once the adaptive curve has been generated in the parameter domain, we must map it onto the surface mesh M_s, using the parameterisation f. We convert the points $h_l \in [0,1]^2$ to local triangle barycentric co-ordinates on the parameter mesh M_p, and hence to Cartesian co-ordinates on $M_s \subset \mathbb{R}^3$, resulting in a set of Hilbert vertices on the mesh, $H_l = f(h_l)$. During this process, we also perform a bilinear interpolation of the density function, and the normals for each of the curve vertices. We interpolate these values to allow us to calculate the discrete density s_l for each Hilbert curve vertex H_l as described below.

When performing this mapping, we must first find out which triangle contains each h_l. To do this rapidly, we use the graphics hardware to render the parameter mesh M_p in the frame-buffer at the same resolution as the Hilbert curve, and give each triangle a unique colour. We then read it out as a texture, and for each vertex h_l, we may compute the colour of the triangle that matches its position to find the triangle containing it. Due to the adaptive nature of the curve, we must either render the curve at the maximum depth, or cut the parameter domain into sections based on the depth and perform the operation per section. Furthermore, if required, containment checks can be performed after this process to ensure that points close to a triangle edge were computed accurately. This process is very fast, as we show later.

3.4. Sampling along the Hilbert Curve

We now discuss how to compute the cumulative density S_l, approximating the surface integral along the Hilbert curve on the mesh. This allows us to discretely sample a sequence of points p_l along the Hilbert curve vertices H_l on the surface mesh in a similar way to histogram equalisation. We wish to compute $S_L \approx \int_S \delta ds$, where L is the number of vertices H_l, and thus S_L is the cumulative density along the Hilbert curve. Given the discrete nature of H, we can approximate this integral by $S_l = \sum_{k=1}^{l} s_k, l = 1,\dots,L$. Note that the overall surface integral of δ can be ignored as it is simply a scaling factor and we assume the desired number N of sample points is given. We compute the area of the surface patches A_l multiplied by the density δ, approximating $\int_{A_l} \delta ds$. We calculate the area of each element A_l using the technique described by Meyer et al. [MDSB02]: the area of the Voronoi region of the triangles $\triangle PQR$ for each $Q \in \{H_l\}$ and each pair P, R of consecutive neighbours in Q's one-ring is calculated. Then the total Voronoi area for Q is determined by summing the areas of each of the Voronoi regions, giving us an approximation of A_l. Our cumulative density function, for non-obtuse triangles, is thus

$$S_l = \sum_{l=1}^{l} \varphi_l \sum_{j \in N(i_l)} (\cot \alpha_{i_l j} + \cot \beta_{i_l j}) \|x_{i_l} - x_j\|^2$$

where x_j is a neighbour of vertex x_{i_l} of the one-ring neighbourhood N_1, α and β are the opposite angles of the respective triangles to the line segment between x_{i_l} and x_j, and φ is a discrete local approximation of the density δ. For obtuse triangles, as [MDSB02] recommends, we approximate the Voronoi area by taking the mid-point of the side opposite the obtuse angle in the triangle instead of the Voronoi vertex for that triangle to ensure that no area elements overlap. Note that the basic triangulation is constructed using the eight neighbours of each Hilbert vertex, requiring little additional computation.

We then generate a set of 1D jittered samples $\{q_l : l = 1,\dots,N\}$ in $[0,1]$: we take an evenly spaced set of samples, which are moved a uniform random amount up to half the distance towards the next or previous sample. Once we have calculated the cumulative density S_l over H_l, we move along the Hilbert curve, and whenever S_l becomes larger than the threshold described by the 1D sequence q_l (multiplied by S_L), we sample a point p_l at a vertex along the curve on the surface mesh M_s, producing a distribution of jittered points lying on the surface of the input mesh M_s.

4. Experimental Results

We next consider the run-time of the algorithm and visually demonstrate the output using some example models. We then assess the sample distributions produced by our algorithm by computing their discrepancy and visually assessing the quality of point based rendered images. Following this, we illustrate applications of the framework, viewpoint and LOD based rendering, and remeshing. Finally, we discuss our results. All tests were carried out on a 3GHz Pentium 4, with 1GB of single channel DDR RAM.

4.1. Visual Results and Run-Time Analysis

We first consider the run-time of our algorithm, and demonstrate the results visually for some meshes. Our research does not approach the rendering process of point based models, thus to show them, we use a simple technique described in [Ada06], to produce an output similar to Gourad shading.

Fig. 2 shows the processing stages of our algorithm for the Squirrel model: the original mesh (10K vertices), its parameterisation in $[0,1]^2$, the adaptive Hilbert curve in $[0,1]^2$,

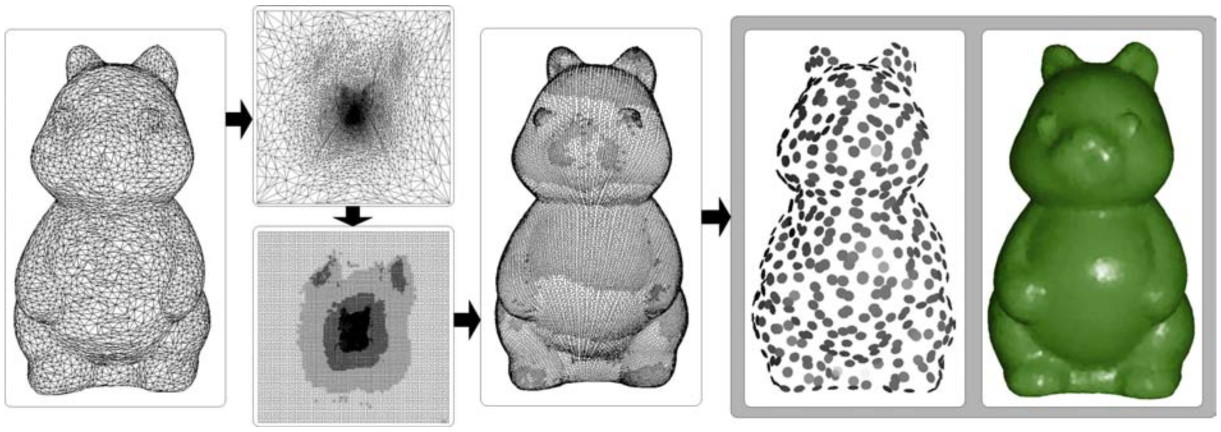

Figure 2: *Squirrel Mesh during the Mesh Processing Stages: Original; Parameterised; Adaptive Hilbert Curve in 2D; Adaptive Hilbert Curve in 3D; Uniform Sampling; Primitive surfel rendering*

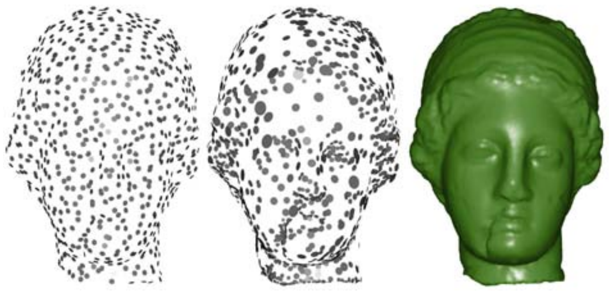

Figure 3: *Uniform, Curvature-Controlled and Rendered Resampling of Igea Mesh*

Mesh	$\mathbf{N_f}$	$\mathbf{N_v}$	$\mathbf{t_p}$	$\mathbf{t_c}$	$\mathbf{t_m}$	$\mathbf{t_s}$	$\mathbf{t}$
J Caesar	49K	24K	5.3	7	1	0.001	13.3
Squirrel	20K	10K	1.3	4	0.8	0.001	6.1
Igea	268K	134K	70	10	1.1	0.0015	81.1
Lion	40K	20K	152	5	1	0.001	158.1

Table 1: *Number of Faces* $\mathbf{N_f}$ *and Vertices* $\mathbf{N_v}$ *and Processing Times for Parametrisation* $\mathbf{t_p}$*, Hilbert Curve Generation* $\mathbf{t_c}$*, Mapping from Parameter Domain to Surface* $\mathbf{t_m}$*, Sampling the Surface* $\mathbf{t_s}$ *and Total Time* $\mathbf{t}$ *in Seconds for Test Meshes*

which is then mapped back onto the mesh and finally a uniform density splatting (1.5K points) and a primitive splat rendering (15K points). Figs. 3, 4, show uniform splatting, splatting according with density given by Gaussian curvature, and a primitive splat rendering for the Chinese Lion and Igea meshes (2K, 2K and 25K points respectively).

Table 1 gives run-times for each stage of our algorithm. Parameterisation ($\mathbf{t_p}$) using Floater's method [Flo97] takes 70 seconds for the 268K triangle Igea. Complex cuts, however, can result in longer parameterisation times, e.g. for the Chinese Lion. Generating the adaptive Hilbert curve ($\mathbf{t_c}$) depends solely on the maximum number of points required in the resampled surface. It typically takes less time than parametrisation: e.g. for the Igea, generating the space-filling curve to a precision necessary to generate 64K points takes about 10 seconds. The mapping stage of the algorithm ($\mathbf{t_m}$) takes about 1 second. After this pre-processing has been done, the mesh can be resampled ($\mathbf{t_s}$) in a fraction of a second regardless of the number of samples required (see Sect. 3.2 for what happens if the density becomes too high to be correctly represented on the Hilbert curve). After pre-processing, even large models with over 100K triangles can be resampled in less than 10 milliseconds. The main bottlenecks are parameterisation and adaptive Hilbert curve generation. The generation of this curve, however, has not been optimised, and could be further sped up, using either a GPU or multi-core CPU approach.

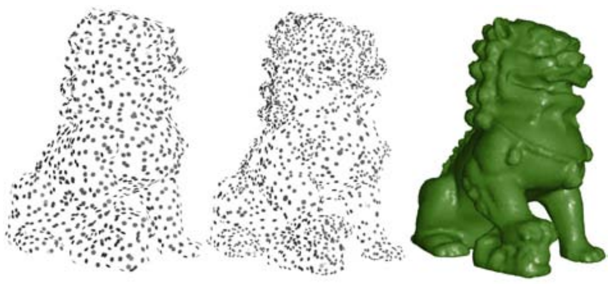

Figure 4: *Uniform, Curvature-Controlled and Rendered Resampling of Chinese Lion Mesh*

4.2. Distribution Quality and Discrepancy

We next assess the quality of the resampled distributions for several example meshes by measuring their discrepancies. We then consider how different distributions can have an impact on the number of samples required for common point based graphics rendering techniques, comparing our technique to other popular methods.

For distribution quality assessment we consider the star discrepancy measure [Zer68]: for all axis-aligned rectangular subsets of a planar rectangular domain, the difference between the ratios of points inside each subset and the relative area of that subset is calculated; the supremum of this difference is the discrepancy. For a given number of points N, the lower the discrepancy the better. Discrepancy tells us how well the new points are placed in terms of surface coverage, and thus how well they sample the underlying surface. The following tests demonstrate that the sample points produced by our algorithm behaves as expected for a low-discrepancy distribution. Theoretically, the discrepancy of such a sampling in $[0,1]^2$ should vary according to $O(N^{-1}\log^2 N)$ [Nie92]. Chazelle [Cha00] proves that a jittered sampling in $[0,1]^2$ has an absolute discrepancy of $O(N^{1/4}\sqrt{\log N})$ for any rotated box, providing a near-optimal rotationally-invariant distribution. As previously shown for differently shaped subset shapes on the plane and the sphere [QLME06], we generalise this idea to triangle meshes. We define a subset of the mesh as a contiguous set of triangles, grown from a random seed triangle to a random number of triangle rings. We then compare the fractional area of this subset of the mesh to the fractional number of points it contains.

We use this discrepancy measure to compare the points generated by our algorithm to random point distributions generated by the following method: for each triangle, given a sample set size N, we approximate the number of points n that should lie in that triangle by calculating the ratio $A_\triangle/A_{mesh}$, then generate each point by randomly interpolating the three vertices of that triangle. Note that this is locally, not globally, random, and actually improves the distribution. We compute the discrepancy for sets of new mesh vertices of size $N = 2^l$ and $N = 2^l + 2^{l-1}$ for $l = 1,\ldots,20$, resulting in 40 sets ranging from 2 (clearly useless in practice) to 1572864 samples. We plot a graph of the logarithm of discrepancy versus $\log N$, to which we fit a least squares line, allowing us to easily compare the gradients. For testing, we used an adaptive Hilbert curve where the Hilbert vertices H_l on the the mesh surface M_s are uniformly distributed, and $|\{H_l\}| \geq 16{,}000{,}000$, thus ensuring the ratio $\omega \geq 10$ as explained earlier. We sample N points on this curve for each test using a constant density of $\delta \equiv 1$ to give an even distribution of points on the surface.

Fig.5 shows the results for various test meshes on a logarithmic scale. In each case, our approach demonstrates discrepancy which scales better with N than the random distribution (which gave almost identical results for each mesh, so is only shown once). Gradients for the slopes shown in Table 2 back up this consistency, demonstrating a mean gradient of about -0.72 compared to -0.5 for the random sequence. These results mirror those in [QLME06] using low-discrepancy point samples generated on a parameterised sphere; they also correspond very closely to the known Hammersley distribution on the sphere [CF97]. This indicates

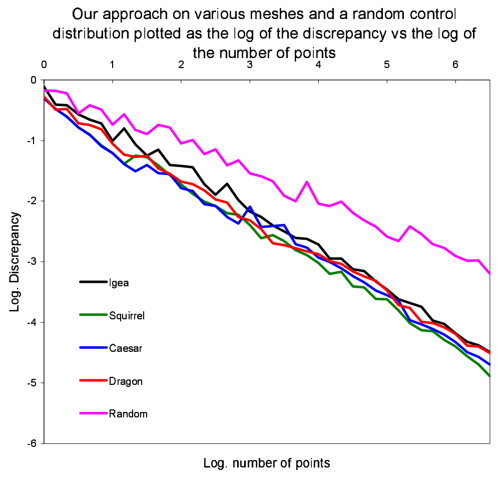

Figure 5: *Discrepancy Test Results*

Mesh	*Discrepancy*	*Metro*		
	Gradient	75%	50%	25%
J Caesar	-0.70	0.009	0.014	0.02
Squirrel	-0.74	0.008	0.01	0.018
Igea	-0.73	0.0032	0.013	0.017
Lion	-0.73	0.009	0.016	0.022

Table 2: *Gradients of Discrepancy Tests, and Hausdorff Distances for Test Meshes*

that our approach generates low-discrepancy point distributions on meshes.

Low-discrepancy and stratified distributions have been widely used in computer graphics, e.g. for applications such as radiosity [Kel97] and ray tracing [Coo86]. In [RWCS05], low-discrepancy and pseudo-random distributions were sampled from triangle meshes, though no investigation was done to assess the quality with respect to the final application: point based graphics. Almost all current techniques employ splatting or surfel rendering techniques [KB04] to provide a contiguous surface coverage. In Fig. 6, we have sampled 3K points in the unit square, drawn using solid circles, first using a random distribution, then the Niederreiter distribution, followed by our technique. It is clear that our distribution produces a considerably better surface coverage for this application (n.b. the distribution produced using our technique is not deterministic). As discussed in [SM03], sampling a Hilbert curve with a jittered 1D sampling is actually a generalised form of stratified sampling. Thus, dividing the plane up into cells and randomly placing a point within each cell would produce the same output. However, the advantages of our technique are: (i) the number of samples need not be a perfect-square (for a square parameter domain), (ii) we can maintain an evenly stratified distribution on an arbitrary surface, (iii) parametric distortion does not effect the neighbourhood and connectivity of the samples.

4.3. LOD and Viewpoint Dependent Rendering

In this section, we look at two applications of our framework: Levels-of-Detail (LOD) and Viewpoint Dependent Rendering (VDR).

We first discuss LOD representation of surfaces using our approach, using two methods. In the first method, we compute the distance d between the centre of the object and the camera. Some simple function (linear or non-linear) of d is used to determine the number of points N used to represent the surface as d changes. We initially pre-process the model, and then execute the equalisation step of the sampling process as d changes (see Sect. 3.4). This step typically takes less than 10 milliseconds, depending on the depth of the curve, allowing for interactive frame rates. A coarser approach, to speed up computation on large models, involves dividing the surface up into regions according to their distance from the camera, and give each region a different density which decreases with distance. The necessary computations are more expensive, but allow *local* level of detail control on a surface, which may be useful for large objects. Again, we can do the necessary computations at interactive frame rates. Fig. 7 shows local LOD control for the cow model, where the arrow indicates the user's viewing direction, but we present the points from the side to show the variation of density with depth. The distance computation can efficiently be divided between multiple CPU cores as each calculation is independent.

For VDR of a given surface, we compute the angle θ_l between the vector from the camera to a point H_l and the surface normal at point H_l:

$$\theta_l = \cos^{-1}\left(\frac{(C-H_l)\cdot N_l}{|C-H_l||N_l|}\right),$$

where C is the camera location and N_l is the surface normal at H_l. We adjust the discrete local density s_l according to θ_l:

$$s_l = \begin{cases} 1-\cos(\theta_l)+s_0 & \text{if } \theta_l > 0, \\ 0 & \text{otherwise,} \end{cases}$$

where s_l is the local density for vertex H_l and s_0 is a constant to specify the start of the density reduction. As the vectors $C-H_l$ and N_l become orthogonal, θ_l approaches 0, and the density increases, allowing us to increase the density around silhouette regions of the surface (see Fig. 8). To do this, we need to calculate s_l for each H_l, and then perform the equalisation step of the sampling process to produce a viewpoint dependent surface sampling. Fig. 8 demonstrates this process for a sphere shown from different angles relative to the camera direction. Our VDR implementation can recompute 50K sample points on an arbitrary surface approximately 40 times per second, and could be further optimised.

4.4. Remeshing

We next investigate another application of our framework: remeshing, which requires sampling an entirely new distribution of points on a surface, and constructing a suitable topology.

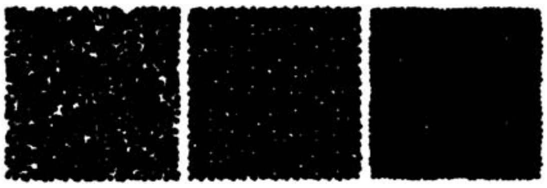
Figure 6: 3*K Points in* $[0,1]^2$*: Random, Niederreiter and our Method*

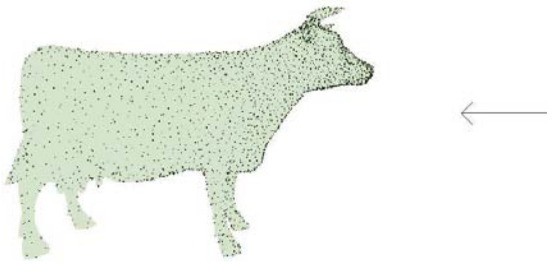
Figure 7: 4*K Points with Localised LOD (arrow represents scene camera)*

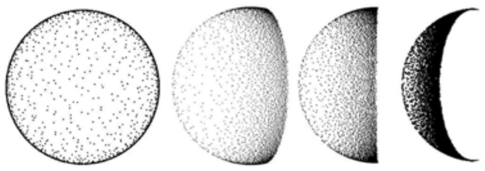
Figure 8: *Silhouette Enhancement: A Sphere Shown from Different Angles with a Fixed Camera*

To generate a remeshed surface, we simply generate a set of sample points p_l on the surface M_s, and triangulate them in the parameter domain using $O(n\log n)$ conformal Delaunay triangulation [She96]. We do not insert any Steiner points, as these would affect the prescribed density. If the surface is to be water tight, we must stitch any cuts used to open out the surface for parameterisation.

We measure the Hausdorff distance between our remeshed surfaces and the original mesh with Metro [CRS98] to assess how well the shape is preserved by our method. For each test mesh, we generated three remeshed surfaces with 75%, 50% and 25% of the original number of triangles, with user-defined density δ proportional to Gaussian curvature. In each case, we computed the Hausdorff distance as a percentage of the bounding box diagonal to normalise the error. Table 2 demonstrates the results, showing a Hausdorff distance of less than 0.01% for 75% of the points. Fig. 9 shows uniform remeshing and remeshing according to Gaussian curvature for the Julius Caesar head (15K vertices; 60% of original). The triangulation took 0.3 seconds, which is adequate to provide the user with interactive control over remeshing.

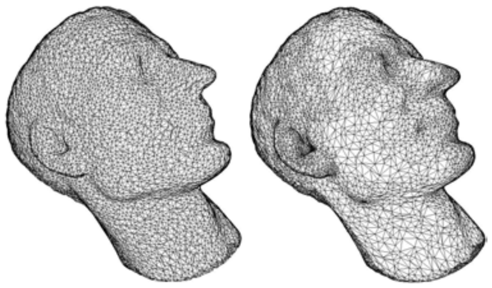

Figure 9: *Uniform and Curvature-Controlled Remesh of Julius Caesar Mesh*

4.5. Discussion

We now discuss the overall properties of our approach based on the above examples and results. The timing results show that, after pre-processing (including parameterisation) which takes about 80 seconds for a large mesh (268K triangles), we can perform density-controlled resampling of a mesh with hundreds of thousands of points at video frame rates or better. Note that during pre-processing, we generate a Hilbert curve taking into account the density function δ, so we generate a higher density curve where we plan to place more points on the surface. However, when resampling interactively, if the user changes δ so much that the Hilbert curve vertices are no longer dense enough in some places, we must further subdivide the curve locally to compensate, requiring extra computation time. For small changes, the distribution is initially approximated on the existing curve, whilst a background thread subdivides the curve and then redistributes the points. For larger density changes everywhere, it may be cheaper to increase the level of Hilbert curve globally, requiring about a tenth of the time taken to compute the initial curve (as no area measurements are required).

The numerical discrepancy measured for of the resampled points is in line with that for points generated on parametric surfaces using space-filling curve methods [QLME06], demonstrating log gradients of about ≤ -0.7. All models tested demonstrated similar results which are consistently better than for random sampling. We also showed that the stratified sampling output produced by our approach, when drawn with circular discs on the plane, provides better coverage with fewer points than other well-known low-discrepancy distributions, perhaps due to the directional invariance of the distribution. Examples of the variance exhibited by some sampling methods are shown in [QLME06], where the discrepancy of some distributions varies with respect to the sampling shape used to approximate the star-discrepancy; an undesirable quality for an arbitrary surface.

Our approach to level-of-detail representations allows arbitrary surfaces to be resampled in less than 10 milliseconds, whilst maintaining the qualities of the sampling, such as the discrepancy. We have also demonstrated a simple viewpoint dependent rendering scheme, showing reasonable frame-rates for fixed-complexity scenes.

We also considered the application of our ideas to remeshing. When measuring the Hausdorff distance of the remeshed surfaces, we obtain a low distance error between the surfaces for all meshes, which scales well with decreasing sample points. Metro gives an indication how well our remeshing technique preserves the overall shape. However, it does not necessarily tell us how well the shape of a mesh has been preserved locally. Other information, such as normals, and in particular, curvature tensors which define the local shape of the surface, might provide useful information for assessing the new mesh. In [LS04], a *depth weighted Hausdorff distance* is proposed, using local surface curvature variance as a weighting. We believe that such a measure could more accurately assess the quality of a surface remeshing process, and may better highlight feature loss due to regular resampling. A popular remeshing technique by Surazhsky and Gotsman [SG03] shows results for the Igea mesh, demonstrating an error of 0.003% for a 105% remesh. Remeshed to the same number of points, our technique demonstrates an error of just under 0.004%.

5. Conclusions

We have presented a mesh resampling method based on low-discrepancy point distributions. Our approach requires a simple surface parametrisation step which only has to fulfils a few basic conditions, and a fast method can be used. Our method is fast, allowing adjustment of the local sampling density at interactive rates after a pre-processing stage. The resulting distributions have low discrepancy and are well suited for point-based rendering as well as remeshing with low error. We have also demonstrated the use of our method for viewpoint dependent rendering and level-of-detail representations that allows models to be resampled at an arbitrary granularity in real-time.

References

[ACSD*03] ALLIEZ P., COHEN-STEINER D., DEVILLERS O., LÉVY B., DESBRUN M.: Anisotropic polygonal remeshing. In *Proc. ACM SIGGRAPH* (2003), Hodgins J., Hart J. C., (Eds.), vol. 22(3) of *ACM Trans. Graph.*, ACM Press, pp. 485–493.

[Ada06] ADAMS B.: *Point-Based Modeling, Animation and Rendering of Dynamic Objects*. PhD thesis, Katholieke Universiteit Leuven, 2006.

[AGGA05] ALLIEZ P., GIULIANA U., GOTSMAN C., ATTENE M.: *Recent Advances in Remeshing of Surfaces*. Research report, AIM@SHAPE EU Network, 2005.

[AGP*04] ALEXA M., GROSS M., PAULY M., PFISTER H., STAMMINGER M., ZWICKER M.: Point-based computer graphics. In *ACM SIGGRAPH 2004 Course Notes* (New York, NY, USA, 2004), ACM Press.

[AMD02] ALLIEZ P., MEYER M., DESBRUN M.: Interactive geometry remeshing. *ACM Trans. Graph 21*, 3 (2002), 347–354.

[BK01] BOTSCH M., KOBBELT L.: Resampling feature and blend regions in polygonal meshes for surface anti-aliasing. *Comp. Graph. Forum 20*, 3 (Sept. 2001), 402–410.

[BLM04] BHAKAR S., LUO L., MUDUR S. P.: View dependent stochastic sampling for efficient rendering of point sampled surfaces. In *WSCG* (2004), pp. 49–56.

[But68] BUTZ A.: Space filling curves and mathematical programming. *Information and Control 12* (1968), 314–330.

[CF97] CUI J., FREEDEN W.: Equidistribution on the sphere. *SIAM J. Sci. Comput. 18*, 2 (1997), 595–609.

[Cha00] CHAZELLE B.: *The discrepancy method: randomness and complexity*. Cambridge University Press, New York, NY, USA, 2000.

[Coo86] COOK R. L.: Stochastic sampling in computer graphics. *ACM Trans. Graphics 5*, 1 (1986), 51–72.

[CRS98] CIGNONI P., ROCCHINI C., SCOPIGNO R.: Metro: Measuring error on simplified surfaces. *Comp. Graph. Forum 17*, 2 (1998), 167–174.

[DLS05] DEY T. K., LI G., SUN J.: Normal estimation for point clouds: A comparison study for a Voronoi based method. In *Symp. Point-Based Graphics* (Stony Brook, NY, 2005), Alexa M., Rusinkiewicz S., Pauly M., Zwicker M., (Eds.), Eurographics Association, pp. 39–46.

[DMB99] DAVIES T. J. G., MARTIN R. R., BOWYER A.: Computing volume properties using low-discrepancy sequences. In *Geometric Modelling* (1999), pp. 55–72.

[Flo97] FLOATER M. S.: Parametrization and smooth approximation of surface triangulations. *J. Computer-Aided Geometric Design 14*, 4 (1997), 231–250.

[GY03] GU X., YAU S.-T.: Global conformal surface parameterization. In *Symp. Geometry Processing* (2003), pp. 127–137.

[KB04] KOBBELT L., BOTSCH M.: A survey of point-based techniques in computer graphics. *Computers and Graphics 28*, 6 (2004), 801–814.

[Kel97] KELLER A.: Instant radiosity. In *ACM SIGGRAPH* (1997), pp. 49–56.

[KF94] KAMEL I., FALOUTSOS C.: Hilbert R-tree: an improved R-tree using fractals. In *20th VLBD Conf., Santiago de Chile* (1994), pp. 500–509.

[KV03] KALAIAH A., VARSHNEY A.: Modeling and rendering of points with local geometry. *IEEE Trans. Vis. Comput. Graph 9*, 1 (2003), 30–42.

[LS04] LEE Y.-H., SHIM J.-C.: Curvature based human face recognition using depth weighted hausdorff distance. In *ICIP* (2004), pp. 1429–1432.

[LWC*02] LUEBKE D., WATSON B., COHEN J. D., REDDY M., VARSHNEY A.: *Level of Detail for 3D Graphics*. Elsevier Science, New York, NY, USA, 2002.

[LWMB03] 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.

[MDSB02] MEYER M., DESBRUN M., SCHRÖDER P., BARR A. H.: Discrete differential geometry operators for triangulated 2-manifolds. In *VisMath III* (2002), pp. 35–57.

[Nie92] NIEDERREITER H.: *Random Number Generation and Quasi-Monte Carlo Methods*. SIAM, 1992.

[QLME06] QUINN J. A., LANGBEIN F. C., MARTIN R. R., ELBER G.: Density-controlled sampling of parametric surfaces using adaptive space-filling curves. In *Proc. Geometric Modeling and Processing* (2006), no. 4077 in LNCS, Springer, pp. 658–678.

[RWCS05] ROVIRA J., WONKA P., CASTRO F., SBERT M.: Point sampling with uniformly distributed lines. *Eurographics Symp. Point-Based Graphics* (2005), 109–118.

[SAL06] SABORET L., ALLIEZ P., LÉVY B.: Planar parameterization of triangulated surface meshes. In *CGAL-3.2 User and Reference Manual*, Board C. E., (Ed.). 2006.

[SG03] SURAZHSKY V., GOTSMAN C.: Explicit surface remeshing. In *Symp. on Geom. Proc.* (2003), pp. 20–30.

[She96] SHEWCHUK J. R.: Triangle: Engineering a 2D quality mesh generator and delaunay triangulator. In *Applied Comp. Geom.: Towards Geom. Eng.*, Lin M. C., Manocha D., (Eds.), vol. 1148 of *LNCS*. Springer, 1996, pp. 203–222.

[SM03] STEIGLEDER M., MCCOOL M.: Generalized stratified sampling using the Hilbert curve. *J. Graphics Tools 8*, 3 (2003), 41–47.

[SSV07] SANKARANARAYANAN J., SAMET H., VARSHNEY A.: A fast all nearest neighbor algorithm for applications involving large point-clouds. *Comput. Graph. 31*, 2 (2007), 157–174.

[WK04] WU J., KOBBELT L.: Optimized sub-sampling of point sets for surface splatting. *Comput. Graph. Forum 23*, 3 (2004), 643–652.

[WZK05] WU J., ZHANG Z., KOBBELT L.: Progressive splatting. In *Symp. Point-Based Graphics* (Stony Brook, NY, 2005), Alexa M., Rusinkiewicz S., Pauly M., Zwicker M., (Eds.), Eurographics Association, pp. 25–32.

[Zer68] ZEREMBA S.: The mathematical basis of monte carlo and quasi-monte carlo methods. *SIAM Review 10*, 3 (1968), 303–314.

[Zwi03] ZWICKER M.: *Continuous Reconstruction, Rendering, and Editing of Point-Sampled Surfaces*. PhD thesis, ETH Zurich, 2003.