Low-discrepancy Point Sampling of 2D Manifolds for Visual Computing

A thesis submitted in partial fulfilment
of the requirement for the degree of Doctor of Philosophy

Jonathan Alexander Quinn

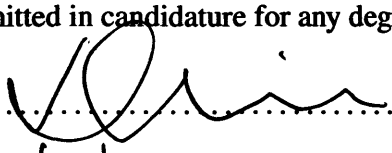
March 2009

Cardiff University
School of Computer Science

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Declaration

This work has not previously been accepted in substance for any degree and is not concurrently submitted in candidature for any degree.

Signed  (candidate)

Date 10.11.09

Statement 1

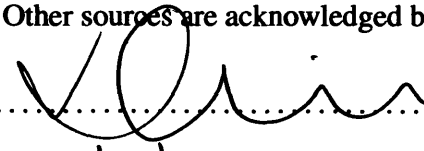
This thesis is being submitted in partial fulfilment of the requirements for the degree of PhD.

Signed  (candidate)

Date 10.11.09

Statement 2

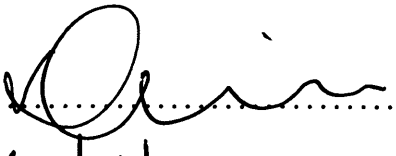
This thesis is the result of my own independent work/investigation, except where otherwise stated. Other sources are acknowledged by explicit references.

Signed  (candidate)

Date 10.11.09

Statement 3

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed  (candidate)

Date 10.11.09


CARDIFF UNIVERSITY
17 NOV 2009
PRIFYSGOL CAERDYDD

To
Jan and Andy
For never quite losing their patience with me on this.

Abstract

Point distributions are used to sample surfaces for a wide variety of applications within the fields of graphics and computational geometry, such as point-based graphics, remeshing and area/volume measurement. The quality of such point distributions is important, and quality criteria are often application dependent. Common quality criteria include visual appearance, an even distribution whilst avoiding aliasing and other artifacts, and minimisation of the number of points required to accurately sample a surface. Previous work suggests that discrepancy measures the uniformity of a point distribution and hence a point distribution of minimal discrepancy is expected to be of high quality. We investigate discrepancy as a measure of sampling quality, and present a novel approach for generating low-discrepancy point distributions on parameterised surfaces.

Our approach uses the idea of converting the 2D sampling problem into a 1D problem by adaptively mapping a space-filling curve onto the surface. A 1D sequence is then generated and used to sample the surface along the curve. The sampling process takes into account the parametric mapping, employing a corrective approach similar to histogram equalisation, to ensure that it gives a 2D low-discrepancy point distribution on the surface. The local sampling density can be controlled by a user-defined density function, e.g. to preserve local features, or to achieve desired data reduction rates.

Experiments show that our approach efficiently generates low-discrepancy distributions on arbitrary parametric surfaces, demonstrating nearly as good results as popular low-discrepancy sampling methods designed for particular surfaces like planes and spheres. We develop a generalised notion of the standard discrepancy measure, which considers a broader set of sample shapes used to compute the discrepancy. In this more thorough testing, our sampling approach produces results superior to popular distributions. We also demonstrate that the point distributions produced by our approach closely adhere to the blue noise criterion, compared to the popular low-discrepancy methods tested, which show high levels of structure, undesirable for visual representation.

Furthermore, we present novel sampling algorithms to generate low-discrepancy distri-

butions on triangle meshes. To sample the mesh, it is cut into a disc topology, and a parameterisation is generated. Our sampling algorithm can then be used to sample the parameterised mesh, using robust methods for computing discrete differential properties of the surface. After these pre-processing steps, the sampling density can be adjusted in real-time. Experiments also show that our sampling approach can accurately resample existing meshes with low discrepancy, demonstrating error rates when reducing the mesh complexity as good as the best results in the literature.

We present three applications of our mesh sampling algorithm. We first describe a point-based graphics sampling approach, which includes a global hole-filling algorithm. We investigate the coverage of sample discs for this approach, demonstrating results superior to random sampling and a popular low-discrepancy method. Moreover, we develop levels of detail and view dependent rendering approaches, providing very fine-grained density control with distance and angle, and silhouette enhancement. We further discuss a triangle-based remeshing technique, producing high quality, topologically unaltered meshes. Finally, we describe a complete framework for sampling and painting engineering prototype models. This approach provides density control according to surface texture, and gives full dithering control of the point sample distribution. Results exhibit high quality point distributions for painting that are invariant to surface orientation or complexity.

The main contributions of this thesis are novel algorithms to generate high-quality density-controlled point distributions on parametric surfaces and triangular meshes. Qualitative assessment and discrepancy measures and blue noise criteria show their high sampling quality in general. We introduce generalised discrepancy measures which indicate that the sampling quality of our approach is superior to other low-discrepancy sampling techniques. Moreover, we present novel approaches towards remeshing, point-based rendering and robotic painting of prototypes by adapting our sampling algorithms and demonstrate the overall good quality of the results for these specific applications.

Acknowledgements

I would firstly like to thank my supervisors, Frank Langbein and Ralph Martin, for their constant and patient help, support and hard work throughout this research, and for their seeming ability to discuss things with me 24 hours a day!

I'd like to thank my colleagues in the Vision and Geometry group, namely, Dave, Paul, Darren, Shenglan, Ming and Weishi for all their suggestions, comments and the many interesting discussions.

I also very much appreciate the support from Nick Fiddian and the whole School of Computer Science at Cardiff during this research. In addition, I am grateful for the advice and for the many discussions with my colleagues at Cardiff, Phil and Florian, during this work.

Further, I'd like to thank the colleagues that I worked with outside the school, namely Gershon Elber, Jonathan Corney and Finlay McPherson.

Finally, I'd like to thank my parents, Andy and Jan, for maintaining a constant interest in my work, and supporting me throughout all of it, and my friends, who, generally speaking, managed to listen to me talk about it the entire time with an unnerving tolerance.

Contents

Chapter 1

Introduction

Generating a high quality point sampling on a manifold, such as a parametric or polygonal mesh, is an important requirement in many areas of computational geometry and graphics. The sampling should be appropriate for its application, such as for point-based rendering, finite element methods or remeshing. For example, we may have a polygonal mesh that we wish to render using a point-based approach. We can represent large flat areas with relatively few polygons in a mesh, and thus few points. However, we need a relatively dense sampling, even of flat areas, if we wish to render them using a point-based approach, and thus we must resample the surface to produce a useful set of points.

Whilst the application affects the type of point sampling required, more general quality criteria also exist for surface sampling, which are desirable for a wide range of applications. A high quality sampling of points should allow a surface to be sampled and represented to a given accuracy with as few points as possible. Control of the density is also important, and allows us to use points more efficiently, such as sampling more points in areas with greater detail or higher curvature. A sampling of points should also ideally not be structured in such a fashion that the sampling itself detracts from what is being sampled; a grid sampling, for example, has a clear, regular structure.

Producing high-quality distributions on arbitrary surfaces is, however, a complicated and generally unsolved problem. In this work, we describe algorithms to produce high-quality point samplings on parametric surfaces and polygonal meshes. Our approach allows density-controlled, evenly distributed, point samplings to be generated quickly. We also investigate and further develop numerical measures in order to assess the quality of such point samplings.

In Section 1.1 we begin with an overview of surface sampling, focusing on the problems involved and how they have been addressed. We then investigate sampling quality in Section 1.1.1, followed by an overview of various applications in Section 1.1.2, including those developed in this work. In Section 1.2 we describe our approach to surface sampling, followed by the contributions of our work in Section 1.3. In Section 1.4 we give an

overview of the remainder of this thesis.

1.1 Sampling Two-dimensional Manifolds

Sampling is an essential process used in many areas of mathematics and engineering, and is intrinsically necessary in computer science. An example of its importance in computer science is the sampling of an unknown function to determine some of its properties in order to discretely represent it for further processing, such as visualisation. Sampling is a statistical method that is used to determine certain properties of a function in a domain. Given set A, choose a set of points $B \subset A$ to represent A. The way in which points B are distributed within A has implications on the sampling quality, and thus how well A is represented by B, or which properties of A may be estimated by B. There are many methods to sample a set A, for example, selecting sample points according to a uniform random probability distribution, or points on a regular grid, the output being a finite discrete point set. In this work, we consider the generation of point sets on manifolds. An s-dimensional manifold is a set A with a topology T, such that for each $x \in A$, there is an open neighbourhood $K \in T$ which is homeomorphic to $\mathbb{R}^s$: locally, a manifold looks like $\mathbb{R}^s$. In this work, we focus on Riemannian two-manifolds, embedded in $\mathbb{R}^3$ with a Euclidean metric, as these cover a large class of surfaces used in computational geometry, visualisation and graphics. If the set is parameterised via a function, or more generally, an atlas provided for the manifold, sampling can be performed in the parameter domain. The properties of the parameterisation, however, must be considered in order to achieve suitable sampling qualities.

Various errors can be introduced when sampling a function. A measurement error may occur when the value at the sample location varies from the actual value of the function at that location. A discretisation, or sampling, error may occur if a function with a continuous domain (e.g. an s-dimensional manifold with $s > 0$) is sampled, and therefore not all values can be sampled. We address this discretisation error in this work. Aliasing is a particular effect of this sampling error, and can occur if the sampling density is not high enough (see Section 1.1.1). The distribution may be controlled by a density function, for example, sampling in places of high function variance. Whilst sampling has many applications in computer science, we are mainly concerned with surface sampling for point based graphics [103, 104, 106], remeshing [7, 119] and the painting of prototype models [83]. Other related applications that we only briefly discuss include improvement of rendering quality in computer graphics [32, 63], providing efficient sampling for probabilistic area/volume measurement techniques [36, 76], and many others (see Section 1.1.2).

The main concept of sampling theory is that of discretisation; a continuous signal or function must be sampled and represented in a discrete domain. Thus, an obvious result of this is that information is lost. The question is whether the original, continuous signal can be reconstructed from the sampling, or rather, how much of it. Significant work in this area was done by Nyquist [94] and Shannon [111], which led to the following thorem: to avoid information loss when sampling, i.e. for a signal to properly be reconstructed, the sampling rate (or density) must be twice that of the highest frequency in the data. If this sampling rate is not achieved, the resulting reconstruction will not contain all of the original data, and thus may be aliased. In computer graphics, the same problem exists. However, in this field, reconstruction of the continuous function is usually not important; the onus is that the discrete output *looks* smooth (or continuous) to the viewer. A common solution to this problem is to perform some variation of supersampling (taking more samples), then averaging (possibly weighted by a function such as a Gaussian) the values of the samples.

The quality of the output produced from the discretisation of a function is largely dependent on the quality of the sampling. Whilst quality is something that we investigate throughout this work, we define a high quality sampling as a uniformly distributed set of points that are of low-discrepancy, are equally distributed, and have minimal visual structure. High quality sampling techniques are aimed at improving the discretisation results; fewer samples are required, providing a faster rate of error reduction, and can have a predictably bounded error. One measure that addresses this quality of sampling is discrepancy, which essentially measures the equidistribution (or uniformity) [92] of a sample distribution. Zaremba [132] and Niederreiter [92] demonstrate numerous standard methods to measure the discrepancy of a set of samples. However, conceptually, discrepancy can be described as the deviation from a regular, rectangular sampling grid [92]; a low discrepancy indicates a small deviation from such a grid. The property of low-discrepancy is generally advantageous [29], and achieving this without using a regular grid is ideal, in order to avoid aliasing problems [87]. Aliasing occurs when a continuous function is not sampled densely enough, and is very apparent with evenly-spaced samples because it produces a very obvious, regular, error. The number of samples required for a grid sampling also scales poorly as the number of dimensions of the sample domain increases [92]; for an s-dimensional cube, there must be N^s samples to achieve the same discretisation, where N is the number of samples required to discretise the unit interval. Uniform random sampling on the other hand does not suffer from aliasing problems, and the number of samples required does not increase exponentially as the number of dimensions increases. However, there is no guarantee of an even distribution (implying a relatively constant distance between neighbouring samples). Whilst low-discrepancy sampling methods do not

suffer from the exponential growth rate associated with grid-sampling, it has been shown that in very high dimensional cases, random sampling actually produces better results[92].

Various different methods have been developed for the generation of low-discrepancy samples for use in Monte Carlo integration, including Hammersley [54], Halton [52], Sobol [116] and Niederreiter [91] sequences; the later two are generally considered the leading techniques in this particular field. It has been shown that using low-discrepancy sequences (quasi Monte Carlo [105]), numerical integral computations converge to a solution much faster than with random distributions [126]. In fact, it is accepted that as the discrepancy of a set drops, so does the error of an integral evaluation of a function [29]. Discrepancy bounds have been determined for many distributions [92], and practically, can be approximated using numerical methods [38]. Whilst much of the original interest is related to very high dimensional integral problems, other fields have seen the benefit of low-discrepancy sampling. For example, Shirley [115] used discrepancy as a sampling quality measure in computer graphics, and investigated a number of methods to produce such a sampling, also considering various approaches to measuring the discrepancy discussed by Zaremba [132]. Other distributions which have commonly been used in computer graphics include jittered sampling, dart throwing, N-rooks sampling, and Poisson disc sampling among others, all of which are discussed by Shirley [115]. However, most work focuses on planar distributions, and does not consider the discrepancy of surface sampling. Producing high-quality low-discrepancy distributions on arbitrary surfaces, whilst maintaining good visual quality and density control is a complex problem, and has not been solved in the literature. Thus, in our work, we develop an approach to solve this problem.

The problem of low-discrepancy sampling of an arbitrary manifold is that inevitably, it either must be performed in a space it has been embedded in (typically $\mathbb{R}^d$ with $d \geq s$) or in a subset of $\mathbb{R}^s$ as the domain of a parameterisation $\mathbb{R}^s$ [43]. For example, a 2D surface embedded in $\mathbb{R}^3$ may be parameterised over a subset P of the unit square in $\mathbb{R}^2$ via some function $f : P \subset [0,1]^2 \rightarrow \mathbb{R}^3$. P is defined as a subset of $[0,1]^2$ for normalisation. Various techniques consider manifold sampling in the embedding space [7, 106], though it can be a very computationally expensive problem (especially for complex manifolds or dimensions greater than $\mathbb{R}^3$), and the results supporting the quality of the final sampling are often weak. Thus, as mentioned, we parameterise the two-manifold and sample in the parameter domain, P, simplifying the problem. However, if we sample $[0,1]^2$ with an equidistributed set of samples S_p, and use the parametrisation $f(S_p)$ to map these samples to the original manifold, the result is a set of samples S_m of the two-manifold which does not exhibit the same equidistribution in general. We refer to this as parametric distortion, as areas of the parameter domain are not mapped to equal-sized areas on the manifold, and

angles may not be maintained. This problem is common for general surfaces, and finding a mapping that preserves both areas and angles is very difficult. A good example is that of cartography; to view a planar version of the Earth on a map, it must be parameterised, and thus stretched irregularly in some manner (a spherical object gets distorted to varying degrees depending on the parameterisation method chosen) [43]. Thus, choice of parameterisation is important, generally falling into two categories: equiareal (area preserving), or conformal (angle preserving) [43]. That is, locally, patches only maintain their area, or only their angles, the process of which is investigated in greater detail in Section 6.1.1.

1.1.1 Low-discrepancy Sampling

The discrepancy of a distribution can be investigated numerically, and also with regard to its theoretical error bounds for the approximate evaluation of multi-variate integrals. We now introduce the concept of numerical discrepancy, which is investigated in detail in Section 2.1.1. The star discrepancy [53] of a point set in the unit square $[0, 1]^2$, can be defined as the supremum, over all rectangles with one corner at the origin, of the error in estimating the area of the rectangle as the ratio of the points inside it to the total number of points in the unit square. Computing the discrepancy in this way demonstrates how well a sampling is equidistributed with respect to axis-aligned rectangles, and it demonstrates how quickly this error drops with increasing number of sample points.

An important consideration is that a single sampling method may not be the best solution for all problems, and that a distribution that is of low discrepancy does not guarantee its practicality for a particular application. For example, the Niederreiter sequence generally displays the fastest reduction in discrepancy as the sampling density is increased when using axis-aligned rectangular sampling areas for the discrepancy measure. The Niederreiter sequence (and others) use a construction method called (t, m, s)-nets [91], which produce axis-aligned lattice distributions (see Section 2.1.2). As a result, using different sample shapes when measuring the discrepancy produces less optimal results (see Section 5.1.3). This is very relevant, for example, when performing anti-aliasing on an image [39], as aliased lines are likely to be not axis-aligned. Another example is that of distributions produced by relaxation, which do not generally perform as well as other sampling methods in numerical discrepancy experiments [115], but are useful for the generation of regular, equilateral, triangle meshes [19] (which can simplify computational geometry problems and reduce the likelihood of numerical errors).

If the discrepancy for a spread of sample densities is calculated and plotted, the gradient can be used to assess how well the discrepancy scales with sample density. However,

discrepancy is not the only measure that can be used to assess sample distributions. Blue noise is categorised by a noisy spectrum lacking any concentrated spikes, with a deficiency of low-frequency energy [130], and was first formally applied in a graphics context in [86]. As described in [120], the advantages of blue noise in the context of dithering are that it does not clash with the image by adding its own structure, or degrade it by being *too* unstructured. Whether or not a distribution fulfils the Blue-noise criterion can be decided by estimating and plotting the power spectrum of the distribution. The method of approximation using Bartlett's method [10] is detailed in [120]. Blue noise has a specific power spectrum. Therefore, if a distribution displays a spectrum which correlates well with this known behaviour, it is said to fulfil the Blue-noise criterion. Blue noise and its measurement are discussed in more detail in Section 5.3.

A common testing method for the assessment of resampling (in particular for downsampling) techniques, commonly used in the field of remeshing, is the Hausdorff distance [28] (or some simplified version of it). The Hausdorff distance is used to measure the approximation error between a surface and a resampling of the same surface. This error may be measured in a variety of ways, although a popular method is to align the objects, locally sample patches with a scan-line or random distribution, then calculate the Hausdorff distance

$$d_{\mathrm{H}}(S_1, S_2) = \max_{p_1 \in S_1} (\min_{p_2 \in S_2} (p_1, p_2)) \tag{1.1}$$

where S_1 is a surface, and S_2 an approximation of S_1. The symmetric Hausdorff distance is then defined as $d_{\mathrm{S}}(S_1, S_2) = \max(d_{\mathrm{H}}(S_1, S_2), d_{\mathrm{H}}(S_2, S_1))$. Either distance, $d_{\mathrm{H}}(S_1, S_2)$ or $d_{\mathrm{S}}(S_1, S_2)$, may be used to quantify how similar two models are. Also, the measure can be used as a criterion for mesh decimation: to remove a vertex and hence simplify a model, the Hausdorff distance is calculated between the original surface and the surface with one vertex missing for each vertex that makes up the surface. The vertex that results in the smallest increase in distance is chosen (being the least significant), and removed from the mesh. The Hausdorff distance is conceptually simple, and in Section 8.3, we discuss work investigating a more significant assessment of surfaces, taking local differential properties such as curvature variance into account.

1.1.2 Applications

There are many practical applications of low-discrepancy sampling methods in a variety of fields. In this section, we briefly look at two popular uses of existing methods, including measurement (area and volume computation) and rendering (radiosity, ray tracing and texture filtering). We then investigate the problem of surface sampling and representation

(mesh generation, point-based graphics and prototype painting), looking at the applications developed using our algorithms (see Chapter 7).

Monte Carlo methods are a fast approach for the discrete evaluation of multivariate integrals using random sampling. As already discussed, quasi Monte Carlo methods are a deterministic alternative to random sampling, and have been shown to reduce the number of samples needed to converge upon a solution within a prescribed error bound, and maintain a consistently higher level of confidence in the evaluation [29] where high confidence implies low variation in the convergence quality of the function being evaluated. Applying the quasi Monte Carlo method to measurement of the area and volume of objects has been investigated in [75, 76]. Results show that fewer points are required to measure these properties compared to using a random sampling.

In applications such as radiosity [63] and ray-tracing [32], important techniques in photo-realistic rendering, low-discrepancy sampling helps in numerous ways. Distributed ray-tracing [32], for example, calculates the light intensity at a sub-pixel level and computes the average to calculate the overall intensity for a pixel. If the sampling rate is not high enough to capture the multivariable signal, then, according to the Nyquist-Shannon sampling theorem, data will be lost, resulting in artifacts such as Moiré effects and aliasing in the output. As explained in Section 1.1.1, the non-gridded patterns of low-discrepancy samples reduces the aliasing effect, although it also results in an increase in noise [32], and fewer samples are required when compared to a random distribution.

A further application of low-discrepancy distributions is surface sampling. Resampling is a technique used for downsampling (or decimation) [27], upsampling (generally using higher-order approximations), and redistribution of vertices on a model (which can be useful to convert it to another surface representation, as well as a primary focus of remeshing). Using low-discrepancy distributions to sample surfaces provides the same advantages as in the rendering applications described above; equidistribution is optimal, yielding a good approximation of the surface function, aliasing of the model is reduced, yet no grid-like patterns are present and no sampling artifacts are introduced (such as obvious strips of almost equilateral triangles on a resampled mesh). However, the field has not been particularly well investigated (see Section 3). Thus, in Chapter 7 a number of practical questions are investigated and addressed. Does low-discrepancy sampling accurately sample the features of the existing model, and, is the resulting low-discrepancy sampling a *good* method for then rendering the model? Also, can two distributions with a comparably low discrepancy produce largely differing output when rendered? An example of use of low-discrepancy distributions for surface sampling is given by Rivora et al [106], who investigate using bundles of lines with a low-discrepancy distribution to resample meshes,

in a similar way to sampling methods used for surface area computation [75].

In many applications of surface representation, it is not clear exactly what qualifies as a good distribution. To complicate things, a certain distribution may sample geometry well, but may not prove particularly good for rendering. When sampling an existing manifold, during the discretisation or resampling stage, a distribution that is best at capturing the highest amount of information is generally desirable, whilst for rendering, a sampling that will give a more regular, and thus smoother output is often favoured. Two popular discrete surface representation methods, meshes and points, both rely on high-quality distributions for a visually appealing output and efficient use of samples. However, their requirements can be quite different. For example, in triangle meshing, one perceived advantageous quality is to have equilateral triangles [19], implying a degree of grid-like uniformity. The generally accepted reason for this is that computational geometry problems (such as containment and boundary computation) are less robust when performed on a set of *thin* or degenerate triangles. In point-based graphics, the topological structure of the mesh is not present, and thus such problems do not arise. However, other problems exist, such as coverage; because points are not connected to their neighbours, a 'watertight' representation of the surface does not exist in object-space. Pfister et al. suggest that points should be extended to give a disc representation [98], forming a hole-free rendering in image-space. Thus, equidistribution is often a desired quality for point-based rendering (equidistribution and coverage are investigated in detail in Section 7.1.1). It is clear that there is not a single 'best' solution to all sampling problems.

1.2 Our Sampling Approach

We now introduce the basic idea of our point sampling approach and the methods we use to test it. We seek to generate an equidistributed set of points with low discrepancy on manifold surfaces. Furthermore, we wish to control the sampling density, e.g. sample according to local differential properties such as surface curvature. The generated point distributions should be useful in a variety of applications, particularly for point-based graphics, remeshing, and prototype painting.

Our approach generates low-discrepancy point distributions on arbitrary surfaces by distributing points along a space-filling curve mapped onto the manifold. Given a parameterised two-manifold (referred to here as a surface), we map a space-filling curve from the parameter domain onto the manifold and distribute points along this curve, such that they give a density controlled low-discrepancy distribution on the manifold. By generating the space-filling curve, the problem of distributing points in 2D is reduced to sampling a curve

appropriately in the parameter domain. Sample points are placed along the space-filling curve using an idea similar to histogram equalisation—if we interpret a histogram as a frequency function of the intensities of an image, histogram equalisation is a process by which a mapping is found between the original distribution of intensities, and a uniform distribution of intensities by spreading the samples evenly [59]. Adaptive generation of the space-filling curve allows us to handle parametric distortions where, for example, a small area in the parameter domain is mapped to a large area on the surface. The space-filling curve not only converts the 2D problem to a 1D problem, but also provides the additional benefit of very good spatial localisation of the points, although this is largely dependent on the curve used (see Section 5.4). Localisation, or spatial coherence, means that points close in Euclidean space are generally close along the curve, which is advantageous for problems which require, for example, route planning through a series of points [99] (see Section 7.3).

The primary method used in our work for evaluating the quality of sampling is the numerical discrepancy measure. In Section 1.1.1, we introduced the star discrepancy measure and how we expand our experiments to include various sample shapes. We use various sample shapes to give a better indication of the sampling quality of non-rectangular, axis-aligned, subsets—especially relevant as we investigate the sampling of meshes (see Section 6), and thus we are interested in the quality of a subset of sample points within, say, a triangle of the mesh.

Thus, when measuring the discrepancy of a sample distribution, as discussed in detail in Section 2.1.1, we also use non-axis aligned subset shapes, and measure the discrepancy of surface distributions. When measuring the numerical discrepancy with axis aligned rectangles, our approach produces comparable results to techniques developed specifically to have low discrepancy when measured in this way in a $[0, 1]^n$ domain. For non-axis aligned shapes, well known low-discrepancy constructions demonstrate worse results, whereas our approach maintains almost exactly the same discrepancy for every shape subset and surface tested. Non-axis aligned shapes include semi-circles and triangles, and surfaces include a parametric sphere and arbitrary triangle meshes.

We also investigate the fulfilment of the blue noise criterion by analysing the radially averaged power spectrum of the point sampling. This measure allows us to investigate not only the uniformity and equidistribution of the points, but also the structural regularity. By fulfilling the blue noise criterion, we can ensure that undesirable structure, highly noticeable by the human visual system [120], is not being added by our point sampling.

In order to see how well this approach performs for mesh resampling, the Hausdorff distance is used to assess the accuracy of remeshed surfaces constructed from points sampled

using our algorithm (see Section 6.2.2). Also, whilst assessment of the results is highly subjective, the output surfaces for the applications of point based rendering and remeshing are also visualised (see Sections 7.1, 7.2), as this is still important for use in rendering. Finally, we investigate our sampling approach for the problem of prototype painting (see Section 7.3) by analysing the painted output. All of these results provide strong evidence that our novel approach is a fast and effective way to produce equidistributed low-discrepancy distributions on arbitrary manifolds, that are highly flexible for use in a variety of applications, and can, in certain environments, provide unmatched real-time performance.

1.3 Contribution

The novel contributions of this work are as follows:

- We demonstrate algorithms for density-controlled sampling using low-discrepancy points on parameterised surfaces, based on a novel adaptive space-filling curve sampling approach and reparameterisation.
- We provide an analysis of solutions to the problem of low-discrepancy sampling of manifolds, and the methods used to approach the problem. The analysis includes a qualitative assessment of discrepancy and blue noise, space-filling curve clustering and localisation, the effect of 1D deterministic and probabilistic sampling sequences on the output, and the use of different space-filling curves. We also demonstrate an expanded set of numerical measures for the star discrepancy, highlighting the limitations of testing using only the standard measure.
- We describe algorithms that build upon our space-filling curve sampling approach to allow for high-quality low-discrepancy sampling of triangle meshes. We describe methods to compute the discrepancy of these point distributions, and demonstrate high quality results in line with the results for the parameterised surface sampling. We also compute an approximation error measurement for decimated meshes, showing that those produced using our approach demonstrate errors as small as those produced by the best methods in the literature.
- We demonstrate three applications built from our mesh sampling approach. Firstly, we describe a point-based graphics sampling application, with hole-filling improvements, real-time level of detail, and view-dependent rendering control. We then introduce a high-quality remeshing algorithm. Finally, we describe a novel framework

for sampling and painting engineering prototypes, based on collaborative work, including texture sampling and dithering control.

1.4 Overview

The structure of the rest of this thesis is as follows. Chapter 2 looks at related work on low-discrepancy sampling, space-filling curves, and differential geometry. Chapter 3 reviews the literature on point-based graphics, remeshing, measurement and painting. Chapter 4 introduces the main algorithms for our sampling method, including space-filling curve generation and point sequence generation. Chapter 5 evaluates our approach, comparing it to existing approaches, looking at discrepancy, the blue noise criterion, and the clustering properties of various space-filling curves. Chapter 6 describes our mesh-sampling algorithms and evaluates the output, including discrepancy and an approximation error measure. In Chapter 7, applications of the mesh-based sampling approach are introduced and evaluated, covering point-based graphics, remeshing and prototype painting. Finally, Chapter 8 summarises the results and contributions, and looks at the future work arising from this research.

Chapter 2

Sampling and Related Mathematical Tools

This chapter explains the important mathematical concepts used in this thesis. First we consider sampling and discrepancy, and discuss how the latter can be defined and measured. The low-discrepancy distributions applied in this thesis are then defined and their generation techniques are briefly discussed. In Section 2.2, we define the concept of space-filling curves, and investigate various particular curves and their construction methods. Space-filling curves are utilised in our approach for the generation of low-discrepancy sequences, as introduced in Section 1.2. We then explain the differential geometry concepts that are applied in this work to calculate surface properties such as curvature and area, allowing for density-controlled sampling. We then discuss the discrete differential geometry approximations used for computing surface properties of discrete mesh surfaces.

2.1 Low-discrepancy Sampling

This Section introduces and discusses the concept of discrepancy, along with its intended computational applications. A uniform distribution is one which, in the limit, results in a uniform probability distribution; i.e. that there is an equal likelihood for a single point being at any position. However, uniformity therefore only refers to the probability of each point individually, and hence, a uniform, random, point distribution is not correlated. Low-discrepancy sequences are correlated, and the probability of a point being at some position is dependent on its position in the sequence. This correlation means that the quality of a whole point set is considered, rather than a single, independent point, and this results in a more even coverage of a domain. This correlated coverage property is measured by discrepancy, which allows us to quantitatively assess how evenly distributed a sampling is. Sequences with low discrepancy are desirable for this reason. In this section, we

focus on discrepancy and low-discrepancy distributions in detail, including the principal reason for their development and how they can be constructed. We are interested primarily in 2D distributions, and how the discrepancy of these distributions can be measured. The low-discrepancy methods that we consider fall into three main categories: probabilistic methods, such as jittered sampling (Section 2.1.2), maximally-avoiding methods, such as the Halton sequence (see Section 2.1.2), and lattice-based approaches, such as the Niederreiter sequence (see Section 2.1.2).

2.1.1 Discrepancy

Discrepancy is a concept that became very important when it was demonstrated that as the discrepancy of a sequence decreased, so did the approximation error of a Monte Carlo evaluation of a multivariate integral [29]. Discrepancy is an important measure of the quality of a sequence for such methods, but has been employed in many different fields, such as computer graphics [115] and surface representation [106]. Discrepancy was founded on this relationship with Monte Carlo error bounds, thus we introduce it in this classical context. In this section, we define the integral of a simple continuous function, show two different ways that it can be discretely approximated, and then explore how the discrepancy of the point distribution used for one of these approximations can be measured. Firstly, we consider the integral of some function $f : [0,1)^s \rightarrow \mathbb{R}$:

$$I(f) = \int_{[0,1)^s} f(x)\, d\mathbf{x} \tag{2.1}$$

where s is the dimension of the domain of f. We can simply numerically approximate this integral value using a discrete grid of points,

$$I(f) \approx \sum_{n_1=0}^{m} \cdots \sum_{n_s=0}^{m} \frac{1}{m^s} f\left(\frac{n_1}{m}, \ldots, \frac{n_s}{m}\right) \tag{2.2}$$

using a uniform grid spacing. There are various problems with this approach, such as poor-handling of highly oscillatory functions [29]—if the grid is spaced similarly to the oscillation frequency, a large amount of the signal can be missed—also, the number of points required to grid-sample the domain scales exponentially as the number of dimensions increases. For example, if we sample N evenly spaced points along a line in 1D, to achieve the same rate of sampling in sD, we would require N^s samples. The approximation error bound for an integral evaluation, or discrepancy of this method is of order $O(N^{-2/s})$ [29], and thus scales very poorly as the dimension, s increases. Whilst not hugely limiting on 2-manifolds, this problem is often referred to as *the curse of dimensionality* [29], and makes this technique (or similar) impractical for the evaluation of high-dimensional integrals.

The most common solution to this problem is the Monte Carlo method [55]. Using the Monte Carlo method, we can approximate the integral using:

$$M_P(f) = \frac{\sum_{x \in P} f(x)}{|P|} \tag{2.3}$$

where P is a set of points in $[0,1)^s$ chosen according to a uniform probability distribution, and $|P|$ is the cardinality of P. Because $M_P(f)$ is so dependent on the set P, the properties of P and its size $|P|$ have a large effect on the accuracy. So the study of the quality of a particular sampling P becomes very important. Whilst the classical Monte Carlo technique employs a random sampling P, we refer to the Monte Carlo method as a point based integral approximation method, which can then employ a variety of different point distributions. As mentioned, when the discrepancy of a set P decreases, so does the approximation error for a Monte Carlo integral evaluation. Thus, the study of the discrepancy of P is important, and provides a quantitative measure for the quality of P. We can think of the discrepancy of a set as the difference between the actual value of the integrand, and its approximation. Or more exactly [57]:

$$|I(f) - M_P(f)| \leq D(P)\text{var}(f) \tag{2.4}$$

where $D(P)$ is the discrepancy of the set, and $\text{var}(f)$ the variance of the integrand, defined as:

$$\sigma^2 = \int_{[0,1)^s} f(x)\,(I(f) - M_P(f))^2 d\lambda \tag{2.5}$$

where λ is the Lebesgue measure of a set. Discrepancy is often thought of as the Monte Carlo approximation error as defined by Eq. 2.4, but can also be loosely considered as a sequence's deviation from a uniform sampling of a domain [92]. As demonstrated in Eq. 2.4, by lowering the discrepancy $D(P)$ of P, the approximation error, $|I(f) - M_P(f)|$, is reduced. So by ensuring low discrepancy, we can minimise the difference between the continuous and approximated integral.

There are various techniques to measure the discrepancy of a sample set, though the star discrepancy, D^*, is the most widely used [92]. We now introduce this star discrepancy. Let $\mathcal{B}(w)$ with $w \in [0,1)^s$ be an s-dimensional box with one point at the origin,

$$\mathcal{B}(w) = [0, w_0) \times \cdots \times [0, w_{s-1}). \tag{2.6}$$

We then define the cumulative distribution function $c(P,w)$ associated with the sample set P as

$$c(P,w) = \frac{|P \cap \mathcal{B}(w)|}{|P|}, \text{ for } w \in \mathbb{R}^s \tag{2.7}$$

where $|\cdot|$ is the cardinality of a set. We then define the s-dimensional volume $\text{vol}(\mathcal{B}(w))$ in Euclidean space as:

$$\text{vol}(\mathcal{B}(w)) = \int_{\mathcal{B}} 1dV = \prod_{i=0}^{s-1} \int_{[0,w_i)} 1dx = \prod_{i=0}^{s-1} w_i \tag{2.8}$$

Finally, we define the star discrepancy as the supremum of the magnitude of the difference between the cumulative distribution function and the volume:

$$D^*(P) = \sup_{w \in [0,1]^s} |c(P,w) - \text{vol}(\mathcal{B}(w))| \tag{2.9}$$

Thus, $D^*(P)$ is the supremum of the difference between the exact value for a particular subset and its approximated measure, over all axis aligned hypercubes in $[0,1)^s$ with one corner at the origin. A low-discrepancy point set has a low discrepancy for a fixed $|P|$. So we define D_N^* as the optimally achievable value of $D^*(P)$ for all P with $|P| = N$, i.e. $D_N^* = \inf_{P \subset [0,1)^s; |P|=N} D^*(P)$. To assess the quality of a particular point distribution generation method we are interested in the behaviour of the function $D^*(P)$ with respect to increasing $|P|$ as this determines the convergence properties of integral approximation. Hence, we study the asymptotic behaviour of this curve using the Big Oh (Bachmann-Landau) notation, now simply referred to as its order.

Given a Monte Carlo approach, using a uniform random sequence, the discrepancy is of order $O(N^{-1/2})$ [92], and not dependent on the dimension s. Anything smaller than this will improve the convergence behaviour of the Monte Carlo method. Achieving low-discrepancy is difficult, but improving on a random distribution in general is desirable. The term low-discrepancy is often used vaguely to simply indicate an improved rather than optimal discrepancy behaviour. A bound for the discrepancy of any N-element point set has been proven [110]: for $s = 1, 2$, the star discrepancy satisfies the sharp inequality

$$D_N^* \geq B_s N^{-1} (\log N)^{s-1} \tag{2.10}$$

where the leading term $B_s > 0$, and is only dependent on s. This is also believed to be true for $s > 2$, but has not been proven. Thus we know the order of D_N^* which is the optimal star discrepancy achievable. The order of the star discrepancy $D^*(P)$ for particular point sets is discussed in the following section. Another measure, the mean square discrepancy, or L^2 discrepancy [132], is the mean square distance between the sample points and a grid set of points. Shirley [115] demonstrates the use of this measure, but results are not significantly different than those using the D^* measure.

2.1.2 Low-discrepancy Distributions

In this section, we overview the construction methods for the Hammersley and Halton sequences, the Niederreiter and Sobol sequences, and a jittered sampling method. The theoretical discrepancy bounds of these sequences are considered here, and experimentally assessed in Section 5.1. The Niederreiter and Sobol sequences are believed to be optimal [92] for axis-aligned rectangular subsets, and use a lattice structure to enforce point distribution uniformity 2.1.2. Use of them is avoided for other situations, where sampling with non-axis-aligned shapes is required [39]. The Hammersley and Halton sequences use the van der Corput sequence [33] as a construction method, and are commonly used in the literature. Finally, unlike the other distributions, the jittered approach has a probabilistic construction, and the advantages and disadvantages of this are discussed 2.1.2.

Hammersley and Halton Sequences

The Hammersley [54] and Halton [50] sampling methods are both low-discrepancy sequences, achieving the lowest possible order of magnitude of discrepancy [92]. This property makes them useful as a basis for comparison with our approach to low-discrepancy sampling; in Section 5.1.3 we compare results from these distributions on the plane and the sphere to our work. The van der Corput sequence, ψ_b, is a method to partition or sample the unit interval by maximising the distance between sample points, resulting in a uniform distribution on that interval. The premise is that a positive integer k can be expanded in a base b, expressed as a unique sequence of digits $k = a_0 + a_1 b + \cdots + a_{r-1} b^{r-1}$. Using the radical inverse ψ_b of this expansion, the positive integer k is converted to a floating point number in $[0, 1)$ by reflecting the digits around the decimal point. The k-th element of the sequence can be calculated using:

$$\psi_b(k) = \sum_{i=1}^{r} \frac{a_i}{b^i} \tag{2.11}$$

This defines a Hammersley point $p = (k/N, \psi_{b_1}(k), \ldots \psi_{b_s}(k))$, where s is the dimension, and where b_i are chosen pairwise co-prime. Both the Hammersley and Halton sequences are deterministic, but because the first co-ordinate, k/N, depends on the size of the point set N, changing N will change the output distribution. By adding a single point, the entire distribution is altered, and thus N cannot be increased incrementally: it must be defined prior to construction.

Replacing the first co-ordinate k/N with a new van der Corput sequence with a different prime, b', as the first co-ordinate, results in the Halton sequence [125]. Because the first

co-ordinate is no-longer dependent on N, the Halton sequence allows for an incremental construction.

These two sequences are often used to signify the lower-bound in the classification of low-discrepancy sequences, in terms of their scaling with respect to N [92]. The leading term, B_s of Eq. 2.10, is used to differentiate between such low-discrepancy sequences. For the Hammersley and Halton sequences, the minimum discrepancy is achieved using:

$$B_s = B(b_1, \ldots, b_s) \tag{2.12}$$

where $b_1, \ldots, b_s$ are the first s primes, for $N \geq 2$ [92]. Niederreiter [92], however, shows that, the value of B_s for the Halton and Hammersley sequences grows super-exponentially with s. Thus, the bound in Eq. 2.10 with the value of B_s shown in Eq. 2.12 becomes far less useful as s grows. However, work such as [26], demonstrates specific prime number choices that can hugely increase the quality of the Halton sequence in such higher dimensions.

Niederreiter and Sobol Sequences

Let P be a set of points in $[0,1]^s$, where $|P| = b^m$, for some integer base b. If, for every subset interval J, $|P \subset [0,1)^s| = b^t$, then P is a (t,m,s)-net, where $m, t \in \mathbb{Z}$, and t is a quality parameter (usually small for low-discrepancy sets [69]), and $0 \leq t \leq m$. This is defined by Niederreiter [92], and is simplified greatly in [21]. The subset intervals J are defined as

$$J = \prod_{i=1}^{s} \left(\frac{a_i}{b^{d_i}}, \frac{a_i + 1}{b^{d_i}} \right) \tag{2.13}$$

for integers $d_i \geq 0$ and $0 \leq a_i \leq b^{d_i}$ for $1 \leq i \leq s$ and $\lambda(J) = b^{t-m}$, where $\lambda(J)$ is the Lebesgue measure of J. Note that whilst in practice, $|P|$ may be any integer value, an exact division of points among the subset intervals is only achieved when $|P| = b^m$. Construction methods for (t,m,s)-nets may differ, so long as each J contains the right number of points. This construction approach results in a lattice, enforcing a geometrically even distribution. The Niederreiter and Sobol sequences are both examples of (t,m,s)-nets, which are sets of equidistributed points that satisfy Eq. 2.10 [91]. The value for the leading term B_s also scales far better in practice than that of the Hammersley and Halton sequences [92].

We point the reader towards [93] and [116], for a detailed description of how to generate the Niederreiter and Sobol sequences; such detail is beyond this work as we only use these sequences as a comparison to our own work. As pointed out by [22], in dimensions

$s \leq 7$ the Sobol sequence maintains a discrepancy similar to the Niederreiter sequence. In higher dimensions, however, this is not the case, where it performs worse. In this work, both the Niederreiter and Sobol sequences are used as comparisons to our results to demonstrate the best-case results for the discrepancy of axis-aligned rectangles in low dimensions.

Jittered Sampling

Monte Carlo methods use a uniform random sampling, which provides a solution to the curse of dimensionality (see Section 2.1.1), but suffers from the problem of having a high variance σ^2, and no correlated uniformity between the samples in the sequence can be guaranteed.

Niederreiter [92] shows that for a random point distribution P, the absolute error of $M(f)$ (see Eq.2.3) is $\sqrt{\sigma^2(f)}N^{-1/2}$, with variance σ^2 given by Eq. 2.5. In other words, the mean square error of random sampling is of the order σ^2/N, which demonstrates the relation between the sampling's variance to its discrepancy bound.

One method to reduce the sampling's variance, and thus improve the error bound is jittering points inside strata. Stratified, or jittered, sampling is a reduced variance probabilistic sampling technique. $[0,1]^s$ is divided into k regular sub-sets S_i, $i = 1, \ldots, k$ partitioning the space, then within these strata S_i, N_i points are placed at random, possibly with a weighting towards the centre of the strata, resulting in a numerical integral estimate [92] of:

$$\int_{[0,1)^s} f dV \approx \sum_{i=1}^{k} \frac{\lambda(S_i)}{N_i} \sum_{n=1}^{N_i} f(x_n^i) \tag{2.14}$$

where x_n^i is a random sample within S_i, and λ is the Lebesgue measure of the set, which must be simple to compute. Often, $N_i = 1$, although regardless of this fact, to achieve a uniform density of samples, N_i must be constant. This sampling results in a mean-square error that satisfies the inequality [92]

$$\sum_{i=1}^{k} \frac{\lambda(S_i)}{N_i} \int_{S_i} \left(f - \frac{1}{\lambda(S_i)} \int_{S_i} f d\lambda \right)^2 d\lambda \leq \frac{\sigma^2(f)}{N}. \tag{2.15}$$

This shows that the variance of stratified sampling will never be worse than random sampling. Beck and Chen [11] provide an error bound for stratified sampling, where $N_i = 1$, for any rotated subset, showing a discrepancy of $O(N^{1/4}\sqrt{\log(N)})$.

It is important to note the distinction between this measure and those given for the sampling methods discussed earlier in this section (Niederreiter, Sobol, Hammersley and Halton sequences): for those techniques, the subset used to describe the star discrepancy

bound is an axis-aligned rectangle, whereas the jittered bound is defined for any rotated rectangular subset (i.e. not just axis-aligned rectangles). When an axis-aligned rectangular box $\mathcal{B}(w)$ is used as the sampling subset of $[0,1]^s$ for star-discrepancy calculations, both in theory and experimentally (see Section 5.1.3), lattice-based sampling methods such as the Niederreiter sequence perform better than jittered sampling. However, as pointed out by [39], the discrepancy of such techniques becomes far worse when measured using non-axis-aligned rectangles or arbitrary shapes. Thus, for applications such as super-sampling in computer graphics, where non-axis aligned shapes and lines must be sampled, such techniques are less useful. In fact, results in [39] showed that jittered sampling performs second to the deterministic low-discrepancy method (described in [124]) for axis-aligned rectangles, and produced the best results in a measure of the discrepancy of subsets defined by arbitrary edges in $[0,1]^2$, performing considerably better than the Poisson disc and dart throwing algorithms tested. In Section 5.1.3, we show a similar outcome for generalised star discrepancy measures, demonstrating considerably worse results for the deterministic low-discrepancy methods when a variety of rotated sample shapes are used, but very consistently good results for the jittered sampling approach.

Our approach generates space-filling curves, which are then sampled, reducing the complexity of the sampling problem, and producing high quality sample distributions (see Section 1.2). Steigleder and McCool [117] define the Hilbert Curve as a linear series of irregularly shaped, but area-constant strata (which are then sampled). Thus, instead of sampling regular, rectangular strata, the irregular strata defined by sections of the space-filling curve are sampled, the observation being that the technique produces the same output as jittered sampling (hence being referred to as generalised stratified sampling). It is worth noting that this parity is only true at the limit of the curve (see Section 2.2.1), as the discrete approximation of the curve only actually passes within a certain maximally-bounded distance from every location in the continuous domain. However, as shown in Section 5.1.3, this does not have a practical effect on the results in the experiments investigated. The uses and effects of applying space-filling curves in the generation of high quality sample distributions is discussed further in Section 2.2.

2.2 Space-filling Curves

Space-filling curves are a continuous mapping from the unit interval onto a higher dimensional domain, such as the unit square. Our method uses space-filling curves to reduce complex higher-dimensional sampling problems to a more simple problem of sampling a 1D interval. In this section we define and investigate the properties of space-filling curves.

Peano first defined space-filling curves in 1890, and shortly after, Hilbert described them geometrically (cited by [108]) along with visualisations considerably increasing their acceptance. Publications in a variety of fields make use of space-filling curves, ranging from their construction [25, 103], to applications taking advantage of their unique properties: their construction provides a linearisation of a domain and a mapping between a one-dimensional interval and higher dimensional spaces, allowing some problems to be reduced in their dimensional complexity. These two properties make them a popular approach for half-toning [121], spatial indexing (applied to the travelling salesman problem) [99], vertex caching [16], compression [15], and triangulation [107]. The Hilbert curve is one of the best known space-filling curves in the context of these problems, and provides near-optimal spatial coherency in 2D [44] (i.e. points close on the curve are close in the straight line distance in Euclidean space). In this section, we first define the general case and construction of a space-filling curve, followed by a discussion of the space-filling curves used in this work and applications of curves in the literature. Finally, we introduce the Hölder measure which considers the spatial-coherency of space-filling curves.

2.2.1 Definition

In this section, we provide the definition of space-filling curves. For a much more thorough analysis of space-filling curves and their construction, we direct the reader to the excellent survey of the field by Sagan [108].

Space-filling curves provide a continuous mapping from the unit interval, $[0,1]$, onto $[0,1]^s$, though we will focus on the common case of mapping to the unit square, $[0,1]^2$. Space-filling curves are are a surjective, self-intersecting, mapping between $[0,1]$ and $[0,1]^2$, and cannot be injective or, therefore, bijective. We now briefly define a space-filling curve: Let C be a bounded sub-space of $\mathbb{E}^s$, where $\mathbb{E}^s$ is the s-dimensional Euclidean space with $s \geq 2$, and let $f : [0,1] \xrightarrow{onto} C$ be continuous, where the $\xrightarrow{onto}$ notation represents a surjective mapping, and let $J_s(f([0,1])) > 0$ where J_s is the s-dimensional Jordan volume. Note that $f(A)$ where A is a set is defined as $f(A) = \cup_{x \in A} f(x)$. Then f is a space-filling curve in $\mathbb{E}^s$ [108].

Space-filling curve approximations often use a geometric construction (cited by [108]). Such constructions approximate the continuous curve, and converge, in the limit, to the actual space-filling curve. Such approximations, for example, as used for the Hilbert curve, are often in-fact self-avoiding. Whilst this property of self avoidance may be irrelevant for applications relying just on a spatial ordering of the discrete elements of the curve, it is an important property to ensure that points close along the length of the curve

are close in their straight line distance in $\mathbb{E}^s$ (i.e. the curve has good spatial coherence). For simplicity, we refer to both the approximation and its limit as space-filling curves in the following.

2.2.2 Common Space-Filling Curves

In this section, we describe two popular space-filling curves, the Hilbert and Peano curves. These curves are used for various reasons. Primarily, they fill the unit hypercube, rather than some other shape. They can also be defined by simple recursive constructs. Finally, they have, to varying degrees, good spatial coherence—points close in E^s are close along the path of the curve. Two other curves were initially considered, the zig-zag curve (see Figure 2.1) and the Z curve (see Figure 2.2). Both curves can be described with simple recursive constructs and fill the unit hypercube. However, both curves suffer from poor spatial coherency, and are thus not used in this work.

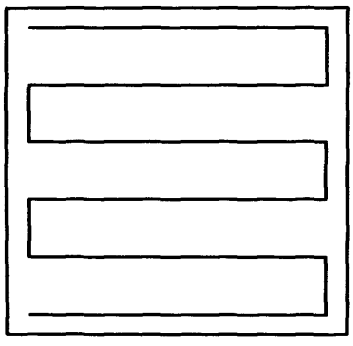


Figure 2.1: A zig-zag curve.

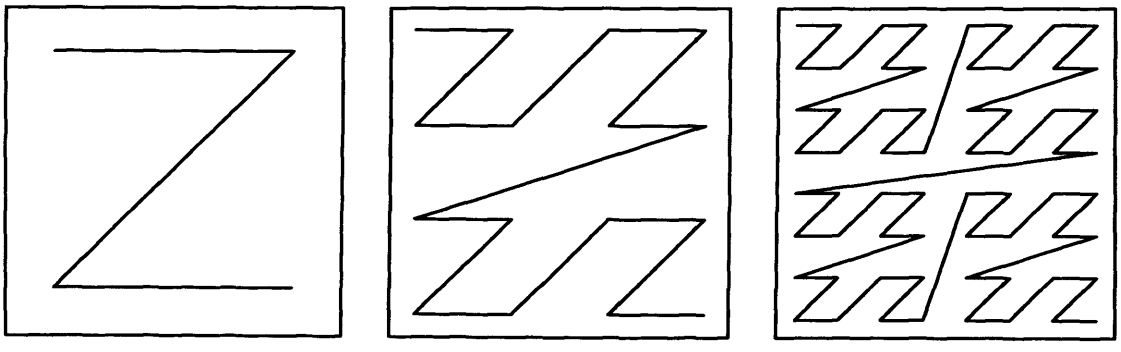


Figure 2.2: The first three stages of the Z curve.

Hilbert Curve

The Hilbert curve is by far the most widely used space-filling curve due to its construction simplicity and good spatial coherence [44], and most of the work detailed at the beginning of this section makes use of it. Hilbert discovered a space-filling curve which he described with a simple geometric construction (see Figure 2.3). The first order approximation of the Hilbert curve C_h, can be constructed by dividing $[0,1]^2$ into four congruent squares, with the curve following a self-avoiding path through the centre of each square, starting and ending in the centre of a square. These square centres we refer to as Hilbert curve vertices. Each higher order approximation is constructed by splitting each existing square into four, repeating the same process, and permuting the path through the new squares so that a contiguous curve is maintained resulting in 2^{2d} subsets where d is the approximation order. The first three stages of the Hilbert curve are shown in Figure 2.3. This converges in the limit to the Hilbert curve $C_h : [0,1] \xrightarrow{onto} [0,1]^2$. It has been demonstrated that the Hilbert curve has the best theoretical [44] and experimental [70] spatial coherency. That is, points close in $\mathbb{E}^2$ are close on C_h.

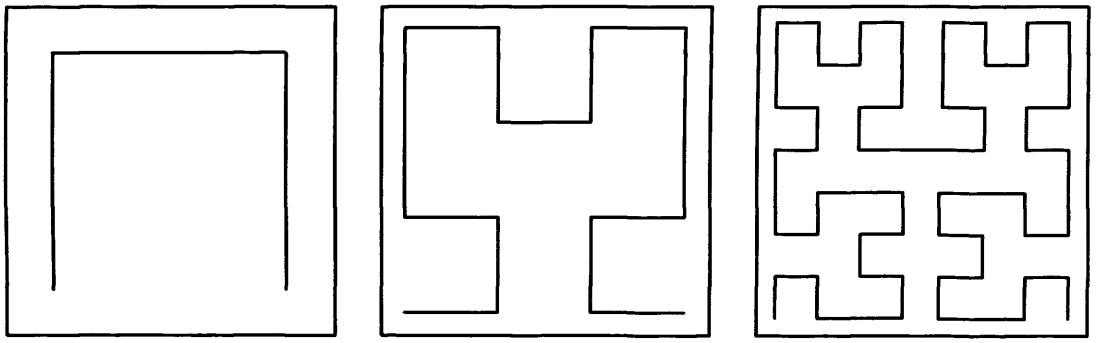


Figure 2.3: The first 3 stages of the Hilbert curve.

Many methods have been proposed to construct the Hilbert curve, including recursive approaches [23], direct computation of the desired approximation using bitwise operations [25, 117] and adaptive methods [103]. The bitwise approach described by Butz [25] provides the fastest computation for a standard Hilbert curve. However, using the adaptive approach (see Section 4.2.1) allows us to generate a Hilbert curve that is approximated to different depths locally. This is useful to compensate for parametric stretch when the curve is mapped to a surface using a non-uniform parameterisation (see Section 4.5), or if the sampling density of the curve is non-uniform.

Peano Curves

Giuseppe Peano was the first to define space-filling curves (cited by [108]), resulting in their alternative name, Peano curves. However, he did not describe them geometrically, but rather, as a series of digit permutations defined by a unique operator [108]. Since then, Peano's curve has been described geometrically with varying interpretations, though it has been proven that these produce exactly the same curve as Peano's original technique [108]. To simplify, we consider the geometric interpretation, focusing on two of the most common representations, referred to here as the Peano curve (see Figure 2.4) and the Peano II curve (see Figure 2.5).

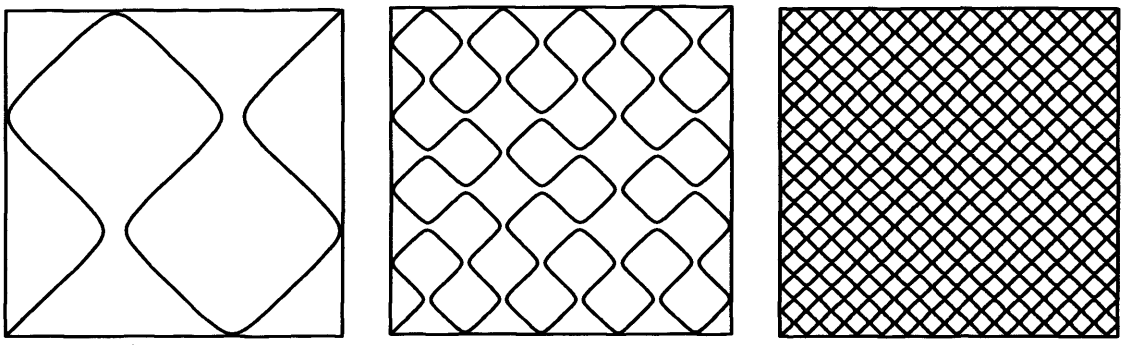


Figure 2.4: The first three stages of the Peano curve. Corners have been smoothed to clarify the curve path.

The general geometric interpretation of the Peano curve has been extrapolated in the literature, and is summarised in [108]. The construction of the Peano curve is in fact very similar to the Hilbert curve. Instead of dividing $[0,1]^2$ into four sub-squares, $[0,1]^2$ is divided into nine congruent squares. Sections are subdivided and permuted as with the Hilbert curve, with 3^{2d} subsets, resulting in a curve $C_p : [0,1] \xrightarrow{onto} [0,1]^2$. Alternative paths through these sub-squares have then been defined; the first three construction stages of the two popular representations, the Peano curve and the Peano II curve are shown in Figures 2.4 and 2.5. The Peano curve in various forms has also been used extensively for applications similar to the Hilbert curve, although many authors have applied the Hilbert curve due to its better spatial coherence. Note also that the Peano II curve visits the same vertices more than once, which is not ideal for our sampling application.

2.2.3 Spatial Coherency of Space-filling Curves

We primarily look at the Hilbert and Peano curves due to their good spatial coherence, and also their self avoiding-nature. Other curves, such as the Z curve [88], have been applied

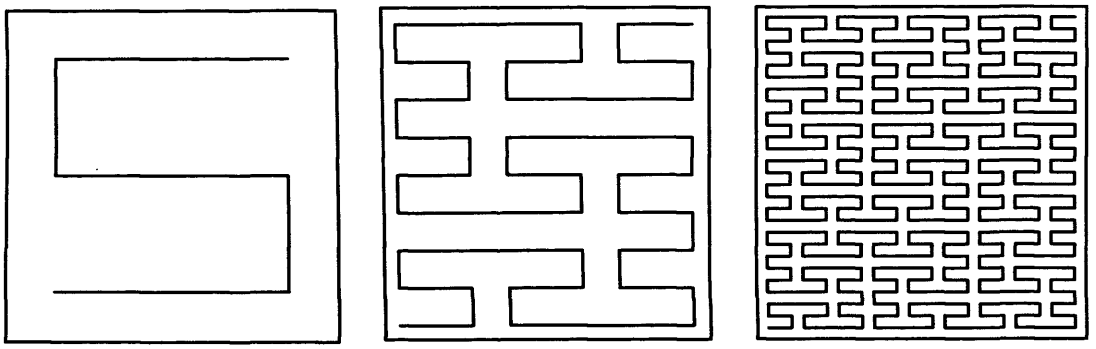


Figure 2.5: The first three stages of the Peano II curve.

to a variety of problems in a similar manner to the Hilbert curve. It has the advantage of being very simple to generate [62], though it has the disadvantage that points close on the curve are sometimes very distant in Euclidean space (as the curve makes a 'jump' between blocks; see Figure 2.2). The set of vertices making up the Z curve (see Figure 2.2) is arranged in a uniformly spaced grid, with a horizontal and vertical Euclidean distance of e between vertices, where e depends on the recursion level of the curve. However, the distance between every second vertex along the path of the curve is actually $\sqrt{2e}$, due to the fact that every second vertex is connected along the diagonal of the grid of points. This problem of spatial coherence is investigated by Faloutsos and Roseman [41], who experimentally demonstrate that the Hilbert curve has better spatial coherency, or 'clustering', than the Z curve. Another curve, the zig-zag, is generally used for tasks where spatial coherency does not matter, such as for robot path generation [9]. In this case, the curve is appropriate because it changes direction very infrequently which introduces less vibration to the robot, meaning that less time is required for the robot to pause and settle before performing its task at points along the curve. Whilst there are no 'jumps' in the curve, the curve's poor spatial coherency is undesirable. For these reasons, we now look more closely at the Hilbert and Peano curves.

To investigate the distances between points on the unit interval compared to the distances between them in the unit square under the mapping of a space-filling curve $\mathcal{C}$, we consider Hölder continuity. The principle is to compute the Hölder constant $c_{\mathcal{C}}$ that defines the upper bound of the difference between distances on $[0,1]$ and $[0,1]^s \in \mathbb{E}^s$. For a space-filling curve, the mapping $\mathcal{C} : [0,1] \xrightarrow{onto} [0,1]^2$ is Hölder continuous of order $1/k$ if, for all $s,t \in [0,1]$ [24]:

$$||\mathcal{C}(s) - \mathcal{C}(t)|| \leq c_{\mathcal{C}}|s-t|^{\frac{1}{k}} \tag{2.16}$$

For $k = 1$ this is the same as Lipschitz continuity. An exact value for the Hölder constant $c_{\mathcal{C}}$ for the Hilbert curve, $\mathcal{C}_h \in [0,1]^2$, of $\sqrt{6}$ is known [24]. Both Peano curves demonstrate a larger upper bound of $3\sqrt{5}$ for the Hölder constant. This shows that the Peano

curve maps distances in $[0, 1]$ onto less equal distances in $[0, 1]^2$ than the Hilbert curve, indicating a poorer spatial coherence for the Peano curve.

2.2.4 Sampling Space-filling Curves

Our sampling approach uses a space-filling curve to generate a set of points in the $[0, 1]^2$ parameter domain. When sampling points with space-filling curves (see Chapter 4), we build up a density histogram along the curve, and distribute the points according to this, essentially equalising the histogram. Each vertex that forms part of the Hilbert curve approximation is assigned a density, and due to the histogram equalisation, a vertex with a very high density would increase the chances of a point being placed not just at that vertex, but also at surrounding vertices. If the space-filling curve jumps large distances in Euclidean space (i.e. it has poor spatial coherence), a vertex with a high density before the jump may cause a local increase in samples after the jump, even if the density there is low. This would result in an incorrect distribution of sample points, and is the primary reason for not employing the Z curve (see Section 2.2.3). We effectively aim to approximate an integral over the density along the curve to estimate the local sampling density. The approximated integral is more likely to be more accurate for a given number of samples if those samples are evenly spaced.

2.3 Geometry of Curves and Surfaces

Differential geometry considers the differential and integral properties of shapes. In this section, we only consider the properties of smooth curves and surfaces in 3D Euclidean Space. For an introduction to this topic, see [82], and for further details, see [81]. In particular, we focus on the definition of the first and second fundamental forms of surfaces, and the geometric properties derived from them, such as tangent spaces, area, normals and curvature. These properties are later used to compute point distributions for our algorithm (see Section 4.3).

We now consider the analysis of curves on surfaces, passing through a given point, in order to define the first and second fundamental forms. The first fundamental form defines metric properties on a surface such as angles and distances, whilst the second fundamental form (with the first fundamental form) allows us to compute curvature properties. We consider these forms in Euclidean space with a Euclidean metric (and any sub-space gets the canonically induced metric from the Euclidean metric).

Let M be a surface with parameterisation $X(u,v) \in \mathbb{E}^3$ and let $c(t), t \in [a,b]$ be a curve on M. We can then write $c(t) = X(u(t), v(t))$ and

$$c'(t) = \frac{\partial X}{\partial u}\frac{du}{dt} + \frac{\partial X}{\partial v}\frac{dv}{dt} = u'X_u + v'X_v. \tag{2.17}$$

Let $s(t)$ be the arc-length along c with $s(a) = 0$, i.e. $s(t) = \int_a^t \|c'(s)\| ds$. Therefore,

$$\left(\frac{ds}{dt}\right)^2 = \|c'(t)\|^2 = (u')^2 E + 2u'v'F + (v')^2 G \tag{2.18}$$

with

$$E = \langle X_u, X_u \rangle,\ F = \langle X_u, X_v \rangle,\ G = \langle X_v, X_v \rangle \tag{2.19}$$

where $\langle \cdot, \cdot \rangle$ denotes the Euclidean scalar product. Then the first fundamental form can be expressed as

$$\left(\frac{ds}{dt}\right)^2 = E\left(\frac{du}{dt}\right)^2 + 2F\left(\frac{du}{dt}\frac{dv}{dt}\right) + G\left(\frac{dv}{dt}\right)^2. \tag{2.20}$$

To explain this, define a tangent space of a point p on M: Take all curves through p (differentiable at p). The set of all tangent vectors of these curves at p gives the tangent space T_pM, which is a 2D vector space with X_u and X_v (at p) as basis. Now $\mathbb{E}^3$ induces onto T_pM the scalar product given by the first fundamental form, which means the matrix of the first fundamental form $\mathcal{I}$ is

$$\begin{bmatrix} E & F \\ F & G \end{bmatrix}. \tag{2.21}$$

The area of a surface is defined via the first fundamental form as $\text{area}(M) = \iint_M dA$ with the area element

$$dA = \|X_u \times X_v\|\, du\, dv = \begin{vmatrix} E & F \\ F & G \end{vmatrix} du\, dv = (EG - F^2)\, du\, dv. \tag{2.22}$$

We now consider the second fundamental form, and how it can be used to compute the curvature at a point on M. Considering $c''(t)$ as the acceleration of a point along the curve, we can break this down into two components with respect to M: $c'' = c''_t + c''_n$ where c''_t is tangent and c''_n normal to M. Assume c is parameterised with respect to arc-length s, $c(s) = X(u(s), v(s))$. Then, with Eq. 2.17,

$$c'' = u''X_u + v''X_v + u'X'_u + v'X'_v \tag{2.23}$$

(where $'$ means d/ds), which expands to

$$c'' = u''X_u + v''X_v + u'u'X_{uu} + 2u'v'X_{uv} + v'v'X_{vv}. \tag{2.24}$$

To decompose this into a tangent and normal component, let $N = X_u \times X_v / \|X_u \times X_v\|$ be the unit normal of M. X_u and X_v are already tangent to the surface, but X_{uu}, X_{vv}, X_{uv} still need to be decomposed. Their decomposition yields the Gauss formulae:

$$X_{uu} = \Gamma^u_{uu} X_u + \Gamma^v_{uu} X_v + L_{uu} N, \tag{2.25}$$

$$X_{uv} = \Gamma^u_{uv} X_u + \Gamma^v_{uv} X_v + L_{uv} N, \tag{2.26}$$

$$X_{vv} = \Gamma^u_{vv} X_u + \Gamma^v_{vv} X_v + L_{vv} N. \tag{2.27}$$

The Γ are not necessarily similar for X_u, X_v as X_u and X_v are not orthonormal in general. However, we can compute the normal components L as projections:

$$L_{uu} = \langle X_{uu}, N \rangle,\ L_{uv} = L_{vu} = \langle X_{uv}, N \rangle,\ L_{vv} = \langle X_{vv}, N \rangle. \tag{2.28}$$

These give the matrix of the second fundamental form $\mathcal{II}$:

$$\begin{bmatrix} L_{uu} & L_{uv} \\ L_{uv} & L_{vv} \end{bmatrix}. \tag{2.29}$$

It describes the extrinsic geometry of M, i.e. how it is embedded into $\mathbb{E}^3$, and can be written as the differential form,

$$\mathcal{II} = L_{uu}\, du^2 + 2L_{uv}\, du\, dv + L_{vv}\, dv^2. \tag{2.30}$$

Given a curve c through a point $p = c(s)$ on M with normal $N(c(s))$ we consider the normal component of its curvature, called normal curvature, to study the curvature of M. By construction we have $\langle c''(s), N(c(s)) \rangle = \mathcal{II}(c'(s), c'(s))$. By differentiating the equation $\langle c'(s), N(c(s)) \rangle = 0$ with respect to s, we get

$$\langle c''(s), N(c(s)) \rangle = -\langle c'(s), \mathbf{D}N(c'(s)) \rangle = \mathcal{II}(c'(s), c'(s)). \tag{2.31}$$

This motivates the definition of the shape operator S as the negative derivative of the normal map $N : M \to \mathbb{S}^2$, i.e. $S = -\mathbf{D}N$, where $\mathbb{S}^2$ is the unit sphere. It also follows that S is self-adjont with respect to the first fundamental form as a scalar product, and so it has orthonormal eigenvectors e_1, e_2 with corresponding eigenvalues k_1, k_2. As e_1, e_2 are orthonormal we may write $c'(s) = e_1 \cos\theta + e_2 \sin\theta$. Here, θ is the angle between the curve's tangent in the tangent plane, and the eigenvector associated with k_1, and hence with Eq.2.31 we get Euler's Theorem

$$\langle c''(s), N(c(s)) \rangle = k_1 \cos^2\theta + k_2 \sin^2\theta. \tag{2.32}$$

We call the maximal and minimal normal curvatures arising from this formula the principal curvatures. They are given by the eigenvalues k_1, k_2 of S and have corresponding

principal curvature directions e_1, e_2. Conveniently in matrix representations these result from the eigenvalue analysis of the shape matrix

$$\mathcal{S} = \begin{bmatrix} L_{uu} & L_{uv} \\ L_{uv} & L_{vv} \end{bmatrix} \begin{bmatrix} E & F \\ F & G \end{bmatrix}^{-1}. \tag{2.33}$$

Furthermore, the Gaussian curvature is defined as $K = k_1 k_2$ and the mean curvature is $H = (k_1 + k_2)/2$.

2.4 Summary

This chapter has presented the main mathematical concepts applied in this work. Related work and applications are discussed in Chapter 7. We have introduced the concept of discrepancy and discussed its relevance to our work when sampling surfaces. We have defined the sampling methods used in this work, and explained how we use them to both construct and numerically assess sample distributions. We have also identified a need for a low-discrepancy sampling approach for arbitrary surfaces that performs well for a more general discrepancy measure. In Section 4.3 we detail our use of the point sequences discussed in this chapter to sample points on surfaces and in Section 5.1 we apply and expand the definition of the star discrepancy to numerically assess the quality of sample distributions. We have also defined and investigated a small selection of relevant space-filling curves. In Section 4.2, we discuss their construction in far greater detail, and in Section 5.4 we present experiments demonstrating the spatial coherence of the curves investigated. Finally, we have introduced the important differential geometry concepts used to calculate metric surface properties, such as area and curvature, that are applied in this work.

Chapter 3

Related Work on Sampling Applications in Visual Computing

This chapter investigates work relating to the applications of our sampling method. We consider the applications of point-based graphics (see Section 3.1), remeshing (see Section 3.2), and prototype painting (see Section 3.4).

We first consider point-based graphics, and whilst our emphasis is on surface sampling, it is also important to consider the field in general to understand the requirements for the sample distribution. We provide a brief survey of the field of point-based graphics (see Section 3.1.1). We then consider existing sampling methods (see Section 3.1.2). We also look at point primitive representation, and how a lack of explicit connectivity is solved in the literature (see Section 3.1.3). In our point-based graphics sampling work (see Section 7.1.3) we consider Levels of Detail (LoD) and View Dependent Rendering (VDR) to reduce scene complexity and improve rendering quality. In this section therefore, we consider existing popular LoD and VDR techniques (see Section 3.1.4).

We investigate the field of remeshing, considering its application in complexity reduction and local density control. We discuss standard mesh decimation techniques, and the relation to remeshing (see Section 3.2.1). In Section 3.2.2, we review existing remeshing techniques. We then consider mesh cutting and parameterisation, looking at when cutting is required, and how the parameterisation affects the sampling of the mesh (see Section 3.2.3), and discussing which methods we employ. Finally, we consider the triangulation of sample points, which is required once a point sampling has been generated on the surface (see Section 3.2.4).

In Section 3.3, we consider existing methods to evaluate the sampling produced for point and mesh based surfaces, and the approaches we use in our work. We then investigate the field of robotic prototype painting, looking at existing approaches and their limitations due to sampling issues (see Section 3.4). Finally, we summarise this chapter, highlighting how existing methods relate to our own work, and discuss our contributions to these fields (see Section 3.5).

3.1 Point-based Graphics

In this section, we investigate work related to point sampling in the field of point-based graphics. Point-based graphics is the name given to the approach of representing and rendering shapes using unconnected point primitives. Triangle meshes are the most commonly used rendering primitive, yet recent advances in technology make them less ideal: when the number of visible triangles in a scene becomes the same as the number of pixels in the display, then the average area of each triangle is reduced to only a single pixel, and thus triangles become an inefficient way to represent the scene. Points are a viable alternative to using triangle primitives, and have both advantages and disadvantages. Points are generally represented using discs (often referred to as surfels [98] or splats [135]) that are oriented to the surface normal. The disc gives them a measurable area and if the disc radius is controlled by surface curvature, makes them more suited to represent flat surfaces, as flat areas can be covered by fewer, larger discs. An alternative to splatting is direct rendering in image space [80], which uses a different approach to address the problem of rendering artefacts from hidden surfaces, and uses a multi-resolution filter to achieve a continuous appearance. Lacking an explicit topology and connectivity makes them useful for the representation of deformable or freeform surfaces [65], as the topology of a model may be completely changed, and no complex tessellation (such as a triangulation) has to be updated. Model acquisition techniques (such as laser scanners) natively sample with points, and allow for fine and organic models to be easily represented [98]; generating a mesh from these points is a time-consuming and error-prone process [98].

However, as meshes have been a popular method of surface representation for such a long time, a considerable number of models are available. To represent these models using a point-based method, simply removing the connectivity and increasing the sample density may result in a poor distribution of samples. Thus, resampling existing meshes with high-quality point samples is an important area of research.

An overview of point-based graphics is given in [4]. This includes work on acquisition, surface representation, rendering (including LoD and VDR), neighbourhood computation, surface simplification (especially iterative methods and particle simulation) and error metrics such as the Hausdorff distance. Kobbelt and Botsch [65] give a broad review of point-based techniques, with a concise introduction to different rendering techniques and LoD methods. They also look at the approaches for purely point-based and surfel based representations, briefly considering the sampling requirements for these methods. Both surveys also highlight the need for fast, robust, k-nearest neighbour operations for point-based approaches to provide fast neighbour look-up. A spatial-partionining datastucture is recommend as a solution to this neighbourhood problem [65]. Zwicker [133]

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 [7], and non-spectral properties, namely discrepancy.

3.1.1 Sampling Requirements

Usually, points sampled for a specific application require certain characteristics. When rendering surfels, an important characteristic is that samples are generally equidistant (relative to sampling density) from each other in all directions. Achieving equidistance from neighbours in this way results in a uniform amount of overlap of the surfels (for a constant surfel size assuming a uniform density), ensuring a complete coverage of the surface. If points are not regularly spaced in this way, a higher density will be required to ensure complete coverage. The radius of the surfels could be increased up to a certain point, but detail may be lost. However, as the distances between points becomes more regular, the distribution becomes more and more grid-like, which may cause sampling artifacts. As discussed in Section 1.1.2, low-discrepancy distributions have been employed to avoid such artifacts. However, the sampling pattern for such distributions also matters: Dobkin et al [39], demonstrate that for distributed ray-tracing, sampling patterns that have a very low discrepancy when measured with rectangular subsets (such as the Niederreiter sequence [92]) generally result in rendering artifacts. We investigate this property of surface coverage in Section 7.1.2, with results demonstrating much better coverage with our novel jittered sampling method (see Section 4.1) when compared to the other deterministic low-discrepancy distributions (see Section 2.1.2). Finally, for point-based graphics it is also important to be able to control the sampling density, for example, to make it proportional to the mean curvature; the inverse of which is generally used to control the radius of the splats, so that smaller surfels reside in areas of higher curvature with higher sampling density. By controlling the density in this way, many points can be used to represent complex areas of the surface, and fewer points to represent less complex, or flat, areas.

3.1.2 Sampling Approaches

Wu et al [128] suggest a point sampling technique generalised for LoD splatting [129], employing a progressive error-based decimation of the original set of input points, used to resample models, whilst 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 (tens of thousands of triangles). 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.

Proença et al. [100] demonstrate a method of point sampling implicit surfaces. They first define the surface as a set of multi-level partition of unity impicits [95] (MPU implicits), which also yields an octree spatial partitioning. The octree is defined and created according to surface curvature. In each octree cell on the surface, they then place a constant number of particles, achieving a good initialisation for the following particle relaxation process. Their sampling algorithm is faster than other surface relaxation methods, mainly due to the accuracy of the initial particle placement because of the octree structure. The point set they generate has a very uniform structure, prevalent in most relaxation approaches. However, the most significant problem with this approach is the presence of artifacts in the output; large white gaps are clearly visible in areas of high curvature.

Rovira et al. [106] describe how to sample an input mesh by generating a set of uniformly distributed lines within a bounding box, computed from pairs of sample points generated on tangent planes of the bounding box. The lines are intersected with the model, and a set of point samples is generated at the points of intersection on the model's surface. 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 are 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 also superior to common low-discrepancy distributions for point-based graphics applications (see Section 7.1.2).

Kalaiah and Varshney [61] describe a method for rendering and storing points with differential point geometry; each sample is stored as a 'differential point', providing discrete differential surface properties such as principal curvatures and the surface normal. Whilst computing such properties for surface rendering is not a novel idea in principal, it focuses on a reduction of points by describing and rendering each point with a lot of information, providing a compact approach to representing point-based models. The focus of this paper is not on the sampling quality, but rather on making the best use of a minimal number of samples to render the surface. As described in Section 4.3, we utilise this point rep-

resentation model to compute surface integrals and to provide high quality rendering of surfaces.

3.1.3 Neighbourhood Computation

Point-based representations do not store connectivity between the discrete points, and thus there is no explicit topological information about the underlying manifold. However, for many computations, such as rendering (e.g. not rendering hidden surfaces), LoD, deformation and boolean operations, knowledge of the neighbourhood for a point is necessary [65]. Thus, various techniques have been suggested to address this problem of neighbourhood computation. For isotropic, i.e. direction invariant, density distributions, computing the k-nearest neighbours using a spatial data-structure provides a solution of linear time complexity with respect to the number of samples [4] (after pre-computation of the data-structure). Pointshop3D [134] is a point based surface editing package that employs spatial data structures to facilitate much of the basic editing functionality provided, such as boolean operations and freeform deformation. Both kD-trees and dynamic grids [56] are used in Pointshop3D; kD-trees provide the fastest neighbourhood look-up, whilst dynamic grids are used when new points need to be inserted. Bhakar et al. [13] use an octree data structure, which partitions the data set, allowing for occlusion-based view-dependent computation. In our work, we utilise a quad-tree data-structure to generate the space-filling curve (see Section 4.2.1), which is used to compute k-nearest neighbours efficiently. Whilst this solution computes the nearest neighbours on the surface in $\mathbb{R}^3$, if less accuracy is required, adjacent points along the curve can simply be used directly as neighbours. All of the approaches described here rely on a point set having an isotropic density. To compute the correct neighbourhood for a set of points with an anisotropic density, however, other methods must be used, for example, tangent plane Delaunay triangulation [4].

3.1.4 Levels of Detail

Levels of Detail (LoD) are employed within computer graphics to provide a solution to the problem of overly complicated (and thus slow to render) scenes by reducing the geometric complexity of objects as their distance from the camera increases. LoD are important in various applications, including real-time rendering and visualisation, and progressive display and transmission of models. Luebke et al [77] give a thorough introduction of this field, with described techniques falling into three categories:

- Discrete LoD, where a set of objects of varying complexity are pre-computed (multi-resolution objects), and used to describe an object depending on the distance from the camera [30]. Kobbelt et al. [66] describe an interesting discrete technique that reduces the amount of data needed to be stored by implicitly defining the geometric differences between detail levels instead of maintaining a global hierarchy. Due to meshes being dependent on connectivity, the paper focuses on dynamic mesh connectivity, modifying edge lengths, vertex valencies and removing thin triangles;
- Continuous LoD, where a data structure allows for the level of detail of an object to be changed at runtime, with varying degrees of granularity;
- View dependent LoD, whereby the complexity of a geometric object is dependent on its position with respect to the camera, not just its distance from it. This last approach also allows for geometric silhouette enhancement.

Point-based graphics represent an interesting approach towards LoD, as the process is simplified greatly by the lack of a need to maintain a persistent connectivity between samples, and is thus one of its advantages over mesh based approaches (removing a point, unlike for meshes, does not require topology modification). In [65], LoD is discussed, and several data structures for rendering are considered. In Section 7.1.3, we demonstrate the speed at which we can resample a surface; once pre-processing is complete, the surface can be resampled in real-time as the distance between the model and the camera changes. We can represent an object with an arbitrarily fine-grained level of detail after initial pre-processing. This results in a continuous and view dependent LoD method that allows for real-time rendering.

Bhakar et al. [13] describe a stochastically sampled octree approach to generate a view-dependent LoD point sampling of a two-manifold, but no timing information is given and there is no evidence of any analysis of distribution quality. Similar to this technique, 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 from the camera, and then sample this part at the desired density. This is also useful in situations where the application requires a fixed scene complexity regardless of viewing angle.

Our approach can also increase the sampling density around silhouettes in real-time (see Section 7.1.3). This is useful, for example, in real-time graphics, where a high quality texture may mostly hide the lack of detail of a low-complexity model, but the low-complexity is still highly visible along the silhouette. In Section 7.1.3, we discuss a modification to the algorithm, utilising the quad-tree structure of the Hilbert curve to accelerate the process of our continuous, view dependent LoD approach.

Popping, is the name given to visual artifacts introduced by a change in geometry between two levels of detail [77]. A situation where this is most obvious is along the silhouette of the object. Continuous LoD approaches reduce this problem as there is no significant 'jump' in object complexity. Hoppe [58] describes an approach to progressive, view dependent refinement of meshes, where the mesh is refined at run-time based on a set of criteria. One approach to solving this problem with LoD meshes is to maintain the tessellated topology [18], and ensure that each coarser mesh is built from a subset of vertices of the mesh with higher level of detail. However, view dependent LoD provides a better solution for hiding the popping effect by maintaining high complexity around the object's silhouette. Our approach achieves both continuous and view dependent LoD in order to minimise the popping effect. However, in our approach, lower levels of detail are not a subset of the higher levels. Instead, new, jittered, point sets are generated on the surface at run time (see Section 6.2.3). This jittering, however, is not only probabilistic, but points cannot be incrementally added. An evenly distributed set of points is deterministic, but a uniform distribution cannot be created using an incremental construction. By instead using an incremental, deterministic 1D construction, such as the Niederreiter sequence, we can ensure that lower levels of detail are a subset of higher levels, further reducing the popping effect. This is discussed in Section 7.1.3.

3.2 Remeshing

This section reviews and introduces related work in the field of remeshing. Remeshing is the process of sampling an existing mesh, and tesselating this sampling to produce a new mesh, generally with (more) desirable characteristics, such as more regular polygons, or a different, or non-uniform, density of vertices.

As an introduction to recent remeshing techniques, Alliez et al. [6] give a review of various requirements and approaches in the field. 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 [75] 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 so obvious that a triangulated set of such points has similar desirable characteristics. Least number of triangles may not always be the most important requirement—for example, in finite element methods, having a regular mesh of equally sized, almost equilateral triangles is more important.

In the following we first discuss conventional decimation techniques. We then investigate remeshing techniques with varying emphasis on desirable properties of the output mesh,

and feature-preserving remeshing techniques. We also consider related mesh parameterisation and triangulation techniques.

3.2.1 Mesh Decimation

Decimation is the reduction of the number of polygons and vertices in a mesh, where the vertices of the output mesh are a subset of the vertices of the original mesh. Kobbelt et al. [67] describe a general framework for mesh decimation using greedy optimisation of a Hausdorff distance error metric. Other work by Wu and Kobbelt [127], provides a significant performance advantage over this earlier work by using a probabilistic multiple-choice approach for vertex selection and removal, while providing a similar surface approximation error. The vertex preservation property of decimation can be important in certain applications such as generating multi-resolution models, as previously discussed with regard to point-based rendering in Section 3.1.4.

Reducing the complexity of a mesh in this way may be a prime motivation for remeshing, although it may not be the only concern. Remeshing approaches, such as our approach, allow for a much greater flexibility and control of, e.g., sample density of the output, as the output vertices are not constrained to a subset of those in the original mesh. With our approach, we sample the mesh with a low-discrepancy vertex set, with the intention of capturing the original shape with the best accuracy possible given a predefined number of points.

3.2.2 Remeshing Techniques

Alliez et al. [5] describe a remeshing method that performs all computation on the surface without the need to compute a parameterisation, using intersections of principal curvature tensor fields to construct polygons that are aligned and shaped according to the curvature, thus creating 'anisotropic' polygons. Visual results are demonstrated, showing that remeshed surfaces which consider the local shape of the object are better than those produced using isotropic sampling. Timings in the region of a minute are typical, although mesh complexities are not explicitly listed. Although the quality of the output is high, control over point and triangle placement is somewhat limited due to the nature of the approach. Nevertheless, considering the curvature tensor field improves the output mesh, and also helps to preserve features [17].

Gu et al. [46] define an approach to the storing and rendering of objects as geometry images, using a computationally intensive parameterisation preprocessing stage (taking

about an hour). They have devised a remeshing technique which relies on this very good parameterisation, so that the parameter domain can be regularly sampled, and thus the geometry can be stored as a set of images. The use of geometry images produces very regularly remeshed surfaces, and inherently facilitates geometry compression. Our remeshing approach allows for much greater control over the sample distribution, requires far less preprocessing time, and allows the user to modify the sample density at interactive rates.

Alliez et al. [7] also describe a remeshing framework using a mesh parameterisation approach, that, after some pre-processing, allows for density controlled triangle distributions to be produced at interactive rates. Dithering and relaxation techniques are used to sample the surface regularly, and graphics hardware operations are used to perform these computations quickly. Results are good, though testing only considers the blue-noise criterion and visual assessment. Overall timing for the process is difficult to discern, but after the parameterisation step, a few seconds are needed for preprocessing, then an interactive rate is achieved for remeshing. Our technique allows a similar speed of interactive remeshing after a similar time for pre-processing, and provides experimental evidence of the high quality of the point sampling. Qu et al. [102] extend Alliez et al's [7] technique to remeshing based on surface patterns, such as textures and bump mapping. They reduce the mesh complexity intelligently such that no loss in quality is apparent. Voronoi tessellation and Lloyd relaxation are used to produce a sampling of the parameter domain, again resulting in a very regular mesh.

Surazhsky and Gotsman [119] describe a remeshing scheme with an emphasis on regularity and mesh connectivity, using an area based smoothing technique, and a dynamic patch-based parameterisation method—they do not attempt to parameterise the entire mesh, but work explicitly on patches of the mesh, allowing the technique to be easily applied to arbitrary genus models. They demonstrate low remeshing errors, very similar to results produced by our approach, with an overall remeshing time of 74 seconds for a complex, 50,000 vertex model. They also consider refinement of certain mesh areas before remeshing but do not consider the quality of point sample placement.

Feature preservation is also important in remeshing, especially with regard to engineering applications [123], where the preservation of sharp edges and other features is essential. Important information may otherwise be lost due to approximation of the original mesh by the remeshing technique. Vorsatz et al. [123] describe a method primarily aimed at addressing the problem of feature-loss due to aliasing. A technique is described whereby the surface is sampled, followed by a global relaxation and feature snapping method. Visual results are provided, demonstrating remeshed surfaces which preserve sharp edges. Though we do not specifically address the preservation of sharp, manufactured edges, we

approach the problem of feature loss using density functions, such as curvature. This approach means that the placement of a point at a very specific location cannot be guaranteed and so exact, sharp, edges are unlikely to be reproduced well. However, such features will still be sampled more densely, and features other than sharp edges will be preserved well.

3.2.3 Mesh Cutting and Parameterisation

Several remeshing approaches are based on cutting arbitrary-genus meshes into one or more topological discs, parameterising these patches, remeshing them, and then stitching them back together. Care has to be taken to stitch the mesh back together along the cuts to produce smooth boundaries, which has been largely solved [40, 49]. The approach described in [46] cuts the mesh into a single topological disc, and performs an iterative cut-parameterisation process, improving the cut at each step. Whilst the output of this approach is a stretch-minimising parameterisation, processing time is very high. We therefore apply the simple cutting approach used in [107] before parameterising the mesh, correcting for stretch using our adaptive sampling approach.

Our remeshing approach requires construction of a mesh parametrisation. We thus briefly consider some relevant approaches. An isometric parameterisation, i.e. a mapping that uniformly preserves all distances (thus also preserving angles), is ideal, but one that in practice is almost impossible to find. As a result, most approaches attempt to preserve either area (distances) or angles, referred to as *equiareal* and *conformal* respectively. An equiareal parameterisation results in area elements in the parameter domain mapping to similarly sized area elements on the mesh, but they are generally subjected to a large non-uniform stretch due to the lack of angle preservation.

One useful method is the technique described in [48], which minimises geometric stretch and iteratively optimises the parameterisation until no further improvement can be achieved. While this technique has a high computational cost, it ensures that the parameter domain is sampled uniformly. Floater et al. [42] describe a fast technique to generate shape-preserving (i.e. angle preserving, or conformal) parameterisations of triangle meshes. Yoshizawa et al. [131] give a fast and simple stretch-minimising technique based on using Floater's approach with progressive refinement steps. It provides quite good results with a run-time in the range of tens of seconds for models with about a hundred thousand vertices. In order to avoid artifacts that may result from such a non-uniform stretch, in our approach, we desire a conformal parameterisation, f, of the mesh M_s. The non-uniform areas can be handled by the adaptive space-filling curve produced.

Floater and Hormann [43] provide an excellent survey and tutorial of surface parameterisation, in which they define a conformal parameterisation as one that preserves all angles of intersection between arcs defined on the surface. A continuous function, $g \neq 0$, mapping a surface S to a parameterisation of this surface, S^*, is only conformal if the coefficients of the first fundamental form on S, $\mathcal{I}$ (see Equation 2.20), and on S^* with first fundamental form $\mathcal{I}^*$ are proportional,

$$\mathcal{I} = g^{-1}(u,v)\mathcal{I}^*. \tag{3.1}$$

We utilise Floater's [42] conformal parameterisation method, as we desire triangle shape preservation in order to avoid anisotropic stretch. Such unwanted anisotropic stretch would reduce the quality of the output point distribution. Because we can adaptively sample the parameterised mesh, a relatively large difference between the area of a triangle on the surface and a triangle in the parameter domain does not affect the final quality of our results (although it may have a minor impact on speed). Floater's technique is also very fast. If we were to alternatively use the technique described in [48], we could use a uniform (non-adaptive) Hilbert curve to sample the mesh, which would increase the speed of the pre-processing step of our algorithm, but at the expense of the total time taken including the parameterisation step; the adaptive curve generation is far faster than their parameterisation.

3.2.4 Triangulation

Triangulation is an essential step in remeshing; by performing a triangular tessellation of samples, a mesh can be produced. Bern and Eppstein [12] provide a survey of mesh generation for finite element methods. They consider the production of optimal triangulations with respect to specific parameters (minimum angle size and area), in the context of a fixed point set, and one with additional (Steiner) points. Shewchuk [113] demonstrates efficient implementations of popular Delaunay triangulation algorithms, demonstrating that an optimised divide-and-conquer approach [71] is the fastest ($O(n \log n)$ for n sample points) and most robust approach. It may be possible to construct a triangulation by basing the neighbourhood relations on the original triangulation of the input mesh. However, the step of triangulation in our remeshing algorithm is not our main concern, and thus we simply apply the algorithm of [71], treating the problem as the planar triangulation of an arbitrary point set in the parameter domain.

3.3 Point and Mesh Sampling Evaluation

We now briefly discuss methods to evaluate point resampling methods. There is no universally applicable method to assess the quality of a resampled surface, and the quality desired may be dependent on the application. Various assessment criteria are employed in the literature, e.g. discrepancy [106], minimum angle statistics [119], the blue-noise criterion [7]; the Hausdorff distance is used for measuring the accuracy of remeshed surfaces [28] and point sampled surfaces [129]. We compare the quality of meshes produced by our remeshing implementation to other competitive techniques by measuring the geometric distance between the input and output meshes using Metro [28], showing results competitive with one of the most popular remeshing techniques. We use Metro specifically due to its common usage in the literature, allowing comparison of results. Even visual assessment can be useful, especially if the final application is rendering. However, it is largely subjective.

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 [75]. An experimental technique, used by [106] to approximate the discrepancy of their point sets, involves sampling a unit square polygon using their method, divided into two triangles, and calculating the star discrepancy for a varying number of points. The technique demonstrates the discrepancy in this special case, but does not address the discrepancy of a more complex surface. In Section 6.2.1, 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 tested on parametric surfaces (see Section 5.1.4). This verifies that the mesh sample distributions exhibit low-discrepancy properties, i.e. they cover the surface area well yet do not suffer from aliasing problems associated with regular sampling. We also assess the coverage of points in the unit square, comparing low-discrepancy and random distributions to our technique.

3.4 Robotic Painting

Continuing reduction in product development lead-time has caused a huge increase in the development of rapid prototyping and manufacturing processes, also resulting in a greater focus on secondary processes, such as painting, finishing and assembly [118]. Whether for a one-off prototype, or for a production batch, hand painting of objects, such as toys or ornaments, is a labour intensive and expensive task which introduces a large, manual

bottleneck into the rapid manufacturing process. Bi and Lang [14] note the dependence on manual painting and provide motivation for robotic automation, citing as an example the ship building industry, which employs a large proportion of its workforce in this task.

Sung et al. [118] provide a solution to the problem of robotic prototype painting, using a tri-colour laser to expose points on photosensitive paint. The laser unit is mounted on a four-axis robot with an additional two-axis rotary tilt-table (incorporating the object holder). The system has produced grey scale images up to 300dpi from textured 3D meshes. Robot end effector positions and laser intensity values are then generated from the mesh by extracting surface data (such as the position, normal vector, colour) for individual sample points on the models surface. This data is used to generate a path for the robot (and the laser it carries), and so paint the image (i.e. expose the photosensitive paint). The painting process requires no physical contact, so potentially images could be exposed on any surface as long as there is no occlusion [60].

The above work uses a laminar slicing of the input mesh, producing a fixed number of samples around the perimeter of the models. However, this approach results in a raster scanned image on the model, the regularity of which can be easily detected by the human eye (see Section 7.3.2). A surface orthogonal to its slicing plane can be uniformly sampled using this raster scan method. However, if the surface is not orthogonal to the slicing plane, the resulting sampling density will not be even. Unless the surface very closely resembles a cylinder, this can produce artifacts in the resulting image (see Section 7.3.2). To overcome these sampling issues, in Section 7.3, we apply our mesh sampling approach to this problem. We demonstrate high quality surface samplings that are equi-distributed on the object surface, with the ability to produce dithered output. This application provides a very practical example of our sampling approach, demonstrating considerable advantages over the original approach.

3.5 Summary

In this chapter, we have reviewed the literature for three applications of our sampling approach considered in Chapter 7. We introduced the field of point-based graphics, and the requirements involved for producing a high quality point sampling for rendering. Existing sampling approaches were then discussed, covering quality issues, and solutions to rendering artifacts and coverage. We discussed the importance of fast neighbourhood computation in point-based graphics, and provided an overview of how our space-filling curve sampling approach addresses this problem. We then looked at levels-of-detail techniques,

the advantages which point-based methods have in this field, and how our approach provides a real-time fine grain solution. We then gave an overview of remeshing, followed by an analysis of standard mesh decimation techniques. We considered current remeshing approaches, followed by a look at the importance of feature preservation, discussing how we address this problem using density functions. We further discussed cutting, parameterisation and triangulation methods and their evaluation, considering sample distribution and surface approximation. Finally, we considered prototype painting, existing techniques and how our sampling method solves the existing problems of poor sample distribution and visible artifacts.

Chapter 4

Sampling with Space-filling Curves

This chapter describes our novel main sampling algorithm. Our approach provides a solution to the problem of generating high quality density-controlled point distributions on surfaces in $\mathbb{R}^3$. The definition of quality varies between applications (see Chapter 7, where we discuss specific qualities required for point-based graphics and remeshing). However, for the general case of sampling, it has been demonstrated that the property of low-discrepancy is broadly advantageous [115] (see Section 2.1). Thus, to ensure we get a uniform distribution, we require point distributions on surfaces with a low discrepancy. We may also require that the local point density of the surface distributions can be controlled, so that, for example, more points can be sampled in areas of greater detail or higher surface curvature. Both criteria describe high quality point distributions for many applications. The problem of sampling 2D arbitrary surfaces with a density-controlled low-discrepancy distribution is not trivial. Our solution to this problem is to decrease the complexity of the sampling problem by distributing points along a space-filling curve that has been mapped to the surface. We can then sample the curve, which reduces the sampling problem to a simpler 1D problem. We sample points on the curve in a manner similar to histogram equalisation [59].

In this chapter, we first provide an overview of our sampling algorithm (see Section 4.1). We introduce space-filling curve generation, describing the space-filling curves considered and our adaptive Hilbert curve algorithm (see Section 4.2). We then describe our approach to the computation of integrals for space-filling curve sampling (see Section 4.3), followed by a description of the 1D sequences investigated as a basis for this sampling (see Section 4.4). We consider approaches to the problem of stretch introduced by a parameterisation, and evaluate the solutions of adaptive curve generation and surface reparameterisation (see Section 4.5). Finally, we summarise this chapter (see Section 4.6). Our approach, and the accompanying results, were originally published in [103].

4.1 Algorithm Overview

We first briefly overview our approach. Our algorithm takes a parametric surface, X, which is assumed to be a regular, two-dimensional Riemannian manifold in 3D Euclidean space, a user-defined density function $\delta : X \rightarrow \mathbb{R}_0^+$, and the number, N, of required sample points as input, and computes a set of N points sampling X. δ is a user-defined, non-negative function indicating the desired sampling density for the construction of the new distribution; δ may be constant, or proportional to surface curvature, for example. This is a relative density, specifying the relative number of sample points required per unit area with respect to the overall number of points N. For simplicity we assume that X is parameterised over the unit square $[0,1]^2$ by $f : [0,1]^2 \rightarrow X$, but note that it can easily be changed to a more general parameter domain. We sample X using a space-filling curve: we generate an adaptive space-filling curve in $[0,1]^2$ which is mapped onto X by f. The space-filling curve is created adaptively to different recursion depths in the parameter domain to account for the stretch between the area element in the parameter domain and on the surface such that the surface is covered by the adaptive curve according to the density δ. Then sample points $\{p_l : l = 1,\ldots,N\}$ are placed along the curve on X based on how the integral $\int \delta dA$ increases along the curve, using an approach similar to histogram equalisation. While this is an integral along a curve, we interpret it as a surface integral over the area 'belonging to' the section of the adaptive approximation to the space-filling at the given level of refinement. This is justified as the adaptive curve approximates the space-filling curve that covers the whole surface in the limit, and in practice, lines of the curve are very close in the approximation (see Section 4.3). As demonstrated in detail in Chapter 5, this results in a low-discrepancy point distribution in the sense that for each subset U of X (in particular subsets of the type used for discrepancy computation), the ratio of the number of sample points lying inside A to the overall number of points N approximates the surface integral ratio $\int_U \delta\, dA / \int_X \delta\, dA$.

We now give an overview of the steps of the algorithm. Pseudocode for the overall point distribution algorithm is given in Figure 4.1.

First, the adaptive space-filling curve C is generated, (see Figure 4.1, lines 1–3) with respect to the surface X (via the parameterisation f), according to a minimum area coverage parameter a, a density δ, and a maximum recursion depth r. The depth r is unlikely to be reached, and is set only as a maximum limit on memory usage. As shown in Section 2.2.2, a space-filling curve can be generated to different approximation levels, with the number of vertices in the curve increasing with the approximation level. We generate an adaptive curve so that we can control the number of vertices locally. This process of adaptive generation is referred to as curve subdivision, and each local section of the curve may have its

Algorithm DISTRIBUTEPOINTS (f, δ, r, a, N, C)

Input: f—parametrisation $f : [0,1]^2 \rightarrow \mathbb{R}^3$ for surface X
δ—density function $\delta : X \rightarrow \mathbb{R}_0^+$
r—maximum recursion depth for space-filing curve approximation
a—parameter for the minimum required area associated with a curve vertex h_l
N—number of points that should be distributed on X
C—type of space-filling curve

Output: A low-discrepancy point distribution on X 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, f, o)
3. $[h_1, \ldots, h_M] \leftarrow$ OUTPUT(T)
4. $[H_1, \ldots, H_M] \leftarrow f([h_1, \ldots, h_M])$
5. $S_0 \leftarrow 0$
6. **for** $l \leftarrow 1, \ldots, M$ **do**
7. $\quad S_l \leftarrow S_{l-1}+$DENSITY ($f$, δ, H_l)
8. $[q_1, \ldots, q_N] \leftarrow$ CREATE1DPOINTS(N)
9. $[p_1, \ldots, p_N] \leftarrow$ EQUALISE($[q_1, \ldots, q_N]$, $[H_1, \ldots, H_M]$, $[S_1, \ldots, S_M]$)
10. **return** $[p_1, \ldots, p_N]$

Figure 4.1: Point Distribution Algorithm.

own approximation level. The curve is subdivided using a quad-tree (or similar) structure until the desired density is achieved. The minimum area parameter, a, defines a minimum lower bound for the size of a space-filling curve quad on the surface. This ensures that even for an area of the surface that is highly stretched, the tree must be subdivided enough, such that even when highly stretched, the quad's area is smaller than a. A suitable value for a can be chosen as half the required maximum Euclidean distance between vertices of the space-filling curve on the surface (see Section 4.2.1). The initial permutation of the space-filling curve is also passed as a parameter (see Figure 4.2). The path through these quads is returned by as a set of ordered space-filling curve vertices h_l, $l = 1, \ldots, M$, which make up a piecewise linear space-filling curve in $[0,1]^2$ (see Figure 4.1, line 3).

The parameterisation f is then applied to the set of vertices h_l, resulting in a set of vertices on the surface H_l (see Figure 4.1, line 4). Surface area elements for each vertex are then

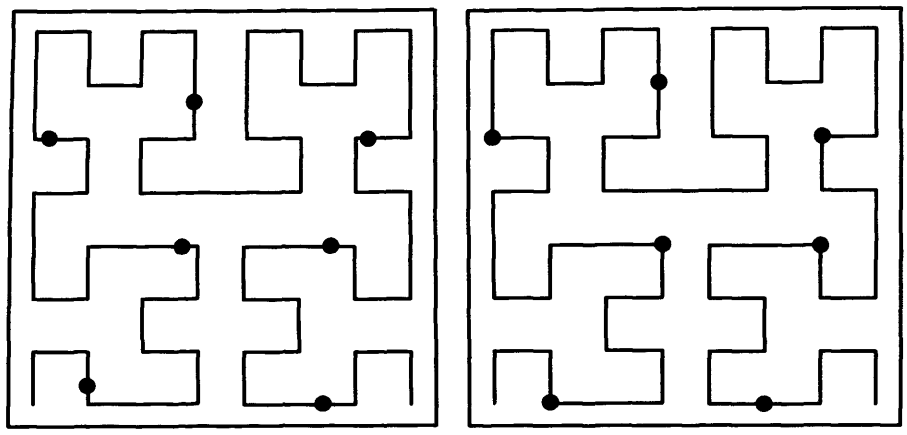


Figure 4.2: Left: sampling at any position along the curve; right: sampling only at the vertices.

calculated, and a cumulative surface integral along the curve is computed using the area, and density function δ (see Figure 4.1, lines 5–7). The cumulative surface area integral, computed along the curve and weighted by the density function, is used in the equalisation step of the algorithm to evenly distribute points on the surface.

A 1D sequence of real numbers in the unit interval $[0, 1]$, q_l, $l = 1, \ldots, N$, is then generated (see Figure 4.1, line 8). The 1D sequence provides the intended distribution of points that is used to sample the space-filling curve. This sequence is mapped to the curve according to the cumulative surface integral and the local level of recursion of the space-filling curve (see Figure 4.1, line 9). This equalisation step adjusts the position of the 1D sequence of points according to the surface integral. By distributing the points according to the local surface area and density in this manner, we correct for the stretch introduced by the parameterisation, and output a set of high quality points p_l, $l = 1, \ldots, N$, with respect to δ (see Figure 4.1, line 10).

4.2 Space-filling Curve Generation

In Section 2.2, a space-filling curve definition was given, along with a brief discussion of the various curves used in this work. We now look at their construction and generation. Interpreted as a mapping: $C : [0, 1] \xrightarrow{onto} [0, 1]^2$, space-filling curves are a special kind of fractal curve [79], which may be constructed using an L-system algorithm [101], where a piecewise linear constructor curve is recursively refined by replacing each of its line segments with a transformed (usually scaled, rotated and translated) version of the initial constructor. This is repeated recursively until a certain depth has been reached. The

result is an approximation of the limit curve. As discussed in Section 2.2.2, the main requirement for this algorithm is that the curve fills $[0, 1]^2$ and is not self-intersecting. Various traditional space-filling curves can be used, including the Peano, Peano II, and Hilbert curves (see Figures 2.4, 2.5, 2.3). Various other curves also exist, such as the Z curve [88] and a zig-zag curve (see Section 2.2.2).

If we treat an approximation of a space-filling curve as a set of connected vertices, to sample a set of points on this curve, we must first decide to sample points only at the vertices, or anywhere along the curve (see Figure 4.2). The answer to this is not immediately obvious. From a purely implementational point of view, it makes sense to sample points at the vertices, as we can pre-compute their locations, and simply flag a vertex as a sample point, or not. This means resampling (see Section 7.1.3) is much more feasible, and the memory overhead is reduced. However, ignoring the computational cost (as we are more concerned with the quality of the distribution), we must investigate what effect it has on the output distribution. At the limit of what may be represented using floating point numbers, the output sample distribution for both techniques would be almost identical. At a very low curve iteration, due to the coarse discretisation, neither method could be used to produce a high quality sampling. However, allowing points to be placed on the curve, between vertices, may in fact introduce more error into the distribution, as the position of each sample point will be dictated by the local shape of the curve, not just the local density. So, assuming a high ratio of vertices to sample points, if we sample points only at the curve vertices, we are simply discretising $[0, 1]$ further, and therefore discretising the histogram; sampling at the curve vertices will simply 'snap' a sample to the nearest unoccupied vertex. It also means that sample points will always be minimally separated by the uniform distance between curve vertices. By adaptively generating the curve, we can locally go to a recursion level at which the exact placement is sufficiently close to the vertex position to make no real difference, thus removing the problem of coarse placement.

We now look at the construction of the space-filling curves used in this work. Distances between consecutive, connected vertices in the Z curve (see Figure 2.2) are variable, and the problem that this poses to our sampling application (see Section 2.2.2) makes it undesirable for use in this work. The Peano curve (see Figure 2.4) shares a very similar construction method with the Peano II curve(see Figure 2.5), but vertices are visited more than once (see Section 2.2.2), meaning that sampled points could overlap. Comparatively, the advantages of the Hilbert curve are the low rate at which the number of vertices that it is composed of increases with respect to the recursion depth, its quad-tree structure and simple geometric construction, and its good spatial coherency (see Section 2.2.2). Whilst the rate at which the number of vertices composing the Peano II curve increases is faster than the Hilbert curve, and its spatial coherency is worse, it is still a viable space-filling

	Vertex (i)			
Permutation state (j)	1	2	3	4
A	B	A	A	D
B	A	B	B	C
C	D	C	C	B
D	C	D	D	A

Table 4.1: Permutation rules for the Hilbert curve.

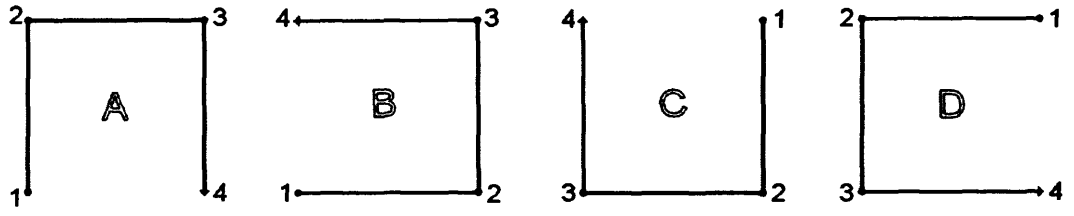


Figure 4.3: Construction permutations of the Hilbert curve.

curve for our application, and thus its construction is described in this section, and the discrepancy of points sampled using it in the plane is investigated experimentally (see Section 5.1.2). The construction principles of these two curves are:

Hilbert Curve

The generation of the Hilbert curve approximation uses a recursive construction, as seen in Figure 2.3). A permutation state j describes an ordering of four quads (A, B, C, D), defined in Figure 4.3. The centre point of a quad is a vertex, and the four quads make a set of vertices $\{e_l : l = 1, \ldots, 4\}$. If a quad needs to be subdivided, we store its index i and the permutation j of the four quads that make up e_l. It is then subdivided into four new quads, and a vertex is placed in the centre of each new quad. These vertices are ordered according to element (i, j) in Table. 4.1, giving a new set of vertices $\{e'_l : l = 1, \ldots, 4\}$. Vertex e'_1 is then connected to e_{i-1}, if $i \geq 1$, and vertex e'_4 is connected to e_{i+1}, if $i \leq 3$. The set of vertices resulting from this subdivision forms a properly connected piecewise-linear curve.

Peano II Curve

The approximate geometric interpretation of the the Peano II curve is a ternary recursive construction (see Figure 2.5). The unit square is divided into nine regular squares, which are recursively subdivided until the required level of approximation is reached. Each cell contains a vertex, and like the Hilbert curve, the path of the Peano II curve can be described by four permutations (see Figure 4.4). Table 4.2

	Vertex (i)								
Permutation state (j)	1	2	3	4	5	6	7	8	9
A	A	C	A	D	B	D	A	C	A
B	B	D	B	C	A	C	B	D	B
C	C	A	C	B	D	B	C	A	C
D	D	B	D	A	C	A	D	B	D

Table 4.2: Permutation rules for the Peano II curve.

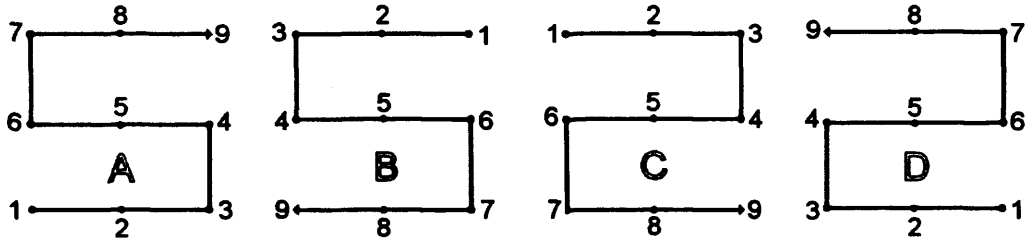


Figure 4.4: Construction permutations of the Peano II curve.

shows these permutations. The curve is otherwise constructed in the same manner as the Hilbert curve.

4.2.1 Adaptive Hilbert Curve

In this section, we describe our algorithm for computing an adaptive Hilbert curve for surface sampling. Whilst there are various space-filling curves that could be employed in our sampling algorithm, the Hilbert curve demonstrates the best spatial coherence (see Section 2.2.3), is straight forward to generate geometrically using a quad-tree structure, and its complexity scales slowly with respect to the recursive depth. Point distributions produced using our algorithm and the Hilbert curve also demonstrate the lowest discrepancy when compared to distributions on other curves (see Section 5.1.2). For these reasons, whilst our approach is generally applicable to many space-filling curves, we focus on the Hilbert curve.

If a space-filling curve with a uniform recursive depth is generated in the parameter domain $[0,1]^2$, and mapped via the parameterisation f onto the surface X, the local scaling between area elements in the parameter domain and the surface will cause the uniform depth curve to not be distributed uniformly on the surface, introducing parametric stretch

to the curve. To correct for this problem, an adaptive space-filling curve is generated that is locally refined based on the scaling of the area elements, resulting in a curve that demonstrates a constant minimum density of vertices on X (see Figure 4.5). This minimum could be reached simply by generating a uniform space-filling curve with a high enough recursion depth. However, that solution, especially for extreme local stretch in the parameterisation, would be highly computationally and memory intensive. Local refinement can further be controlled by the density function δ, allowing the generation process of the curve to not only take into account the area scaling, but also the final required distribution of sample points.

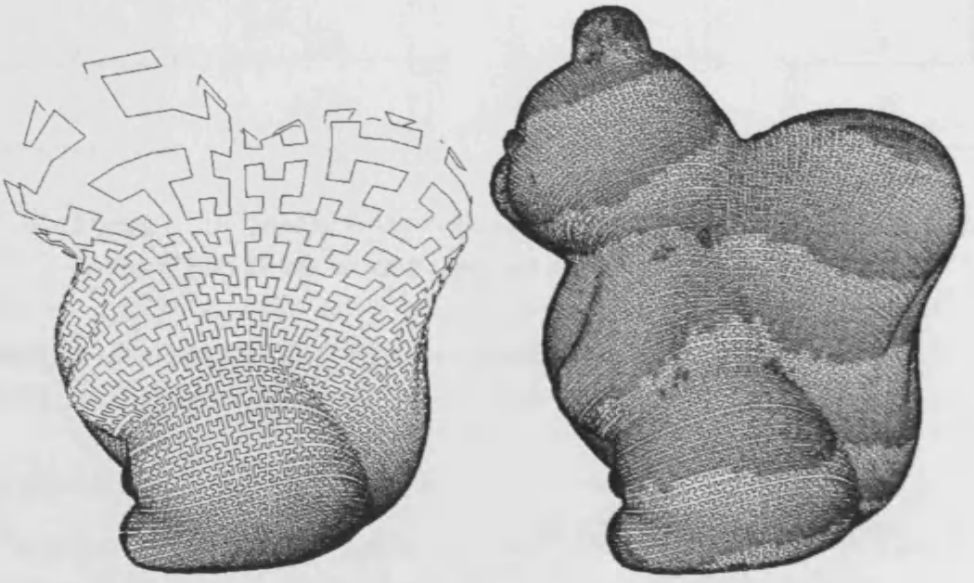


Figure 4.5: Uniform (left) and adaptive (right) Hilbert curve.

We compute an adaptive approximation of the Hilbert curve, $\mathcal{C}_h : [0,1] \xrightarrow{onto} [0,1]^2$, 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 $\{h_l : l = 1, \ldots, M\} \subset [0,1]^2$ expressed by 2D co-ordinates, which uniquely correspond to 1D co-ordinates via the inverse of $\mathcal{C}_h$. Note that although a change in curve vertex density will affect the final positioning of the sample points, it does not affect the quality of the sampling, so long as the minimum density of curve vertices is reached. In Section 4.5, we discuss how the surface parameterisation may affect this minimum lower bound.

An approximate lower bound of the recursive depth of the Hilbert curve is some constant ω times the ratio $\int_U \delta \, dA / \int_X \delta \, dA$, for each subset U of X, in order to provide sufficient sampling accuracy. In practice, due to experimental testing, we find that $\omega \geq 10$ is sufficient to provide the required density of Hilbert curve vertices with respect to the density

function δ, regardless of parametric stretch or distortion. The Hilbert curve vertices are later used to compute the integrals needed to determine the final sample points. Locally on the surface, each surface patch must have a certain minimum number of Hilbert curve vertices so that the final point distribution can be sampled with sufficient accuracy. In practice, a can be set according to the number of points, N, specified by the user, and the approximate ratio of vertices to required points, ω, in one axis of $[0,1]^2$, with respect to the density function δ, $a \approx (\sqrt{\omega N}/2)\delta$.

We list the pseudocode for the adaptive Hilbert curve algorithm in Figure 4.6. We begin by detailing how the required recursion depth of the Hilbert curve is computed, followed by a detailed description of the adaptive Hilbert curve algorithm. The function discussed in this section, POPULATECURVETREE, is called from the DISRIBUTEPOINTS algorithm (see Figure 4.1). POPULATECURVETREE is used to generate an approximation of a Hilbert curve in the unit square. A quad-tree is generated using this method according to the geometric description of the Hilbert curve C_h. We use a tree to facilitate the recursive 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 four quads centred around each curve vertex

Algorithm POPULATECURVETREE (n, r, a, g, f, o)

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 h
g—current depth of recursion
f—parametrisation $f : [0,1]^2 \rightarrow \mathbb{R}^3$ for surface X
o—permutation state defining the local shape of the curve

Output: A space filling curve defined by n, in a tree structure

1. **if** $g < r$ **and** ASSESSSUBDIVISION(a, f, n)=`true`
2. $[c_1, \ldots, c_4] \leftarrow$ GENERATECHILDREN(n, o)
3. $o' \leftarrow$ APPLYPERMUTATION(c, o)
4. **for** $c \in [c_1, \ldots, c_4]$ **do**
5. $c \leftarrow$ POPULATECURVETREE(c, r, a, $g+1$, o')
6. $n \leftarrow$ ADDCHILD(n, c)
7. **return** n

Figure 4.6: Curve generation algorithm.

The inputs to the method POPULATECURVETREE are a root node n, the maximum recursive depth of the curve, r, a, describing the minimum lower bound of the area of a Hilbert quad on the surface X, the current recursive depth g, the surface parameterisation f, and the current permutation of the Hilbert curve o. A node in this quad tree stores the centre point of the quad and its index with respect to its siblings.

First, the current depth g is compared to the maximum recursion depth cap r (see Figure 4.6, line 1). If $g < r$, the method ASSESSSUBDIVISION is called, which applies the supplied parameterisation f to the node n, and calculates the straight-line distances from the centre point to the corners and half-way points along each side. This measurement provides enough accuracy for the initial subdivisions, and the distance approximation has good asymptotic behaviour due to the subdivision of the cells. The maximum distance is taken, and compared to the minimum cell area a. If the value is greater than a, a true result is returned, and GENERATECHILDREN is called on the node n (see Figure 4.6, line 2). GENERATECHILDREN subdivides n, and returns four new nodes $c_1, \ldots, c_4$, with the correct permutation according to o and the index of the parent node n (see Figure 4.1). A new permutation o' is then computed (see Figure 4.6, line 3). The method POPULATECURVETREE is then recursively called on each node c (see Figure 4.6, lines 4–5). r may be set as a limit on the maximum memory requirements of the process (see Section 4.5). The output of POPULATECURVETREE is a subdivided node c, which is then added to its parent node n (see Figure 4.6, lines 5–6). Finally, the node is returned (see Figure 4.6, line 7). If the node is a leaf node (Figure 4.6, line 1 returns `true`), then n with no added children is returned.

As demonstrated in Figure 4.6, line 3, we use the quad-tree structure to generated a list of ordered Hilbert curve vertices. If the quad-tree is desired to facilitate neighbourhood operations and further subdivision, we must store the tree (see Section 7.1.3).

4.3 Computing Integrals for Curve Sampling

Generating the adaptive curve results in an ordered set of vertices h_l (see Figure 4.1, lines 1–3). These must be mapped onto the surface X via parameterisation f, resulting in an ordered set of vertices on the surface: $H_l = f(h_l), l = 1, \ldots, M$. The H_l give the surface space-filling curve C_s. In order to compute the final point sampling of X along C_s, we must compute cumulative densities S_l along the curve, representing the surface integral $\int_{S_l} \delta \, dA$ where S_l is the subset of X covered by the surface space-filling curve from vertex H_1 to H_M (see Figure 4.1, lines 6–7). Then we are able to sample a sequence of points p_l along the surface space-filling curve in a similar way to histogram equalisation (see

Figure 4.1, lines 8–9). To compute S_l, we compute the density s_k at each surface vertex H_k, and then accumulate the values to approximate the surface integral,

$$S_l = \sum_{k=1}^{l} s_k, l = 1, \ldots, M \tag{4.1}$$

Note that S_M is the approximation of the total integral of δ over X. To calculate the local density s_k for each H_k, each of the vertices is associated with a small surface patch A_k of the surface X for which we approximate $s_k = \int_{A_k} \delta \, dA$ (Figure 4.1, line 7). Note that the overall surface integral of δ over X can be ignored as it is simply a scaling factor if we fix the desired number N of sample points.

We have considered several different methods to approximate $s_k = \int_{A_k} \delta \, dA$ (see Section 5.1.4):

(a) Compute the area of the triangle H_{k-1}, H_k, H_{k+1}, and use the value of the density function at its centroid;

(b) Use a small square centred at H_k 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 use the value of the density function at its centroid;

(c) Use $\delta(H_k)\sqrt{\det \mathcal{I}_{H_k}}$ where $\mathcal{I}_{H_k}$ is the first fundamental form of X at H_k, as $dA = \sqrt{\det \mathcal{I}} \, du \, dv$ (see Eq. 2.22).

Also note, that for approach (c), we have to scale the results by $w^{g_{\max} - g_{\text{current}}}$, where $g_{\max}$ is the maximum and g_{current} the current depth of the space filling curve recursion and w is the number of vertices in a curve quad. This is necessary to account for integrating over the area element dA over the adaptive curve where the curve vertices do not correspond to areas of the same size (otherwise, in the non-adaptive case, this is just a constant scaling factor for each element and could hence be ignored). However, it is not required for approaches (a) and (b) as the computed area of a patch is of similar size to the surface patch represented by each H_k vertex.

We then generate a set of 1D samples $\{q_l : l = 1, \ldots, N\}$ in $[0, 1]$, where N is the user-defined number of points in the sampling (see Section 4.4). Once we have calculated the cumulative densities S_l, we move along the curve, and whenever S_l becomes larger than the threshold described by the 1D sequence q_l (multiplied by S_M), we sample a point p_l at a surface space-filling curve vertex on X (Figure 4.1, line 9), producing the output distribution.

4.4 1D Sequence Generation

We now describe various methods for generating the 1D point sequences (Figure 4.1, line 8) in the unit interval $[0, 1]$, to be mapped onto the space-filling curve based on fractional arc-lengths. We considered four different point sequences:

- *Randomly spaced points.* This is mainly intended to be a control under the assumption that a random 1D sequence would create a 2D point distribution with similar properties to a random 2D distribution. A pseudo-random number generator was used to produce the sequence. Sampling with a random sequence of points demonstrates poor equidistribution, and poor results with respect to discrepancy. The main reason for this is that samples are placed independently of each other, so a good coverage is harder to ensure. We have chosen this sequence to experimentally investigate whether the mapping process alone has any influence on the discrepancy behaviour. Results demonstrate, as expected, that the discrepancy of this set behaved in a similar way to a 2D random distribution (see Section 5.1.2).

- *Evenly-spaced points.* The interval is simply divided by the number of required samples, giving a uniform distance between the points. Initially this was intended to be a control, but as shown in Section 5.1.2, it has unexpectedly good properties, which motivated the following jittered sequence. A 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, the points are regularly placed with respect to the grid of vertices, resulting in highly visible aliasing artefacts.

- *Jittered evenly-spaced points.* Jittered sampling was discussed in Section 2.1.2 as a method to reduce the variance of random sampling by splitting the domain into a number of equally sized strata, then randomly placing a point within each stratum, giving a certain uniformity to the distribution. Jittering is performed by computing an evenly-spaced distribution of samples, and shifting each point by a random amount up to half-way toward the next or previous point. Jittering the points removes the effect that the curve structure has on the point set because of the random element, and removes the possibility of aliasing when the number of points and vertices is commensurate.

- *Low-discrepancy points.* The use of low-discrepancy sequences is based on the argument that having 1D low discrepancy may lead to 2D low discrepancy on the surface. All of the four low-discrepancy sequences introduced in Section 2.1.2 are tested: the Niederreiter, Sobol, Hammersley and Halton sequences. To generate the

samples, Java versions of the four construction algorithms are employed based on standard implementations of these sequences. The Niederreiter sequence construction algorithm is based on the *Collected Algorithms from the ACM:* Algorithm 738 [22]. The Sobol sequence is based on the *Collected Algorithms from the ACM:* Algorithm 659 [20]. The first 100 numbers in both the Niederreiter and Sobol sequences are discarded to avoid the leading zeros phenomenon [21]. The Hammersley and Halton sequence construction algorithms are based on the algorithms described in [125].

Results of experiments computing the star discrepancy for all of these 1D sequences sampled on the Hilbert and Peano II curves are shown for point distributions in the unit square in Section 5.1.2. They indicate that the jittered sampling method produces the most consistent results, and, also, samples can be more quickly generated than for the low-discrepancy sequences. Also, the jittered approach cannot cause aliasing problems like the evenly spaced sampling approach. For these reasons, we recommend the use of a 1D jittered sampling approach as it gives the best general performance for our sampling algorithm. However, it is worth noting that this produces a probabilistic distribution. For some approaches, where repeatable results are required, a robust deterministic sampling method should be used, such as a low-discrepancy or evenly-spaced sequence.

4.5 Reparameterisation

In this section, we discuss the effect that the surface parameterisation has on our sampling approach. Surface parameterisation is the process of mapping from a parameter space onto a surface embedded in some space (here always $\mathbb{R}^3$). If a parameterisation is not given, finding a suitable set of u, v parameters for a known surface in 3D is a difficult problem, typically putting constraints on shape or area in the process (see Section 3.2.3).

In our algorithm, we take, as an input, a parametric surface over the parameter space $[0, 1]^2$, which we then sample with a space-filling curve. This allows us to map a space-filling curve onto the surface. However, in general the parameter space is not mapped onto the surface in a distance preserving manner, meaning the distance between two points in parameter space and image space can differ by a large amount; thus areas of regions in parameter and image space are also not preserved. The stretch is determined by the properties of the parametrisation.

Algorithmically, we generate an adaptive space-filling curve (see Section 4.2.1) to counter this parametric stretch introduced by the parametrisation; more space-filling curve vertices

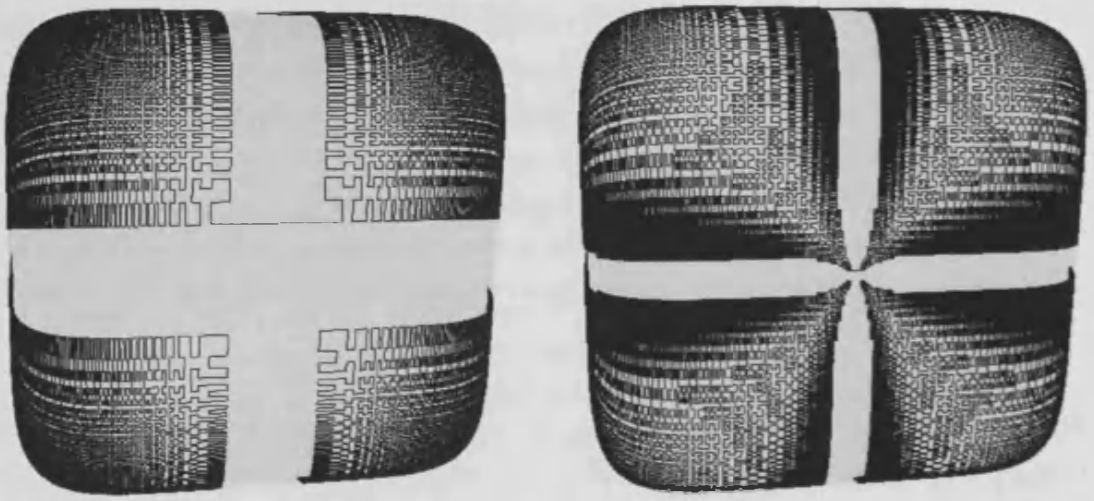


Figure 4.7: Superquadric.

are generated in areas which are stretched to a greater degree. Thus whilst the resulting space-filling curve verices are not necesarily uniformly distributed on the surface, there is a global minimal local density of curve vertices everywhere on the surface. This ensures that there are sufficient vertices on the surface to compute the integrals and produce a final point sampling accurately. Figure 4.7 shows the Hilbert space-filling curve mapped onto the surface of a super-ellipsoid, where $u, v \in [0,1]^2$:

$$x(u,v) = \cos^{1/3} u \cos^{1/3} v, \tag{4.2}$$

$$y(u,v) = \cos^{1/3} u \sin^{1/3} v, \tag{4.3}$$

$$z(u,v) = \sin^{1/3} u \tag{4.4}$$

with and without adaptive curve generation. This surface has a very high degree of parametric stretch. 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 non-adaptively generated curve.

However, 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 using the algorithm in Figure 4.6. The coordinates for a point in 1D are limited by machine precision. Mapping this point via a function into 2D results in in a coordinate with two components generated from the single 1D component. Thus, a 1D point represented by b bits is mapped onto a 2D point with each co-ordinate having only $b/2$ bits. Thus, two distinct points in 1D differ in the 2D parameter domain by at least $e = 1/(2^{b/2} - 1)$ in one coordinate. Otherwise they will

correspond to only one point in the 1D point sequence when a point sampling is being generated. 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. We can compute the area element $dA = \sqrt{\det(\mathcal{I})}\, du\, dv$ from the first fundamental form, $\mathcal{I}$, for a point on the surface. We can further compute the surface area covered by any e^2 sized parameter area E, using $\Delta_E = \int_E \sqrt{\det(\mathcal{I})}\, du\, dv$ (for simplicity we ignore the density δ here). E can then at most be covered by one point on the surface. Or, Δ_E/e^2 gives the local surface stretch and with that a local point density limit achievable with a given machine precision. This depends on the parametrisation, and not just the first fundamental form, due to the E in parameter space. Note that here we only consider the numerical resolution limit and not any other numerical problem arising from evaluating the parametrisation, etc.

In practice, in our algorithm we set a maximum recursion level and the parameter a giving the minimum lower bound on area associated with a surface space-filling curve vertex. In combination, these parameters determine the numerical resolution and with that limit the achievable point density given a certain parametric stretch. As the scaling factors between the surface and the parameterisation become extreme, the desired density may not be achievable as illustrated by the super-ellipsoid example above (Figure 4.7).

One solution to this problem is to use multi-precision arithmetic. However, whilst this would provide a solution to bypass machine precision limits, it is computationally expensive, and is still fundamentally memory-limited due to the recursive depth of the curve that might be required.

An alternative to a multi-precision approach is to find a different parameterisation for the surface, enforcing a limit (or a lower limit), on the first order partial derivatives of the parametrisation (effectively limiting the norm of its Jacobian). In practice, one approach to the reparameterisation of a surface is to distort the parameter space by another function that stretches it: the mapping f between the parameter domain $[0,1]^2$ and the surface in $\mathbb{R}^3$ becomes $f \circ g$ where $g : [0,1]^2 \rightarrow [0,1]^2$. E.g. for the superquadric above, we can reparameterise it piecewise with a simple power law. In the interval $[0, \pi/2]$ we could 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 so on.

Figure 4.8 shows one corner of the super-ellipsoid shown in Figure 4.7, with three curves mapped onto the surface: the image on the left shows a uniform Hilbert curve of depth 7, the middle image shows an adaptive Hilbert curve with a depth betwen 6 and 14, and the image on the right shows a uniform Hilbert curve of depth 7 after employing the above reparameterisation. The adaptive approach results in a considerably better coverage than the constant Hilbert curve recursive depth. In turn, whilst the reparameterisation in this

case is straightforward, it produces a better, more uniform, coverage than the constant and adaptive approaches.

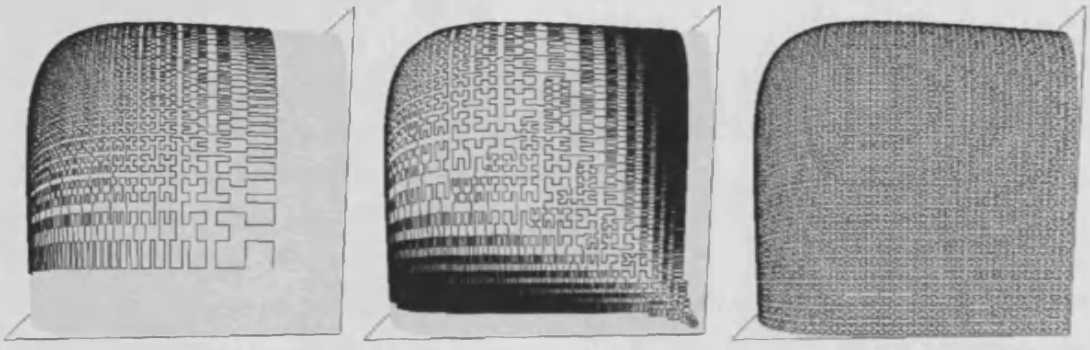


Figure 4.8: Superquadric sampled with Hilbert curve: left, uniform curve; middle, adaptive curve; right, reparameterised uniform curve.

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, Figure 4.8 shows the limitations of this for an extreme parameterisation. To improve this situation for specific parameterisations, we suggest one solution of a reparameterisation of 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). Computing a local reparameterisation of the surface for each quad of the adaptive Hilbert curve would allow for localised reparameterisation that could also take advantage of the adaptive curve generation approach. Reparameterisation in this way also has problems however, as it may be difficult to compute a suitable reparameterisation for more complex functions, especially when a surface is stretched anisotropically (see Section 8.3). We have not further investigated reparameterisation however, as this problem is outside the scope of this work, and has many solutions in the area of mesh parameterisation.

4.6 Summary

This chapter introduced our novel sampling algorithm. This sampling approach produces high-quality low-discrepancy point samples on arbitrary surfaces that are uniformly distributed with respect to the local surface area (see Sections 5.1.3, 5.1.4). The density of the sampling can also be controlled, allowing, for example, higher densities of points in

areas of high curvature. The parametric stretch introduced by the scaling of local area elements between the parameter domain and the surface has also been investigated. Solutions to this problem were discussed, whereby adaptive space-filling curves constructed to machine precision limits and surface reparameterisaiton were used to correct for extreme parametric stretch. In the next chapter, we empirically investigate the quality of the point sample distributions generated using this novel algorithm, using discrepancy, blue noise and visual assessment as measurements.

Chapter 5

Evaluation of Space-filling Curve Point Sampling

In this chapter, we experimentally evaluate the effectiveness of our point sampling algorithm. We consider the discrepancy of point sets (see Section 5.1), the visual quality of the distributions (see Section 5.2), and how they fulfil the blue noise criterion (see Section 5.3). Finally, we briefly test the spatial coherence of space-filling curves (see Section 5.4). We first explain how we measure numerical discrepancy, then present results comparing the distributions produced by our algorithm to other, popular low-discrepancy distributions. We then employ the blue noise criterion, computing the radially averaged power spectrum of point distributions, to determine their visual properties with respect to frequencies of noise. To numerically investigate the spatial coherence of various space-filling curves we compare distances between points along the curve, and distances between the same points in Euclidean space (see Section 2.2.3). Poor spatial coherence decreases the accuracy of our sampling approach (see Section 2.2.2).

5.1 Discrepancy

In order to quantify the quality of point distributions generated by our algorithm, we approximate their discrepancy. Discrepancy measures the uniformity of a point distribution in a set X by measuring the difference between the expected number of points and the actual number of points in a selected subset of X (see Section 2.1.1). The expected number of points is given by the ratio between the area of the subsets and the area of X. The discrepancy can be plotted for point sequences as a scalar series with increasing sequence size. This observed experimental behaviour compared to the theoretical limits of discrepancy and also compared to other sequences allows us to evaluate our novel sampling algorihtm. The aim is to achieve a discrepancy that scales as well as possible as the number of points increases.

In Section 5.1.1, we introduce the numerical measures used to compute the discrepancy of a point set on the plane and on the surface of a sphere, and describe our experimental methodology. We then experimentally evaluate our point distributions according to discrepancy considering three aspects. Firstly, the best combination of space-filling curve and 1D sequence are identified (see Section 5.1.2). As the number of curves tested has been reduced to the Hilbert curve and Peano II curve based on their advantageous properties (see Section 2.2.2), the criteria for choosing the best combination is based on the discrepancy behaviour of the output point sequences for each combination of curve and 1D sequence. This behaviour refers not only to the scaling of the discrepancy as the output point sequences increase in size, but also how robustly the discrepancy scales for various sample shapes used for the discrepancy measure. Secondly, the results for our approach using the best combination of curve and 1D sequence are compared to results for the well-known low-discrepancy point sequences in the plane (see Section 5.1.3). Thirdly, we compare our results to results for well-known point sequences on the surface of a sphere (see Section 5.1.4).

5.1.1 Numerically Measuring Discrepancy

In this section, we describe how the discrepancy of a sequence of points within a unit square is computed, demonstrating how we have generalised these methods to use various sampling shapes. When approximating the star discrepancy of a sequence, an axis aligned rectangle, with one point at the origin, is used as the sample shape. In our experiments, we expand on this to include a quarter circle and a triangle as additional sample shapes, in order to better show how the distribution behaves in more general situations. For rendering, the discrepancy should ideally be independent of any underlying shape. For example, the rectilinearity of the surface being rendered may be unknown, thus sampling it with a point distribution that demonstrates low discrepancy when measured with rectilinear shapes may be a poor approach. The advantages of point sets whose discrepancy is invariant to the sampling shape has been demonstrated for super-sampling patterns used in antialiasing [39]. We also introduce a discrepancy measure on the surface of a sphere, and finally discuss the numerical computation of our discrepancy measures.

Planar Discrepancy

We now introduce planar discrepancy measures. As introduced in section 2.1.1, a commonly used discrepancy measure is the *star discrepancy* D^* (see Eq 2.9). The sampling domain is defined as a set X, chosen to be a unit square, with rectangular, quarter-circle

and triangular test shapes $T \subset X$. The expected ratio of points found in each subset is the ratio of the area of the test shape to the area of X which is 1. The discrepancy, i.e. the supremum of the error, is approximated by computing the maximum error for a finite number of test shapes. As the number of sample points increases, the discrepancy of any sensible sampling method should decrease. The rate of decrease should be high for a low-discrepancy sample distribution; in particular, it should be faster than the rate observed for a random point set. Generally speaking, the faster the discrepancy drops with the number of sample points, the higher quality a point distribution is, at least in the limit. The star discrepancy is also simple to estimate in practice [132]. In our experiments, to numerically estimate the star discrepancy, from the set of all axis-aligned rectangles in the unit square $\mathfrak{R}$ in X, we select a subset $\mathfrak{R}_0(L)$ of L rectangles, with $L = 1000$, with one corner at the origin and the diagonally opposite corner at a random position v_0 in the unit square. The value of L was found by observing the convergence behaviour of the discrepancy measurement, and set slightly above the value where no further change was typically observed. The random positions were computed using a pseudo-random number generator. We then used a standard approach to estimate the star discrepancy:

$$D^*(P) \approx \max_{T \in \mathfrak{R}_0(L)} \left| \frac{|P \cap T|}{|P|} - \text{area}(T) \right| \tag{5.1}$$

where $|P \cap T|$ is the number of points in the rectangle T.

Here we generalise the usual star discrepancy measure to better show how each point sequence behaves by using triangles, quarter-circles, and rectangles as test shapes. These test shapes were chosen as the triangle highlights the discrepancy of shapes with arbitrary edge directions, and the quarter circle for curves. While theoretical results for these discrepancy measures are not widely available (see [132]), we can compute numerical results in the same way as for the rectangles, allowing us to investigate how different point distributions perform with different test shapes (see Figure 5.1). Triangles were chosen with one point at the origin and the other two points, $v_0, v_1 \in X$, chosen randomly. Random quarter-circles were generated by placing the centre of a quarter-circle at a corner of X and computing a random radius $0 < r \leq 1$ such that the whole quarter-circle is confined within X. Whilst this means that a section of X cannot be sampled by a particular subset shape, the corner for the centre is chosen at random for each T to ensure all of X is sampled.

Surface Discrepancy

We now discuss discrepancy measures on surfaces. Low-discrepancy sampling methods have become a popular method to compute surface areas of 3D models [28, 75, 76]. Ap-

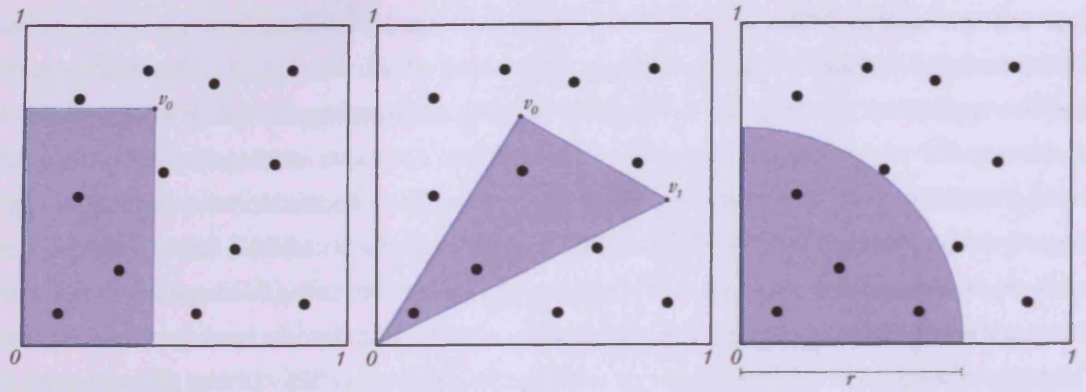


Figure 5.1: Rectangle, triangle and quarter-circle test shapes for the star discrepancy measures.

plying the Cauchy-Crofton formula [109], these methods compute a bounding surface of the model, through which they generate lines, computing intersections between the lines and the original surface. The lines are constructed through pairs of low-discrepancy points, thus the properties, such as uniformity, of these points is very important. However, approaches such as these rely on the accuracy of the application output, surface area computation, as a method of testing. Our algorithm also relies on good uniformity of points for surface sampling, and thus we try to generalise the experimental approach already used on the plane to a surface in $\mathbb{R}^3$. We generalise the star discrepancy on the plane to the surface of a sphere. Generalising the star discrepancy like this provides a good method to evaluate the quality of a surface distribution, as the test is not dependent on the output of an application, and is simpler compared to the various techniques described in [34], which also only provide results for very small numbers of points.

We measure the discrepancy on a parameterised unit sphere rather than an arbitrary surface, for two reasons. Firstly, well known low-discrepancy distributions on the sphere exist as a basis for comparison. Secondly, a simple generalisation of the star discrepancy can be readily defined by

$$D^S(P) = \sup_{T \in \mathfrak{T}} \left| \frac{|P \cap T|}{|P|} - \frac{\text{area}(T)}{4\pi} \right| \tag{5.2}$$

where $\mathfrak{T}$ is the set of all spherical triangles lying on the surface of the unit sphere, and $|P \cap T|$ is the number of points in P inside $T \in \mathfrak{T}$.

To numerically estimate this discrepancy we generate random spherical triangles defined by three points A, B, C on the unit square. For this we select points $(u_l, v_l) \in [0, 1]^2$,

$l = 1, 2, 3$, at random and map them onto the sphere using the following parametrisation:

$$\begin{aligned} x_l &= \cos(2\pi u_l)\sin(\cos^{-1}(2v_l - 1)), \\ y_l &= \sin(2\pi u_l)\sin(\cos^{-1}(2v_l - 1)), \\ z_l &= \cos(\cos^{-1}(2v_l - 1)). \end{aligned} \tag{5.3}$$

This sampling results in a uniformly distributed set of points on the sphere, avoiding a non-uniform concentration at the poles.

The area of a spherical triangle T is given by

$$\text{area}(T) = \widehat{A} + \widehat{B} + \widehat{C} - \pi \tag{5.4}$$

where $\widehat{A}$, $\widehat{B}$ and $\widehat{C}$ are the external angles of the spherical triangle, which can be found using the cosine rule, i.e.

$$\widehat{A} = \cos^{-1}\left(\frac{\cos(a) - \cos(b)\cos(c)}{\sin(c)\sin(b)}\right) \tag{5.5}$$

where a, b and c are the internal angles of the triangle given by, e.g., $a = \cos^{-1}(B \cdot C)$. A point p on the sphere lies inside such a triangle if $p \cdot (X \times Y)$ is positive for $X = A, Y = B$ and $X = B, Y = C$ and $X = C, Y = A$. B and C are computed similarly.

Numerical Approach

In this section we discuss how to compute the numerical discrepancy in the following experiments (see Sections 5.1.2, 5.1.3, 5.1.4). We first investigate in detail the discrepancy properties of 2D point distributions on the plane generated by our implementation using the Hilbert and Peano II curves, sampled using random, evenly-spaced, jittered and low-discrepancy 1D point sequences. We then compare results from our algorithm to other distributions on the plane. Finally, we compare our approach to other distributions on the sphere. To improve clarity of the graphs, when comparing the distributions produced using our algorithm to those produced by other sampling methods, we only compare the other sampling distributions with the space-filling curve distribution that demonstrates the lowest discrepancy: the Hilbert curve with a 1D jittered sampling method. The distributions that we compare our approach to are detailed in Section 5.1.3.

All of the tests were performed for sample sets sizes of $N = 2^l$ and $N = 2^l + 2^{l-1}$, for $l = 1, \ldots, m$. We set m to 20 and so test 40 distribution sizes varying from 2 points to 1572864 points, resulting in a range of data that matches the logarithmic scale the experiments are plotted on which makes it simple to compare the curves' scaling behaviour with the known

theoretical limits. Note that to avoid aliasing problems with evenly-spaced sampling, the number of vertices on the curve generated from the recursive approximation depth g (see Section 4.2.1) and the sizes N were non-commensurate (see Section 4.4).

The adaptive Hilbert curve generation algorithm (see Section 4.2.1) is used to generate the Hilbert curve for the discrepancy experiments described in this chapter. A high ratio of curve vertices to sample points ensures a good output distribution of points. The adaptive Hilbert curve for the planar discrepancy is in fact generated at a constant recursion depth due to the plane's constant area and density. Thus, a large minimum recursion depth for the adaptive curve generation, $g = 12$, is used (resulting in 4^{12} vertices). When tested against the Peano II curve, a recursion depth of $g = 8$ for the Peano II curve (resulting in 9^8 vertices). These high recursion depths are used to ensure that the ratio of vertices to sample points does not affect the results. The Hilbert curve on the spherical surface is generated to ensure the same number of curve vertices as on the plane. We generate a high depth adaptive Hilbert curve on the sphere, resulting in minimum local density of curve vertices on the surface relative to the surface area (resulting in $> 4^{12}$ vertices). Note that the Peano II curve is not tested on the sphere surface because of conclusions drawn from experimental results in Section 5.1.2.

Star discrepancy results are shown using graphs displaying the logarithm of the discrepancy versus the logarithm of the number of points (see Sections 5.1.3, 5.1.4). Although theoretical discrepancy results are characterised by a power law times a logarithmic factor (see Section 2.1.1), the logarithmic factor is hard to determine experimentally due to its minor numerical effect. As can be seen in our graphs, on a log-log graph, 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 scaling behaviour of the different approaches.

For a random sequence the expected slope of a least-squares fitting line is $-1/2$ [92]. Clearly, we hope our point distributions 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 $O(N^{-1}\log^2 N)$ [92], giving a lower bound of approximately -1 for the slope of the best-fit straight line.

5.1.2 Evaluating the Influence of Space-filling Curve Choice on Discrepancy

Our algorithm allows various space-filling curves to be used in the sampling process. The space-filling curves can then be sampled with any 1D sequence to produce the final output

sample distribution. In Sections 2.2.2 and 4.2 we considered various space-filling curves to be employed in our sampling algorithm, and chose the Hilbert and Peano II curves as their geometric characteristics are well suited to our algorithm. In order to find the best combination of curves and the 1D sequences discussed in Section 4.4, we now evaluate their discrepancy.

In this section the discrepancy for both the Hilbert and Peano II curves, sampled with random, evenly spaced, jittered and low-discrepancy points is measured. In order to give a more thorough analysis of the point distributions, we measure the discrepancy using rectangles, quarter-circles and triangles as described in Section 5.1.1.

From now on, we will refer to the Hilbert curve sampled using a jittered 1D sequence as Hilbert-jittered, with a random sampling as Hilbert-random, an even sampling as Hilbert-even, and a low-discrepancy sequence sampling as Hilbert-low. Similarly, we refer to the Peano II curve sampled using a jittered 1D sequence as Peano-jittered, and so on.

Figures 5.2–5.4 show the discrepancy using Hilbert-random, Hilbert-even, Hilbert-low and Hilbert-jittered, measured using rectangles, quarter circles and triangles respectively. Figures 5.5–5.7 similarly show corresponding discrepancies using the Peano II curve, again for rectangular, circular and triangular test shapes. Tables 5.1 and 5.2 summarise the gradients of the least-square lines fitted to the results of these experiments.

1D Sequence	*Rectangular*	*Circular*	*Triangular*
Random	−0.51	−0.49	−0.51
Evenly-spaced	−0.51	−0.70	−0.57
Low-discrepancy	−0.51	−0.71	−0.57
Jittered	−0.73	−0.73	−0.72

Table 5.1: Gradient of least squares fit line for distributions on the Hilbert curve.

1D Sequence	*Rectangular*	*Circular*	*Triangular*
Random	−0.50	−0.49	−0.50
Evenly-spaced	−0.71	−0.71	−0.70
Low-discrepancy	−0.72	−0.70	−0.69
Jittered	−0.70	−0.70	−0.70

Table 5.2: Gradient of least squares fit line for distributions on the Peano II curve.

As we can see in Figure 5.2, using rectangle test shapes, Hilbert-random, Hilbert-even and Hilbert-low perform almost identically. Hilbert-jittered performs significantly better. Figure 5.3 shows the case of quarter-circles test shapes. When compared to the rectangular test shapes, a vastly improved performance is seen for Hilbert-even and Hilbert-

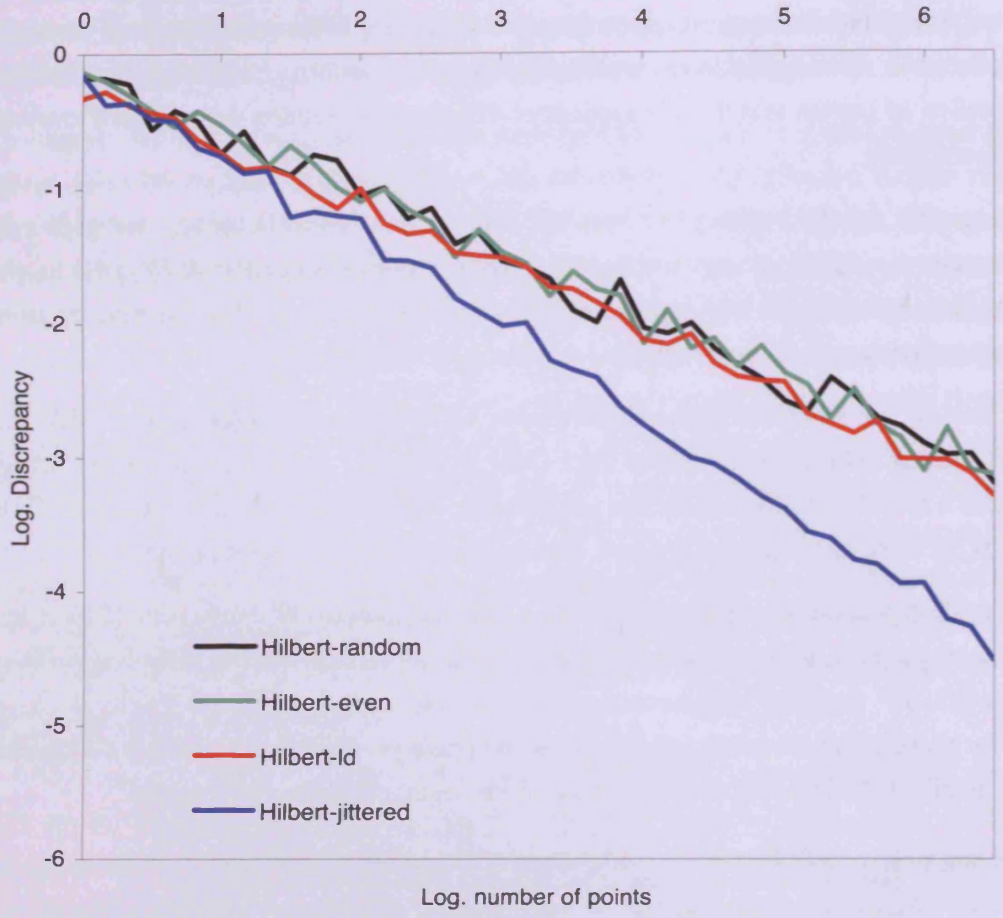


Figure 5.2: The log of the discrepancy of distributions produced using the Hilbert curve, measured using rectangles versus number of points on a logarithmic scale.

low, showing almost as good results as the jittered version; Hilbert-random performance did not improve. Figure 5.4 shows that using triangles as test shapes gives similar results to the rectangle case, although both the Hilbert-even and Hilbert-low distributions slightly outperform Hilbert-random. Hilbert-jittered, however, performs consistently well throughout.

Looking at Figures 5.5–5.7, whereas the Hilbert curve demonstrated varying results for differing discrepancy test shapes, the Peano II curve shows consistent results for all three; Peano-random performs as badly as Hilbert-random, but Peano-even and Peano-low perform consistently throughout, compared to inconsistent results for the Hilbert-even and Hilbert-low. Peano-jittered again performs well throughout.

If we now consider the gradients of the best fit lines for these distributions, listed in Ta-

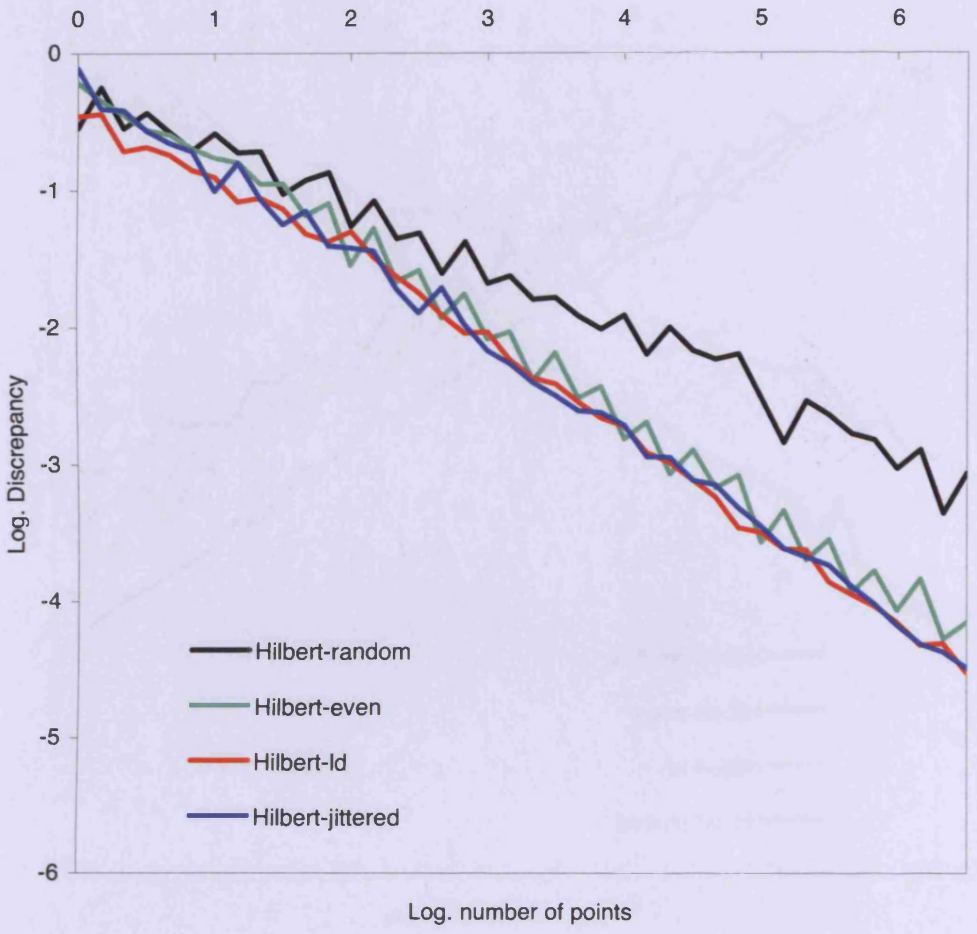


Figure 5.3: The log of the discrepancy of distributions produced using the Hilbert curve, measured using quarter-circles versus number of points on a logarithmic scale.

bles 5.1 and 5.2, we can see that when points are placed randomly along either space filling curve, for any discrepancy measure, the gradient of the best fit line is very close to -0.5, corresponding to the expected result for a random distribution. In the case of the Hilbert curve the gradients confirm our observations of the graphs; only Hilbert-jittered provides consistently good results. For the Peano II curve, Peano-even, Peano-low and Peano-jittered, all show a significant improvement over random sequences, for each discrepancy measure, with a slope of about -0.7 in each case. This is clearly better behaviour when compared to random points, but not as good as can theoretically be provided by a 2D low-discrepancy point sequence, $O(N^{-1}(\log N)^2)$, corresponding to gradient -1.

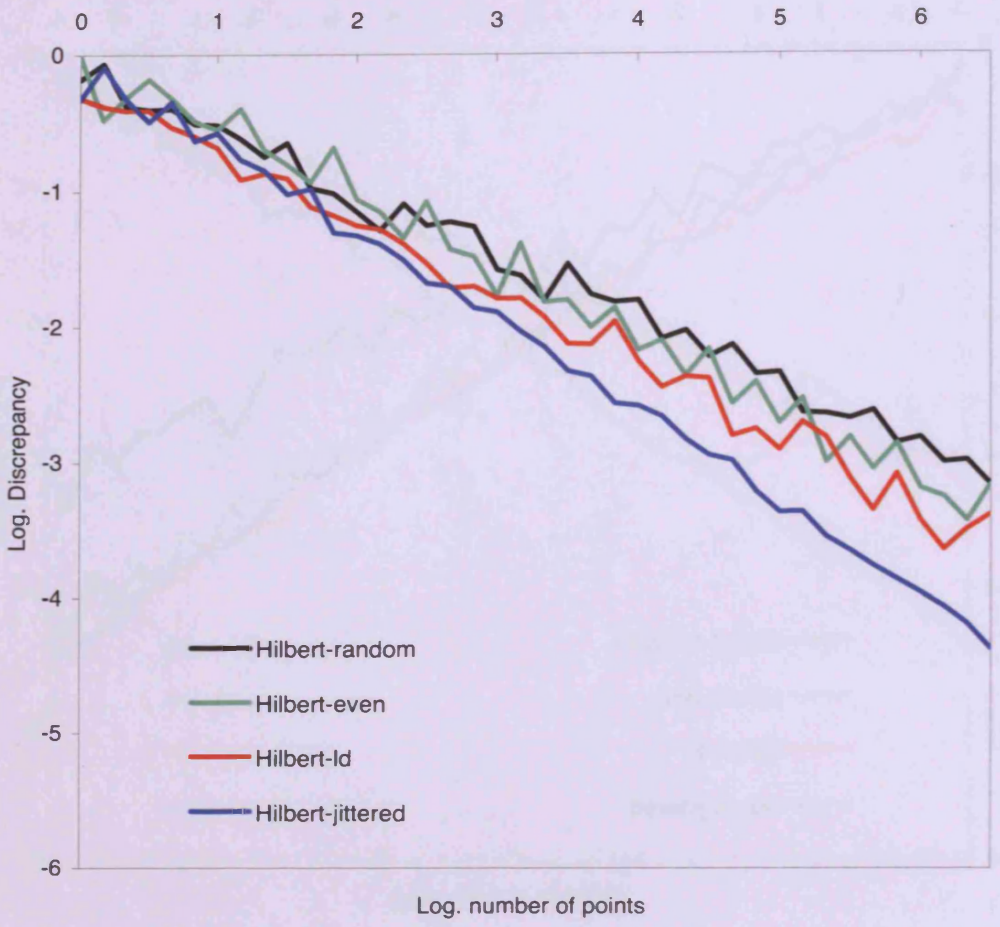


Figure 5.4: The log of the discrepancy of distributions produced using the Hilbert curve, measured using triangles versus number of points on a logarithmic scale.

Discussion

The Hilbert-jittered, Peano-jittered, Peano-even and Peano-low distributions are consistently superior to using either random distribution. It is also clear from the results that there is no need to use a 1D low-discrepancy sequence to generate a low-discrepancy distribution in the plane as Peano-even performs just as well.

Results in this section, specifically Tables 5.1 and 5.2, show that pairing different curves and sequences together results in point sets with significantly different geometric discrepancies. More specifically, for a given 1D sequence, changing the type of space-filling curve can have a very significant impact on the discrepancy of the point set, which further varies based on the geometric sampling shape. For example, sampling the Hilbert curve

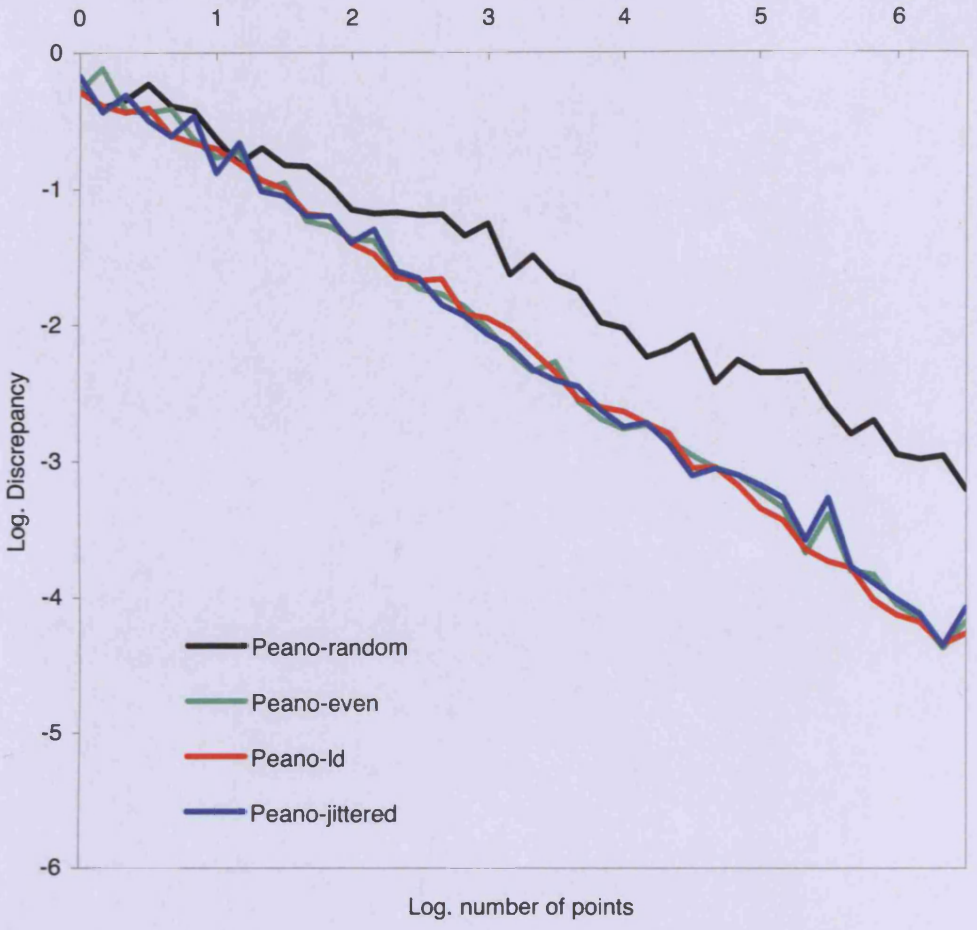


Figure 5.5: The log of the discrepancy of distributions produced using the Peano II curve, measured using rectangles versus number of points on a logarithmic scale.

with an evenly-spaced or low-discrepancy sequence yields a discrepancy very similar to that of random sampling. This observation, that the space-filling curve used can significantly affect the sampling quality, implies that either the number of vertices in a curve, or the underlying structure of a curve causes this variation in point set discrepancy. As a further observation, the random and jittered sequences, which both have a non-deterministic element, demonstrate very consistent results for each geometric test shape. We believe that these observations warrant further investigation.

A possible explanation for the inconsistent behaviour of point distributions on Hilbert curve, is that if the size of N is commensurate with the number of vertices in the curve, aliasing effects can appear if an evenly-spaced distribution is used; points become aligned with the repeated geometric constructs that make up the curve, the effect becoming very

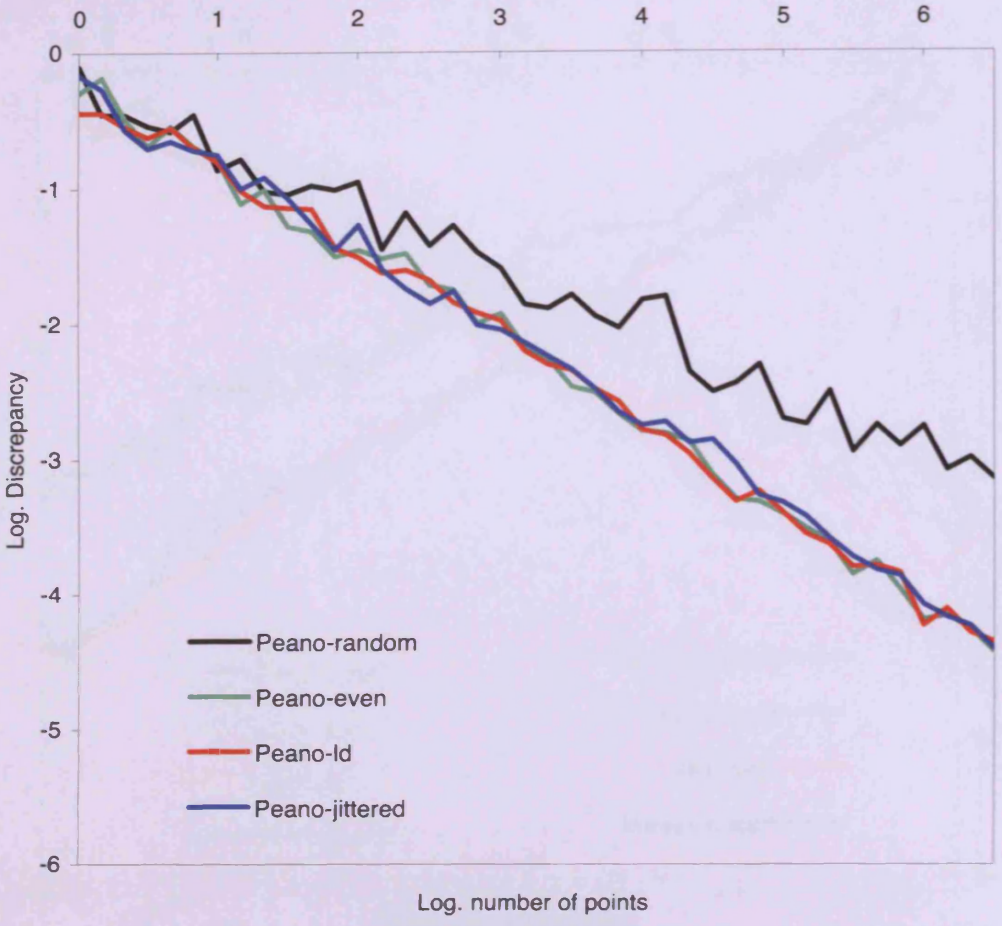


Figure 5.6: The log of the discrepancy of distributions produced using the Peano II curve, measured using quarter-circles versus number of points on a logarithmic scale.

visible to the human eye (see Figure 5.8). By using jittered sampling rather than using evenly spaced points, we we avoid such aliasing problems.

Overall, using a jittered 1D sequence appears to provide the best and most robust, discrepancy scaling behaviour out of all the 1D sequences tested. Furthermore, whilst not producing consistently good results for each 1D sequence, the Hilbert-jittered approach produced the lowest discrepancy results overall. In addition, the Hilbert curve benefits from a better spatial coherence than the Peano II curve (see Section 2.2.3), and the number of vertices increases much more slowly with each recursive step of the Hilbert curve's generation (see Section 4.2). We therefore use the Hilbert-jittered approach for further testing and comparison with other distributions and through the rest of this work.

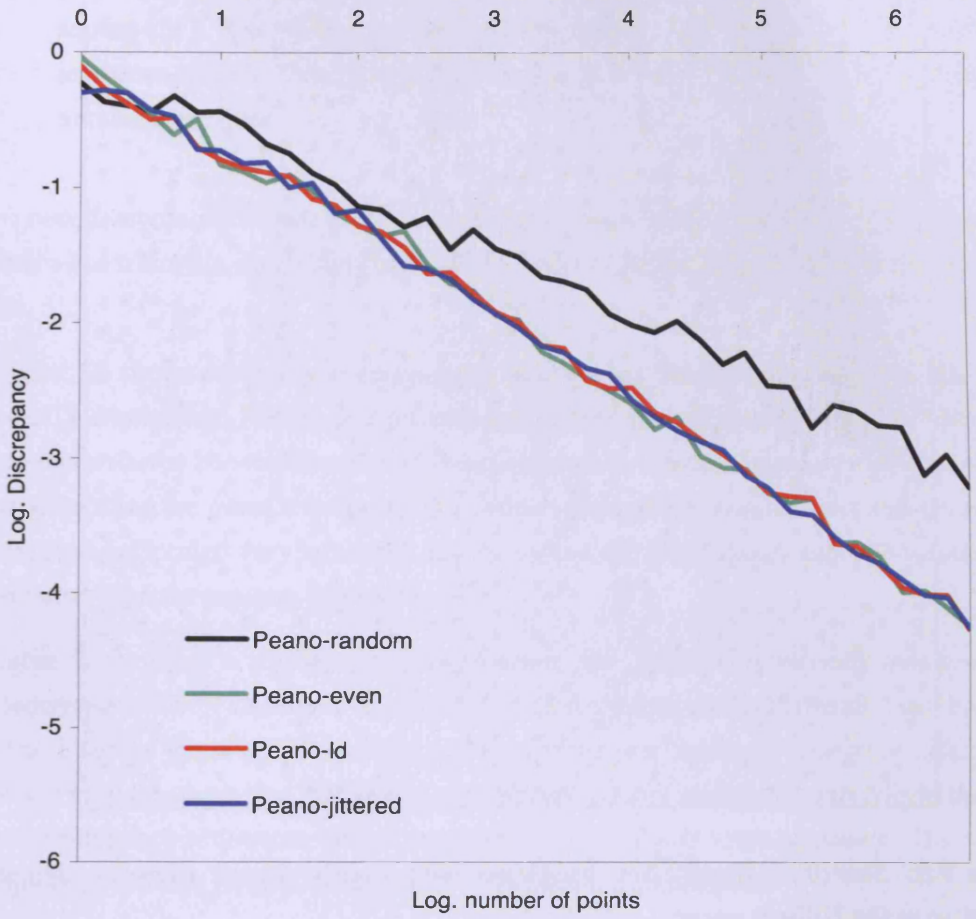


Figure 5.7: The log of the discrepancy of distributions produced using the Peano II curve, measured using triangles versus number of points on a logarithmic scale.

5.1.3 Comparison with Low-discrepancy Distributions in the Plane

In this section, in order to compare our technique to existing methods, we evaluate the quality of the point distributions generated in the plane, comparing the best results produced by our algorithm, using the Hilbert curve with a jittered 1D sequence, to those generated by other well-known sampling techniques for the unit square (see Section 2.1.2): 2D base-2 *Niederreiter* [90] and *Sobol* [116] sequences , 2D base-2 *Hammersley* and *Halton* sequences [125], *jittered sampling* (often called stratified sampling) [115] and *random sampling*:

Niederreiter and Sobol Sampling: Base-2 Niederreiter distributions in the unit square were generated using ACM TOMS Algorithm 738 [22]. Base-2 Sobol distributions

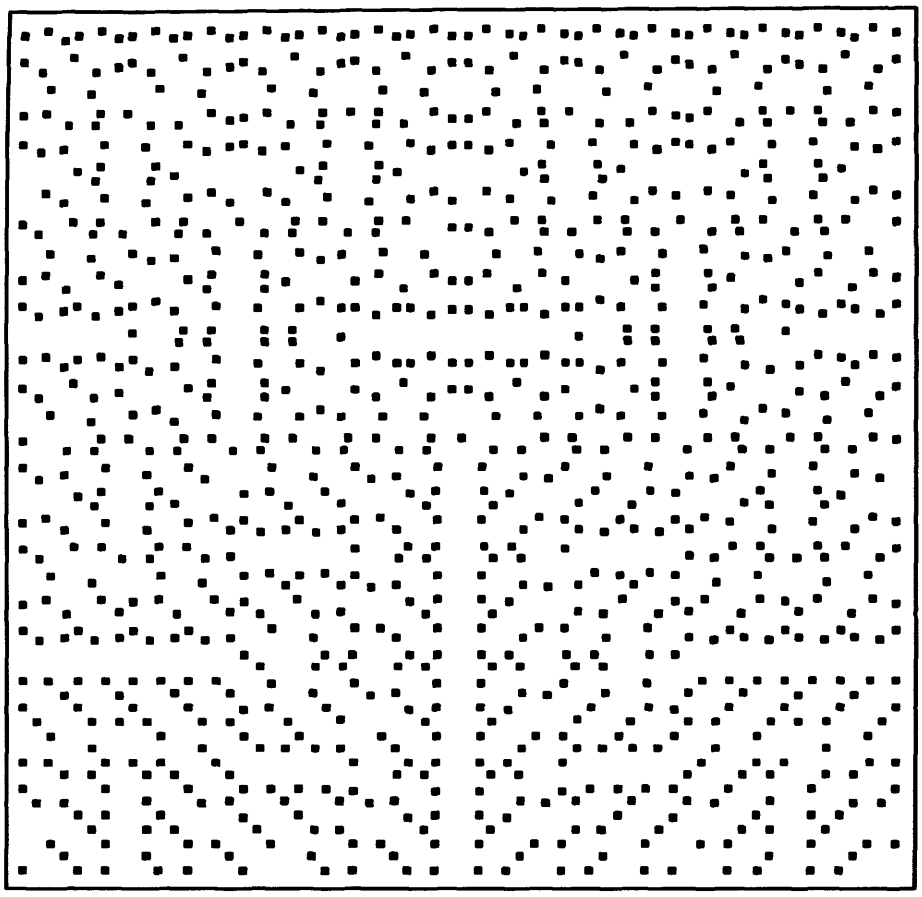


Figure 5.8: Sampling illustrating problems with evenly-spaced samples, demonstrated with the Hilbert curve.

were generated using ACM TOMS Algorithm 659 [20]. The first 100 numbers in both the Niederreiter and Sobol sequences were discarded to avoid the leading zeros phenomenon [21].

Hammersley and Halton Sampling: Base-2 Hammersley and Halton distributions in the unit square were generated using the algorithms described in [125]. These were chosen over other base configurations described in the paper because of their consistently better performance.

Jittered Sampling: Jittered distributions were generated by subdividing the unit square into ceil($\sqrt{N}$) $\times$ ceil($\sqrt{N}$) square cells, and placing a point randomly within each cell. The number of cells, and therefore the number of points actually distributed, is not necessarily the same as the requested number of points, as clearly the number of points must be a perfect square.

Random Sampling: Points are generated in the unit square simply by generating inde-

pendent pairs of random numbers. This gives a random distribution in the unit square [51]. The theoretical discrepancy of a 2D random point set is: $O(N^{-1/2})$ as shown in [68]. Practically, points were generated with the Java pseudo-random number generator.

We now demonstrate results for the planar discrepancy, measured using rectangles, quarter-circles and triangles, comparing our Hilbert-jittered approach to the above sampling methods.

Figure 5.9 shows variation in rectangular discrepancy for the 2D random, Niederreiter, Sobol, Hammersley, Halton, 2D jittered and Hilbert-jittered distributions. It is clear that the slopes for the Niederreiter, Sobol, Hammersley and Halton sequences are the steepest, outperforming the other sequences. The Hilbert-jittered and two-dimensional jittered distributions performed very similarly; not as well as the Niederreiter and Sobol sequences, but better than the random sequences.

Figure 5.10 makes a similar comparison using the circular discrepancy measure. The Niederreiter, Sobol, Hammersley, Halton, Hilbert-jittered and 2D Jittered sequences perform similarly. Figure 5.11 makes a similar comparison using the triangular discrepancy measure. In this case, the 2D jittered and Hilbert-jittered sequences outperform the other low-discrepancy sequences, which perform closer to the random sequence. The random sequence performs poorly, as expected, throughout all three tests.

Least-squares fit lines were computed for each distribution as before, in order to compare the various methods; their slopes are listed in Table 5.3. We see that the Niederreiter, Sobol, Hammersley and Halton sequences perform the closest to the theoretical limits when measuring the discrepancy using rectangles, outperforming the other sequences. However, when measuring discrepancy using quarter circles and triangles, they perform considerably worse. Our Hilbert-jittered and the 2D Jittered methods perform consistently well for all discrepancy measures.

Discussion

When considering rectangular discrepancy on the plane, the Niederreiter, Sobol, Hammersley and Halton sequences perform better than the Hilbert-jittered approach. We also note that the distribution generated by Hilbert-jittered performs similar to the 2D jittering technique as might be 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, Sobol, Hammersley and Halton sequences perform worse in

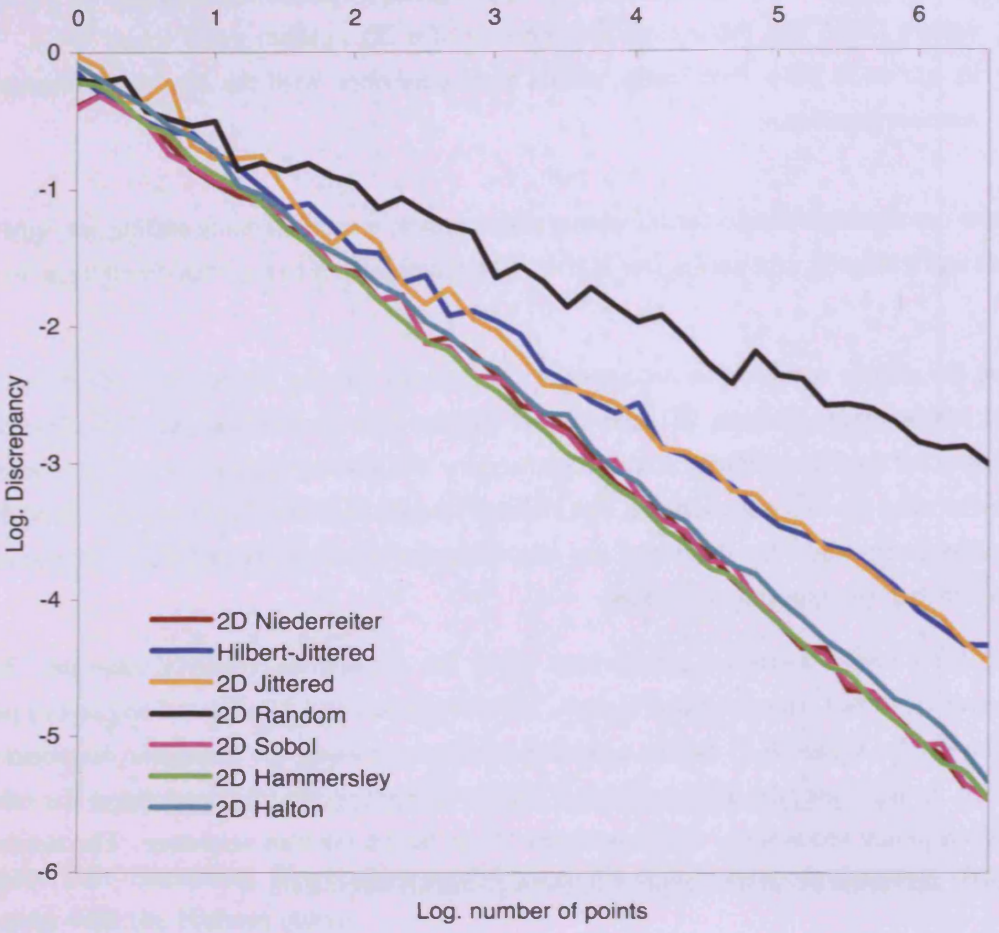


Figure 5.9: Graph of logarithm of discrepancy for various distributions verses logarithm of the number of points measured using rectangular test shapes.

1D Sequence	*Rectangular*	*Circular*	*Triangular*
2D Random	−0.49	−0.50	−0.49
Sobol	−0.90	−0.70	−0.58
Niederreiter	−0.90	−0.69	−0.57
Hammersley	−0.89	−0.71	−0.57
Halton	−0.89	−0.72	−0.59
2D Jittered	−0.75	−0.74	−0.72
Hilbert Jittered	−0.73	−0.73	−0.72

Table 5.3: Gradient of least squares fit discrepancy line for various distributions.

the quarter-circle tests, showing a gradient just below our approach and the 2D jittered

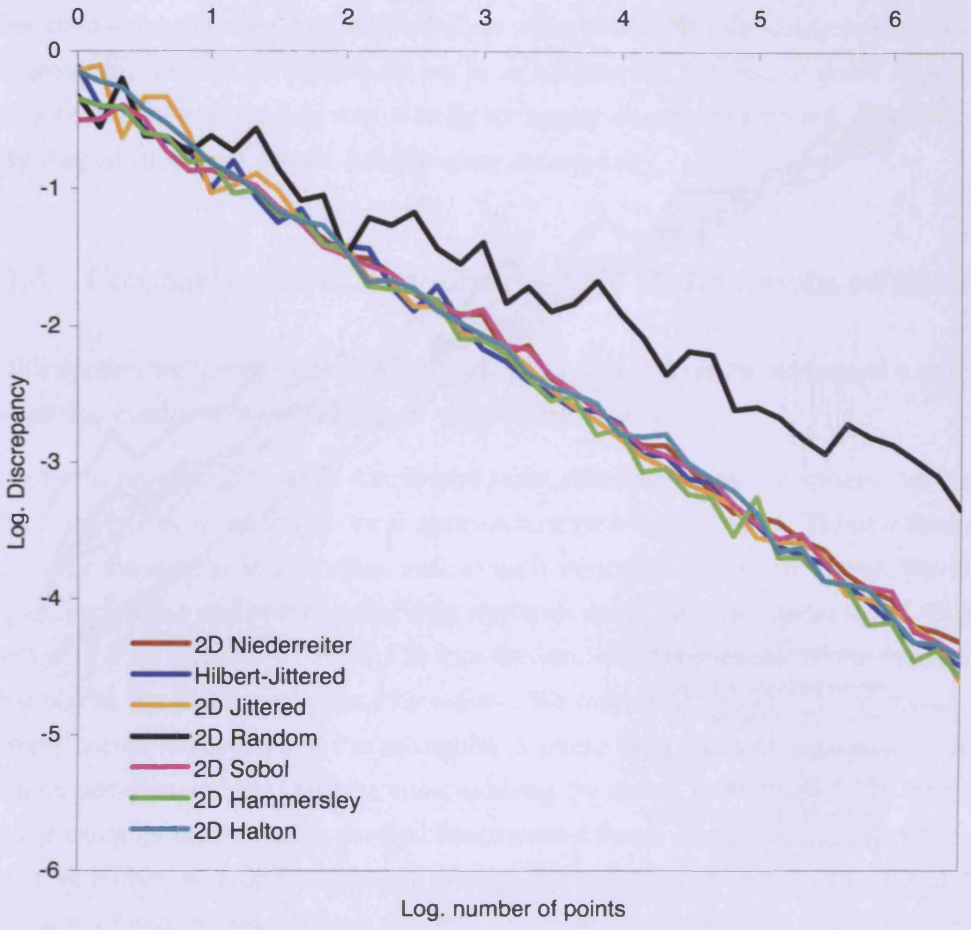


Figure 5.10: Graph of the logairthm of discrepancy for various distributions versus logarithm of the number of points measured using quarter-circle test shapes.

approach, and considerably worse in the triangle tests, 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 Niederreiter, Sobol, Hammersley and Halton methods are *too* specialised to rectangular discrepancy, and the jittered methods are better for all-round usage. Whilst we make no conjecture as to the reason for the inconsistent performance of some of the methods, from these results, it is our conclusion that for general purposes in 2D, the usual measure of star discrepancy using axis-aligned rectangles may be misleading, and that certain methods only produce good results for axis-aligned rectangular boxes. Further to this, in order to better understand the more general case of geometric discrepancy, there is significant scope for further work looking at how the discrepancy of a point set varies with non-convex shapes.

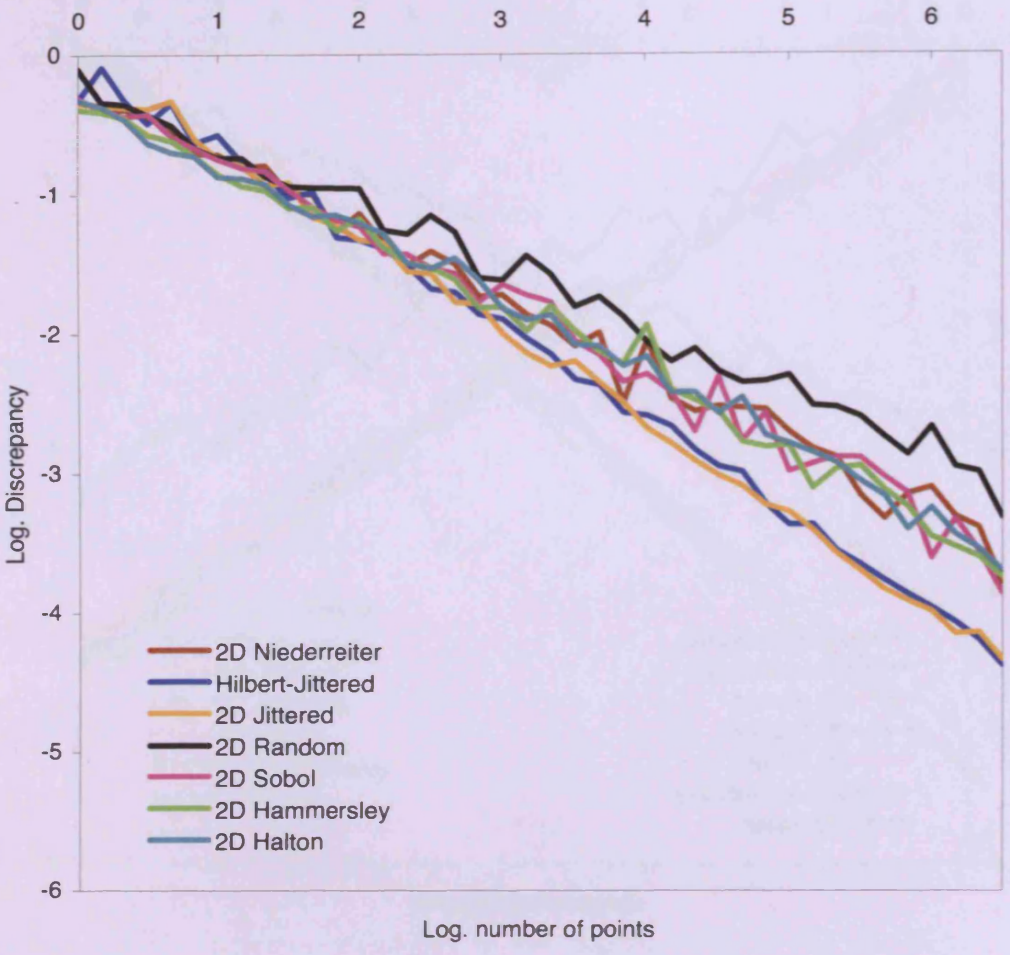


Figure 5.11: Graph of the logarithm of discrepancy for various distributions versus logarithm of the number of points measured using triangular test shapes.

Whilst the point distribution produced by our approach demonstrates a discrepancy very similar to the 2D jittered approach, our approach is far more generally applicable. Using our technique, the number of samples need not be a perfect square (to achieve the current number of cells and density for a square parameter domain), providing full density control over the point distribution, and allowing the same low-discrepancy distributions to be produced on arbitrary surfaces; i.e. the 2D jittered approach is a very special case of our method. The discrepancy of point distributions on the surface of a sphere is discussed in the following section.

Our method is also consistently superior to using a random distribution. In [117] the star discrepancy of the Hilbert curve based sampling approach is briefly investigated. Expanding on these initial results, 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, [117] suggests that the approach is very similar to regular stratified sampling, only with irregularly shaped strata and, hence, has the same discrepancy.

5.1.4 Comparison with Low-discrepancy Distributions on the Sphere

In this section we investigate the discrepancy of point sets on the surface of a unit sphere, comparing results of our algorithm to other distributions.

In order to produce an evenly distributed point distribution on the sphere, we must distribute the points according to local surface area (see Section 4.3). Three techniques are given to compute the local surface area at each vertex of the Hilbert curve, based on triangles, rectangles and infinitesimal area elements using the first fundamental form of the surface: $dA = \sqrt{EG - F^2}\, du\, dv$. In this section, we show results of the change in discrepancy as the number of points increases. We demonstrate two sets of results for the Hilbert curve: Firstly, using the triangular discrete area element constructed using the current, previous and next Hilbert vertices along the curve, and secondly, an infinitesimal area element calculated using the first fundamental form. These two distributions are denoted by Hilbert-jittered-triangle and Hilbert-jittered-fff, and demonstrate the difference in quality of distribution between the least and most exact area approximation methods.

We now demonstrate results for discrepancy on the unit sphere, measured using spherical triangles, comparing our Hilbert-jittered approach to the Niederreiter and Hammersley sequences, and a random distribution (the generation of distributions is explained in Section 5.1.3). Due to the parameterisation of the sphere, sampling uniformly in the parameter domain and then mapping to the sphere surface produces a higher concentration of points at the poles of the sphere, resulting in an overall non-uniform sampling. Using a reparameterisation approach (see Section 4.5), each of the point distributions that are compared to our approach have their polar concentration corrected in order to generate a uniform sampling (see Eq 5.3). Note that the Sobol sequence is not considered here, as the Niederreiter sequence produced consistently better or comparable results on the plane, and it shares a similar construction method. The same applies for the Halton distribution, being represented by the Hammersley distribution on the sphere.

Figure 5.12 shows the variation of discrepancy with point set sizes. The spherical Hammersley sequence, spherical Niederreiter sequence, Hilbert-jittered-fff and Hilbert-jittered-triangle methods all perform similarly well, while the random distribution performs con-

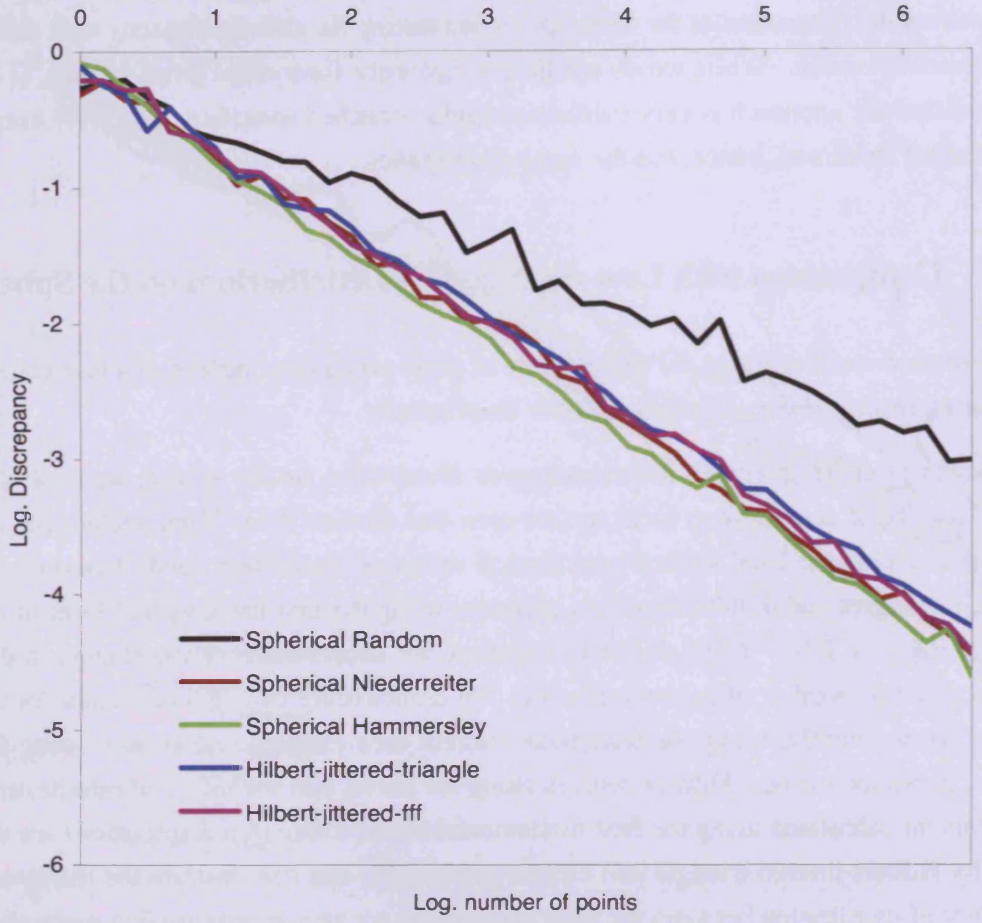


Figure 5.12: Graph of logarithm of spherical discrepancy of various distributions versus the logarithm of the number of points measured using spherical triangles.

siderably worse than the other methods. The gradients of the best fit lines of the various distributions are given in Table 5.4, where we can see that the Hammersley and Niederreiter sequences perform similarly to how the low-discrepancy sequences performed in the plane when measuring discrepancy using quarter-circles. The gradient for the Hilbert-jittered-fff approach is similar to that of these sequences, closely followed by the Hilbert-jittered-triangle approach. The random sequence on the sphere, however, performs poorly.

From the results, we can see that the best fit lines for the spherical Hammersley and Niederreiter point set have very slightly better slopes than our approach. The random distribution on the sphere performed considerably worse than the other three approaches; similar to the 2D random distribution. Hence, using the Hilbert-jittered distribution with our algorithm (see Section 4.3), we can produce a low-discrepancy distribution on the

Sequence	*Spherical*
Spherical Random	−0.49
Spherical Niederreiter	−0.74
Spherical Hammersley	−0.75
Hilbert-jittered-fff	−0.73
Hilbert-jittered-triangle	−0.71

Table 5.4: Gradient of least squares fit line to discrepancy of distributions on the sphere.

unit sphere comparable to other low-discrepancy sequences that have been specifically designed to work with a sphere.

Discussion

From the spherical discrepancy results, we can see that the discrepancy for the spherical Hammersley and Niederreiter points scales slightly better than our approach. The random distribution on the sphere, however, performed considerably worse than the other three approaches; similar to the 2D random distribution. The Hilbert-jittered-fff approach produces slightly better results than the Hilbert-jittered-triangle approach: as expected, a more accurate computation of the surface integrals improves the discrepancy of the output point distribution. The advantage of the first fundamental form area computation (used in the Hilbert-jittered-fff approach) is also likely to increase with more complex surfaces when compared to the more inaccurate measurement given using the triangle area (used in the Hilbert-jittered-triangle approach). So, using the Hilbert-jittered distribution, we can produce a low-discrepancy distribution on the unit sphere comparable to other low-discrepancy sequences that have been specifically designed to work on a sphere. We expect our algorithm, however, to work well for general surfaces (i.e. the sampling approach is independent of the surface), as qualitatively demonstrated in Section 5.2, and can correct, to a certain degree, for severe stretching of the parameter domain (see Section 4.5). In addition it allows the user to adjust the point distributions with a density-function and maintains a a spatially-coherent ordering of points. Throughout the rest of this work, when we refer to our Hilbert-jittered distribution, we use the first fundamental form to compute the surface integral for parametric surfaces.

5.2 Visual Evaluation

In this section, we provide visual results to assess the quality of the point distributions generated by our algorithm. We demonstrate point samplings of various surfaces, using both constant and chosen densities. Whilst not as easily comparable as numerical results, we demonstrate visual results to validate the output of our approach, especially with respect to density control, and also to verify that the good results generalise to arbitrary surfaces. We show points distributed on the unit square, on a Monkey Saddle, an Eight Surface and a Whitney Umbrella, as listed in Table 5.5.

Surface	*Parameterisation*	*Parameter Domain*
Monkey Saddle	$x(u,v) = u$ $y(u,v) = v$ $z(u,v) = u^3 - 3uv^2$	$u, v \in [-1, 1]$
Eight Surface	$x(u,v) = \cos u \sin 2v$ $y(u,v) = \sin u \sin 2v$ $z(u,v) = \sin v$	$u \in [0, 2\pi]$, $v \in [-\pi/2, \pi/2]$
Whitney Umbrella	$x(u,v) = uv$ $y(u,v) = u$ $z(u,v) = v^2$	$u, v \in [-1, 1]$

Table 5.5: Various surface parameterisations used for visual evaluation.

Figure 5.13 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 the results with density $\delta(u,v) = u$, and the image on the right with $\delta(u,v) = u+v$, where u and v are coordinates in the unit square.

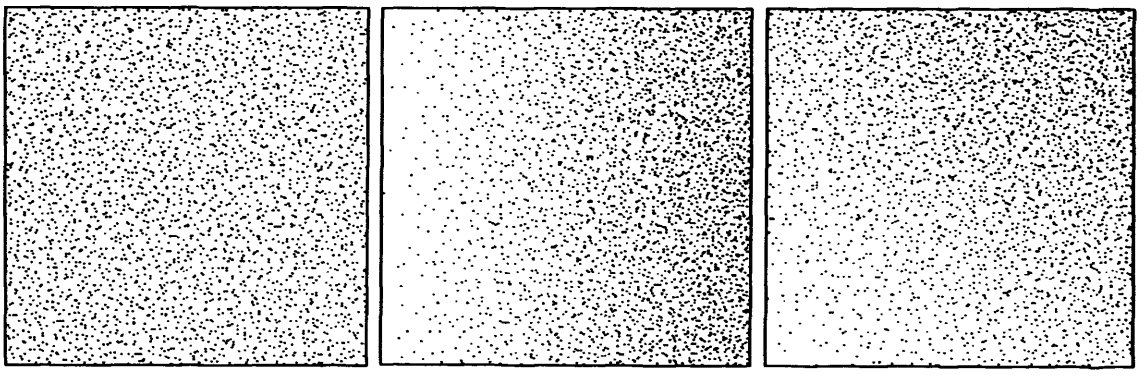


Figure 5.13: Distribution in the unit square: uniform density; density = u; density = $u + v$.

Figures 5.14, 5.15 and 5.16 show images of the Monkey Saddle, the Eight Surface and half of the Whitney Umbrella respectively. Each figure shows a regular grid tessellated with quads, in the parameter domain, mapped onto the surface, the adaptive Hilbert curve mapped onto the surface, and two images demonstrating point distributions generated using our approach. The two point sampled images in each figure show points with uniform density and with density proportional to the Gaussian curvature of the respective surface. The Monkey Saddle is sampled with 3,000 points, the Eight surface with 10,000 points and the Whitney Umbrella with 6,000 points.

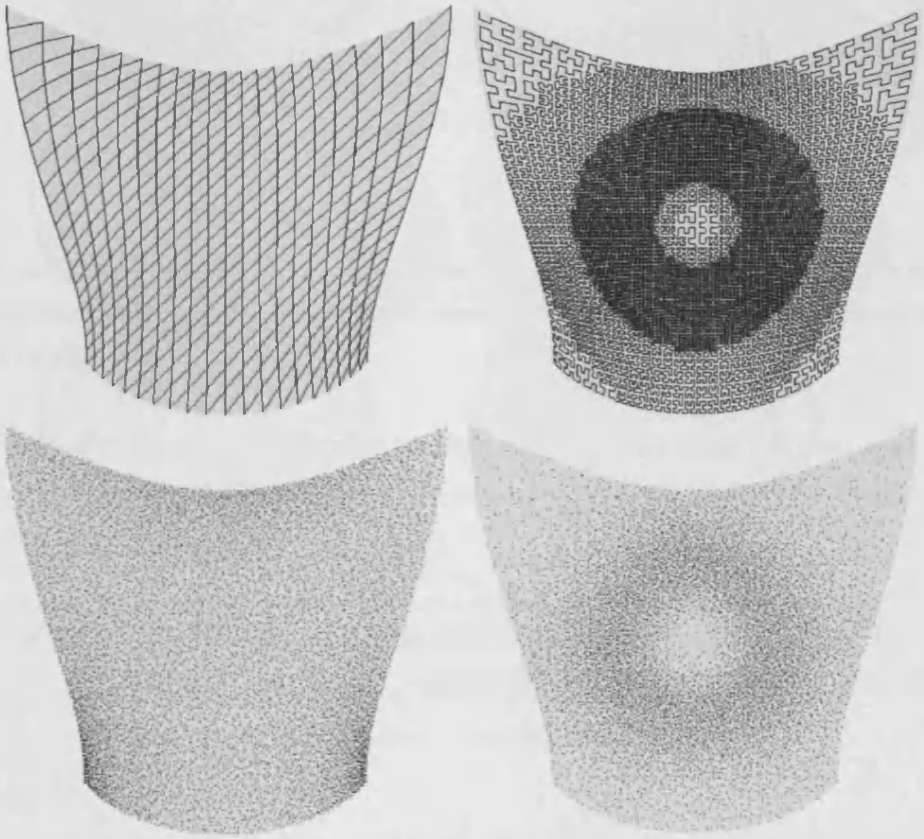


Figure 5.14: The Monkey Saddle: parametric mesh; adaptive Hilbert curve according to curvature; uniform density (3,000 points); curvature controlled density (3,000 points).

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 are not shown to full recursive depth in order to make it easier to see the curve.

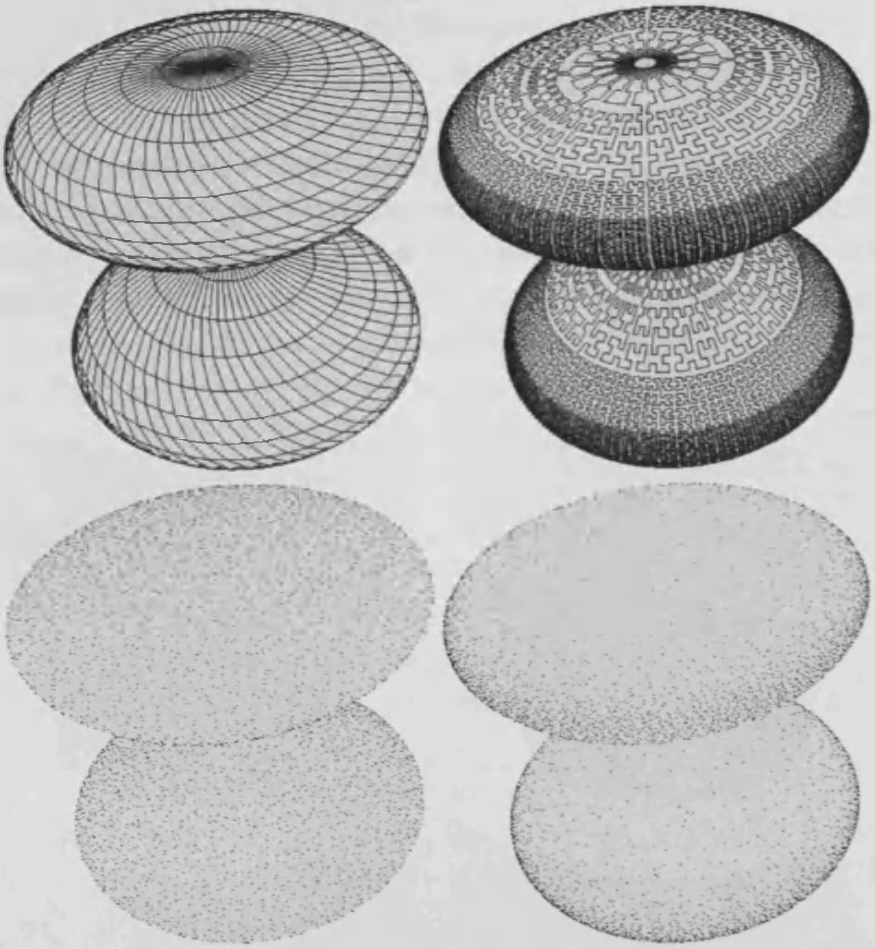


Figure 5.15: The Eight Surface: parametric mesh; adaptive Hilbert curve according to curvature; uniform density (10,000 points); curvature controlled density (10,000 points).

Our Java implementation on a 3Ghz Intel Pentium 4 with 1GB RAM takes about one second to generate the images shown above, with the curve generated adaptively according to surface curvature. For each example shown, the recursive curve depth used was $8 < k < 15$. In this context, where the ratio of areas between the parameter domain, $[0, 1]^2$, and the surface does not vary much, 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

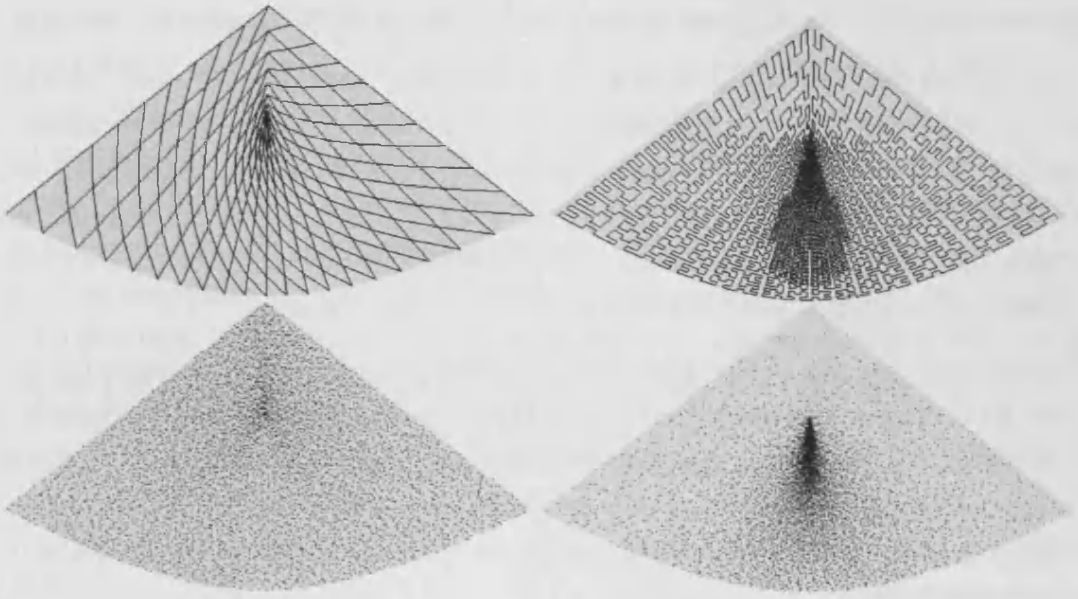


Figure 5.16: Half of the Whitney Umbrella: parametric mesh; adaptive Hilbert curve according to curvature; uniform density (6,000 points); curvature controlled density (6,000 points).

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.

In this section, we have demonstrated visual results showing high-quality point distributions on a selection of complicated parametric surfaces. Our novel approach, unlike earlier surface point distribution algorithms, allows for the high quality sampling of arbitrary parametric surfaces, whilst providing direct sampling density control.

5.3 Blue Noise

In this section, we investigate the property of blue noise for the distributions investigated in Section 5.1.3. Whilst discrepancy gives us a good measure as to the uniform coverage of a distribution, it does not quantify the possible problems caused by the uniform structuring (underlying regular pattern) of the distribution. It is generally accepted that for applications such as surface sampling for visualisation [89] and dithering [120], it is undesirable for a sample distribution to have a structure of its own, interfering with the existing structure or pattern of the surface that is being sampled [120]. Distributions can be constructed, however, that are neither random (resulting in large clumps or holes between

points), nor present an unwanted structure, such as a jittered, or Poisson disc sampling.

Types of noise can be classified by their power spectrum. The frequency power spectrum, listing the intensities of each frequency present, has certain properties. White noise for instance has a uniform power spectrum distribution, while blue noise, defined in the field of visual computing, has minimal spikes, and minimal low-frequency components. The absence of low-frequences means that there is no global density variation. Thus, a sample distribution with blue noise characteristics has a high visual quality.

Regular grid sampling on the other hand does not demonstrate the characteristics of a blue noise distribution, and presents large, regular peaks. Random sampling results in a distribution whose power spectrum has many small spikes, but otherwise a constant energy through all frequencies, resulting in white noise. White noise has a constant energy, even at low frequencies, resulting in an undesirable non-uniform graininess in the distribution [120].

The algorithm described in [120] was used to compute the Radially Averaged Power Spectrum Density (RAPSD) of each sample distribution. The method builds upon Bartlett's approach [10] of computing the Fourier transform of a distribution, and averaging periodograms, which represent the spectral density of a signal. Averaging is done to reduce the variance of the plot, and is performed by computing the Fourier transform of subsets of the sample distribution, squaring their magnitudes, and dividing the total by the sample size. Ulichney [120] builds upon this in order to highlight the degree of radial symmetry in a distribution, by segmenting the distribution into concentric uniform-width rings, which are then averaged in a similar manner to Bartlett's original approach. The assumption made is that the distribution looks the same everywhere, justifying the averaging.

5.3.1 Results

In this section, the RAPSD is plotted as the power spectrum against the radial frequency, and is shown for random, Hilbert-Jittered, Niederreiter and Hammersley sample distributions. The Niederreiter distribution is again used as a proxy for the Sobol distribution, and the Hammersley sequence for the Halton sequence. For each distribution tested, we compute $50,000$ sample points in the unit square. For the probabilistic random and Hilbert-jittered sampling methods, the graph is averaged for 10 distributions. For the Niederreiter and Hammersley deterministic methods, results are shown for just a single sample distribution, as the distribution produced for $50,000$ points will always be identical.

Figure 5.17 shows the RAPSD for a 2D random distribution. The graph shows that it is 'white noise', with a slight peak at a very low radial-frequency—possibly due to

the pseudo-randomness of the generated points. Figure 5.18 shows the RAPSD for the Hilbert-jittered distribution. The distribution performs well with respect to the blue noise criterion, the graph showing minimal low radial-frequency noise, and a constant medium and high radial-frequency noise. Figure 5.19 and 5.20 show the RAPSD for the Niederreiter and Hammersley point distributions. Both distributions perform poorly with respect to the blue noise criterion; whilst showing minimal low radial-frequency noise (no randomness), the large peaks show a large amount of regular structure, consistent with a regular grid distribution. Note that in the Hammersley distribution graph, the signal at the radial frequency of 0.23 actually peaks at a power of 12, off the scale of the graph.

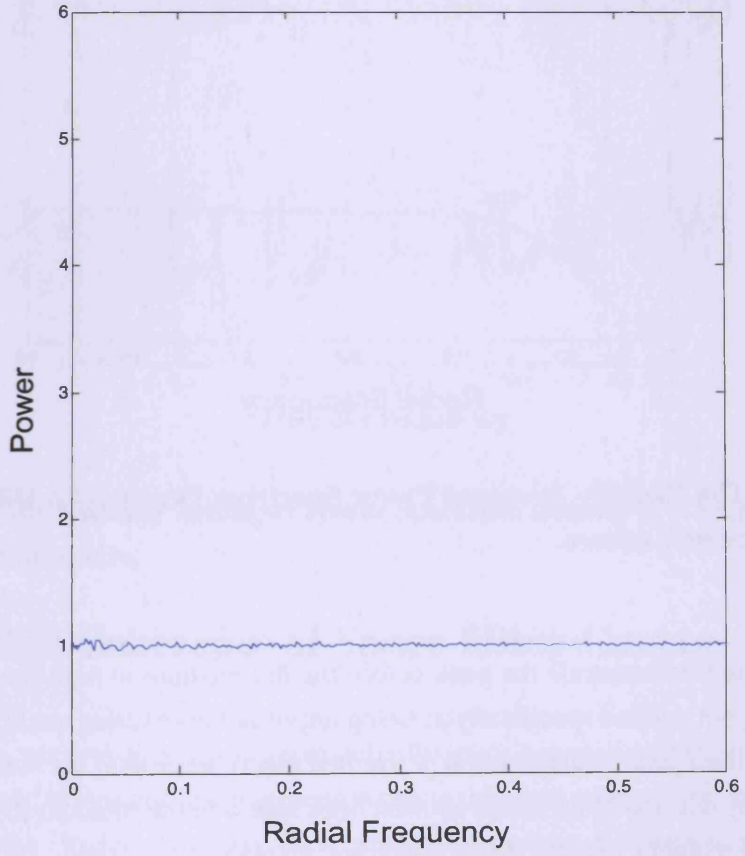


Figure 5.17: The Radially Averaged Power Spectrum Density of a random sampling of the unit square.

5.3.2 Discussion

From the results, we can see that the Hilbert-jittered distribution demonstrates a power spectrum close to the ideal blue noise power spectrum defined for visual computing.

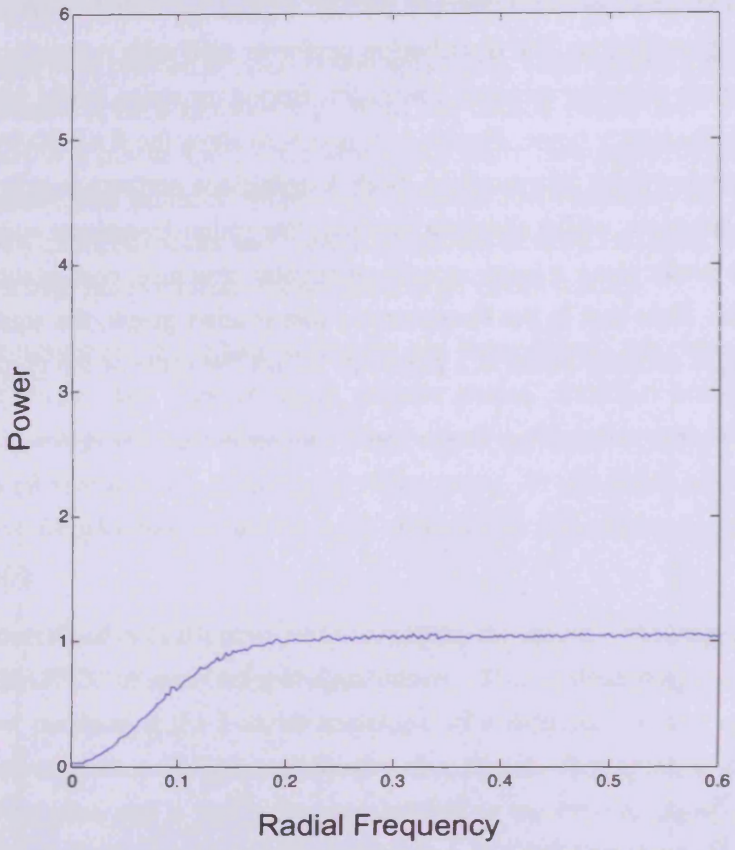


Figure 5.18: The Radially Averaged Power Spectrum Density of a Hilbert-Jittered sampling of the unit square.

Whilst it does not demonstrate the peak before the flat medium to high frequency shown in [120], this is not defined specifically as being important in ensuring quality of the distribution. The Hilbert-jittered distribution performed much better than the two deterministic low-discrepancy distributions, which showed high spikes in the medium to high frequencies, indicating a highly structured regularity, and an underlying grid.

Whilst fulfilling the blue noise criterion was not one of the core aims of our work, it is useful as an additional measure as it allows us to numerically assess the visual quality of a distribution, and has been used in the literature to verify dithering qualities and surface samplings. Combined with the discrepancy measure, we now have information not only on how well the points are distributed with respect to area coverage, but also on how visually appealing they are for the representation of surfaces and patterns.

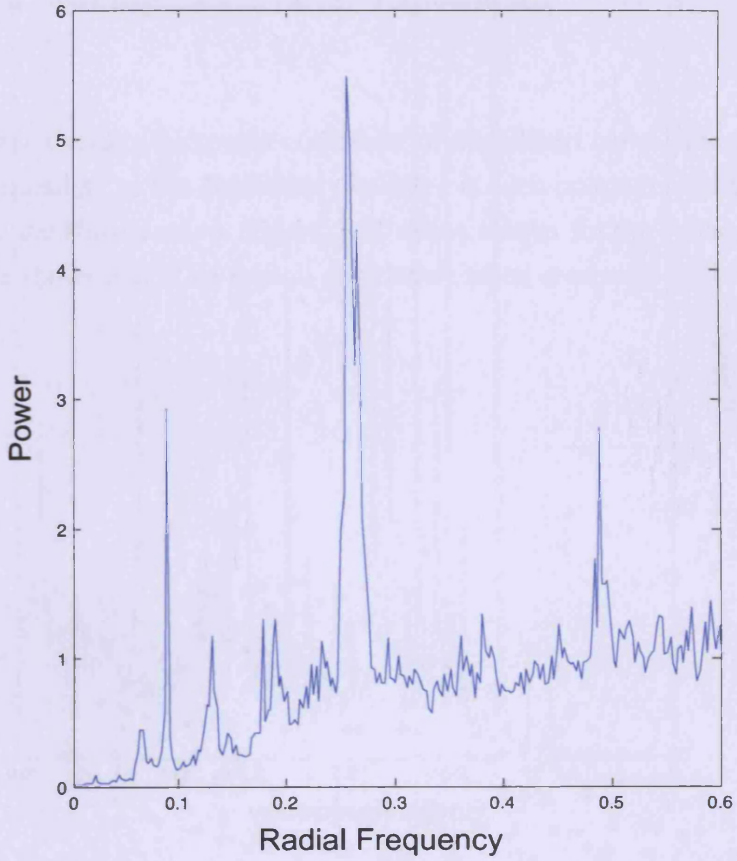


Figure 5.19: The Radially Averaged Power Spectrum Density of a Niederreiter sampling of the unit square.

5.4 Spatial Coherence of Space-filling Curves

In this section, we investigate how accurately distances between points on a space-filling curve correspond to straight-line distances between the same points in $\mathbb{E}^2$. In Section 2.2.3, we looked at the Hölder continuity for a space-filling curve, whereby a constant, representing the upper bound of the maximum difference between distances on a space-filling curve and in Euclidean space in the unit square, is sought. The Hilbert curve has been proven to have the smallest known Hölder constant, smaller than that for the Peano II curve [24].

In this section, we investigate this property of distance similarity, or spatial coherence, experimentally, using discrete approximations of the space-filling curves. Practically, we wish to validate the theoretical properties of Hölder continuity (see Section 2.2.3) on the discrete curves. In this work, good spatial coherence improves the accuracy of fast, ap-

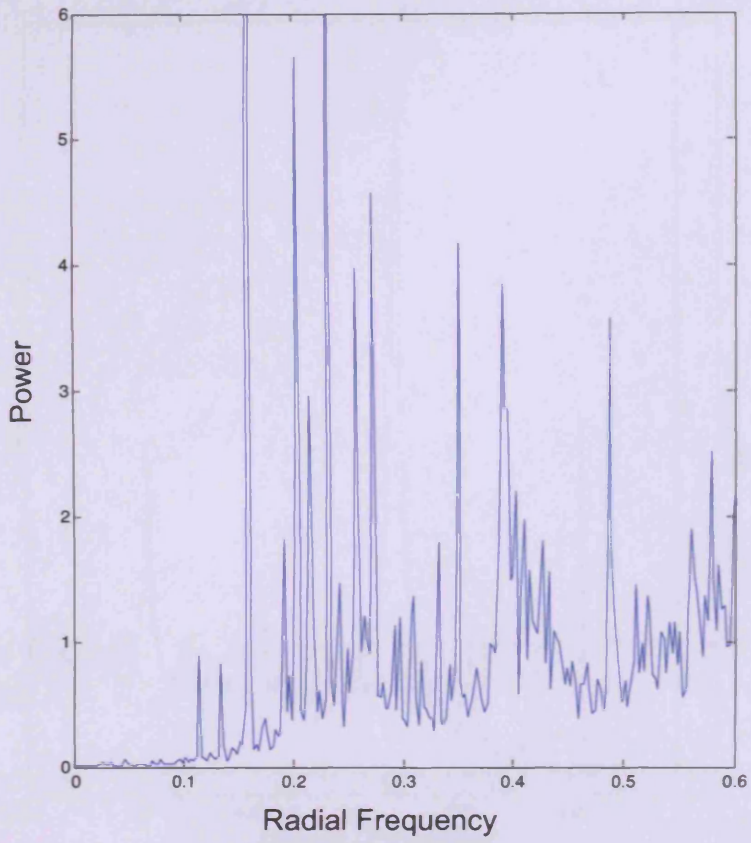


Figure 5.20: The Radially Averaged Power Spectrum Density of a Hammersley sampling of the unit square.

proximate, neighbourhood computation (see Section 7.1.1), and ensures that the structure of the curve does not cause problems with the histogram equalisation step of our sampling algorithm (see Section 2.2.2). For broader application, a good spatial coherence is useful improving compression rates [35] and for vertex caching [16].

For a fixed recursion depth, we generate a space-filling curve, and choose two vertices at random from this curve. We then compute the distance between the two vertices along the curve. For a space-filling curve with L vertices, and a pair of vertices C_i and C_j, the distance along the curve is therefore computed simply using $\|C_i - C_j\|/L$. We then compute the straight-line distance between the two points. This is done for $1,000$ pairs of points, and plotted as a scatter graph to investigate how well the two different distances correlate.

We test only the Hilbert and Peano II curves in this section, as they are the only space-filling curves used in our algorithm due to the reasons explained in Section 2.2.2. The Hilbert curve was generated to a depth of 12, resulting in 1.6×10^7 vertices. The Peano II

curve generated used a depth of 9, resulting in 10^7 vertices.

Figure 5.21 shows results for spatial coherence of the Hilbert curve. The results show a fairly high correlation; as the Euclidean distance between points increases, so does the distance along the Hilbert curve. Figure 5.22 shows results for the Peano II curve. The Peano II curve shows less of an overall correlation when compared to the Hilbert curve coherence.

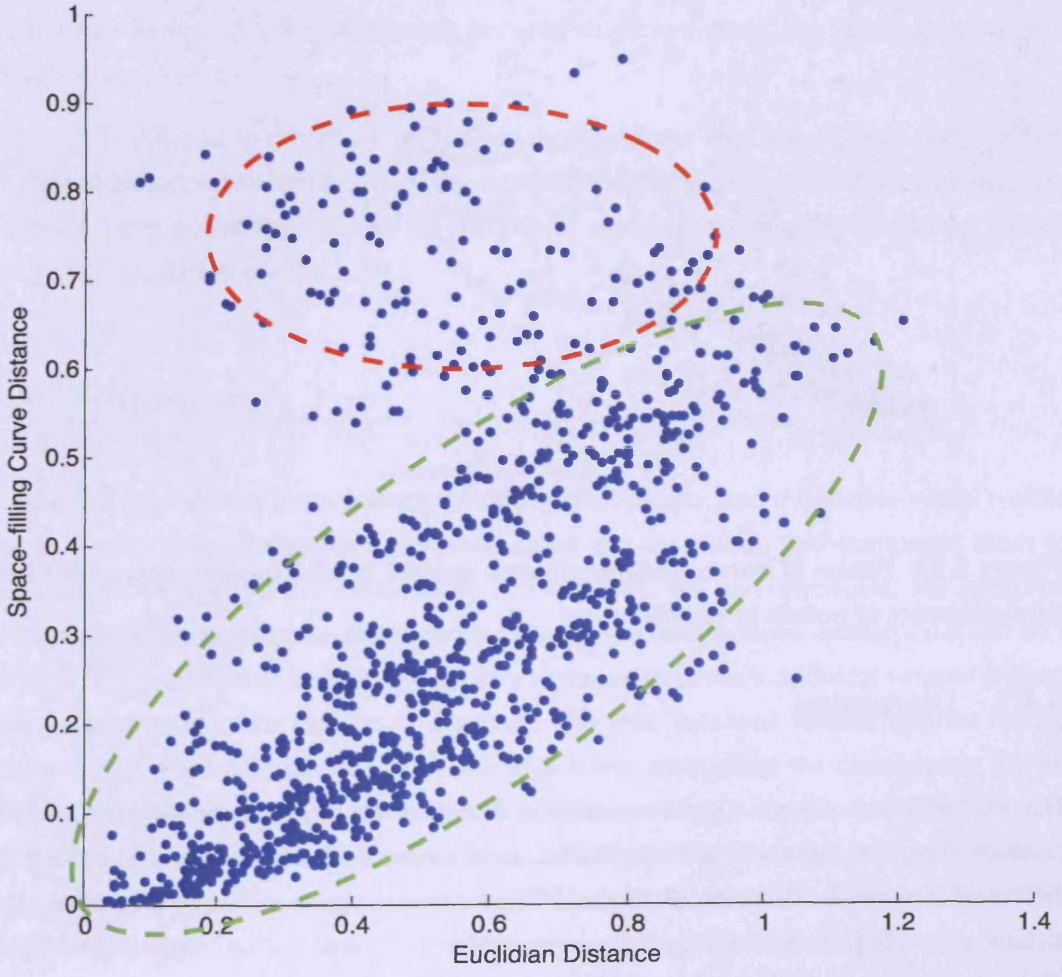


Figure 5.21: Hilbert curve distance shown against Euclidean distance, plotted for random pairs of points in the plane.

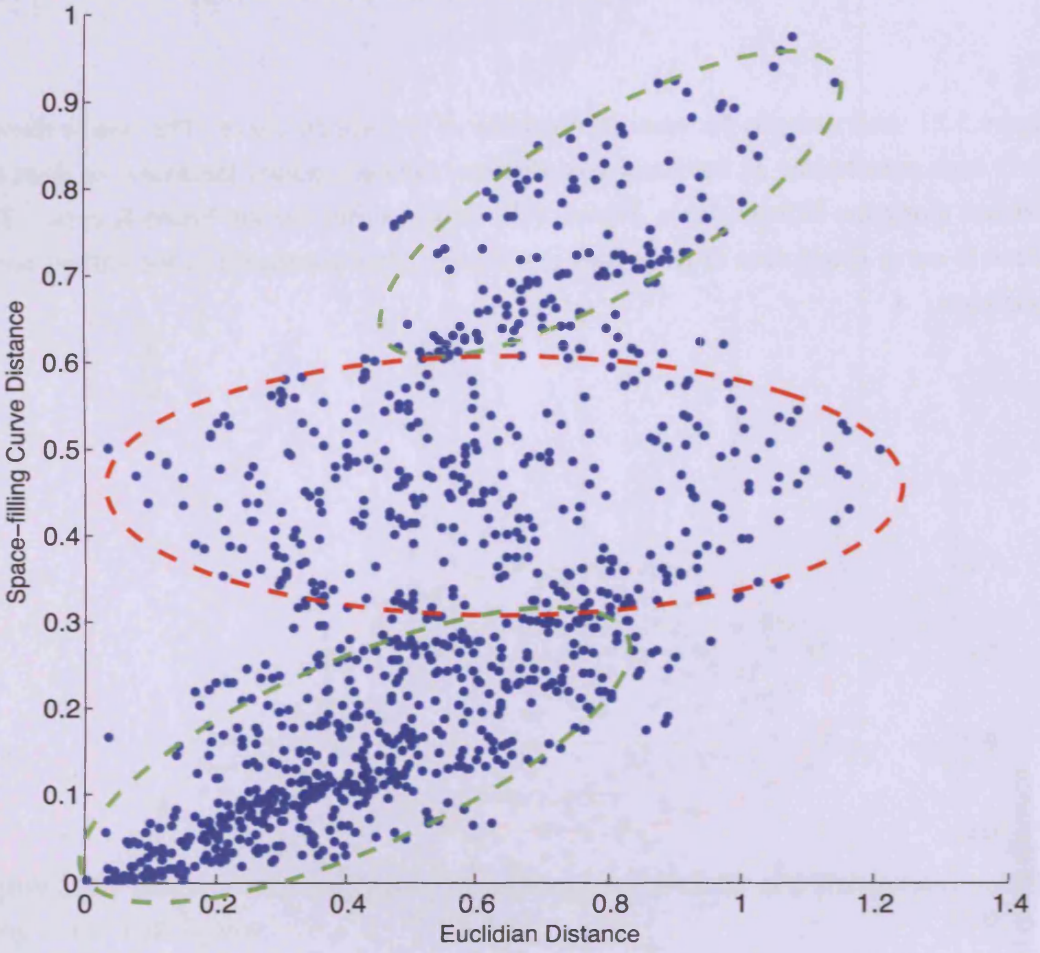


Figure 5.22: Peano II curve distance shown against Euclidean distance, plotted for random pairs of points in the plane.

5.4.1 Discussion

The Hilbert curve shows a good correlation between the straight-line distance and the distance along the curve as the straight-line distance increases. The same can be said for the correlation as the distance along the Hilbert curve increases. We note that for most distances (see Figure 5.21, circled in green), up to about 0.65 of the length of the curve, we have a constant gradient of approximately 0.7. However, the more extreme distances along the Hilbert curve (circled in red), from about 0.65 of the length of the curve upwards, we have a lack of correlation between the space-filling curve distance and the straight-line distance. This problem occurs when the winding course of the curve inevitably ends up almost looping back on itself to cover the whole square, meaning that some points that

are very close in Euclidean space are not close on the curve. In Figure 5.23, we demonstrate spatial coherence on the Hilbert curve. The top row of images shows examples of good spatial coherence. The bottom image shows poor spatial coherence, highlighting the problems that can occur with the Hilbert curve (see Figure 5.21, circled in red).

The Peano II curve demonstrates quite different behaviour. Unlike the Hilbert curve, small distances and large distances (see Figure 5.22, circled in green) along the curve, independently, correlate well to straight-line distances, with a gradient of about 0.5. This is due to the fact that for the approximation of the Peano II curve, the maximum inaccuracy between the Peano II curve distance and Euclidean distance is smaller because of the particular winding of the curve. However, it is clear from the graph that for medium curve lengths (see Figure 5.22, circled in red), especially around 0.5 of the length of the curve, we have an extremely poor correlation between distances along the Peano II curve and straight-line distances.

Whilst it is difficult to draw any strong conclusions from this, the Hilbert curve shows an overall better correlation between Euclidean and space-filling curve distances than the Peano II curve, corroborating with the theoretical analysis provided by the Hölder Continuity measure (see Section 2.2.3).

5.5 Summary

In this chapter, we have given quantitative numerical results, and qualitative visual results for the concerning distributions produced using our algorithm, and compared them to results from well known low-discrepancy distributions. We first introduced our approach to numerically computing the discrepancy of our point distributions on the plane and on a surface. We generalised the star discrepancy measure to include different sample shapes, and measurement on the surface of a sphere. We then assessed various options for the generation of point distributions using our algorithm, computing the discrepancy for the Hilbert curve and the Peano II curve sampled with various 1D sequences. Whilst the well known low-discrepancy methods perform well for the planar rectangular sampling shape, they performed worse for the quarter circle and triangle shapes, and the on the sphere. This lack of consistency makes them less useful for rendering applications [39]. Our Hilbert curve sampling method however, showed consistently good results throughout all of the discrepancy tests. These results indicate that the star discrepancy alone is not sufficient as an overall measure to determine the uniformity of sampling in 2D. Providing a more thorough analysis of discrepancy on general surfaces is a more complex problem, however (see Section 5.1.1). In addition to this, it is also desirable to have more metrics to compare

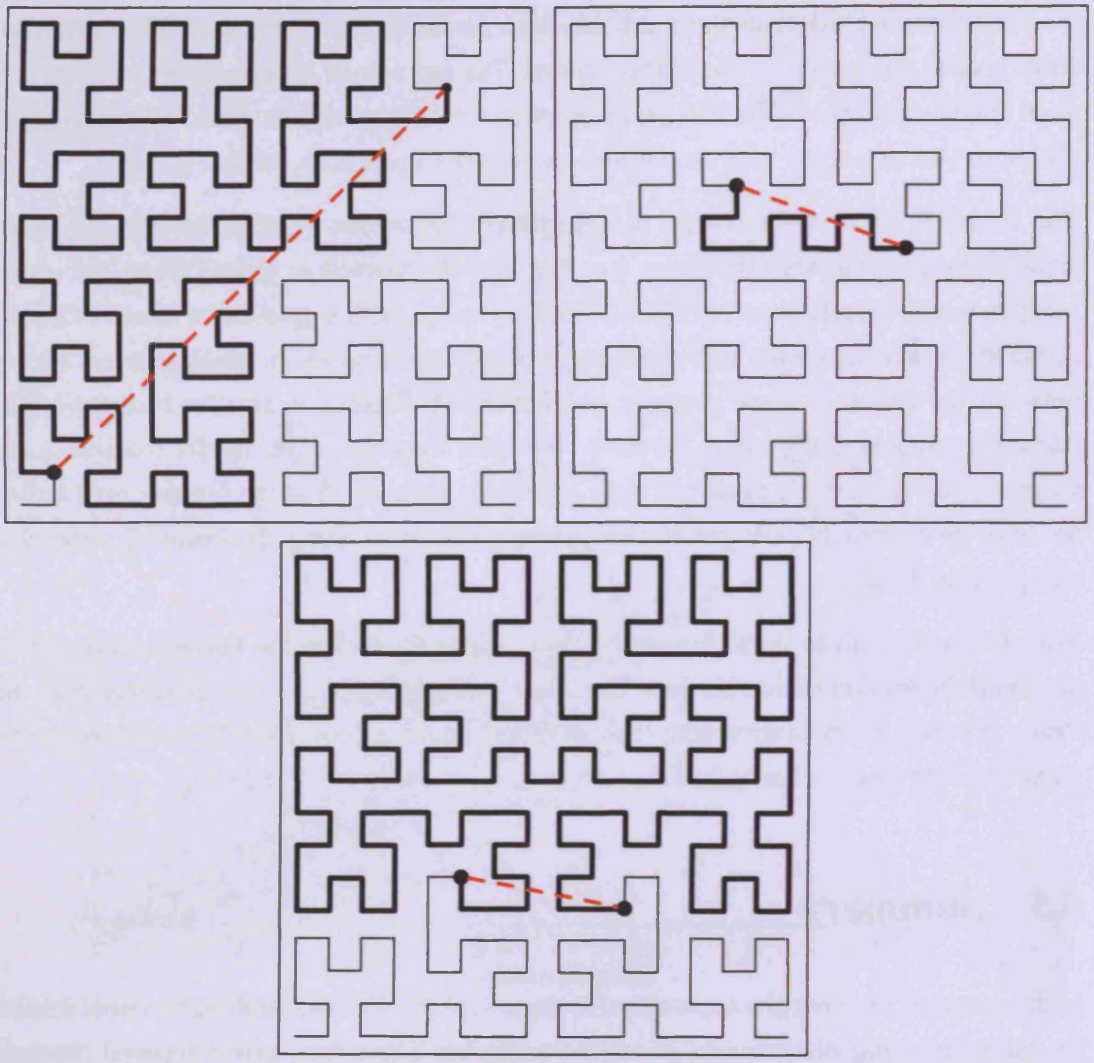


Figure 5.23: Relative straight-line (dashed red line) and space-filling curve path (thick black line) distances for pairs of points on the Hilbert curve. Top two images show good spatial coherence. Bottom image shows poor spatial coherence.

results for various distributions, both generic, and tailored specifically for rendering and engineering applications. Our conclusions from these discrepancy experiments are that the Hilbert-jittered-fff sampling is the best choice for our own method, and performs well throughout when compared to other sampling methods.

We further analysed the visual results produced by our algorithm, demonstrating high quality, density-controlled results on the plane and on various parametric surfaces. In particular, this testing provides evidence that our approach works on general surfaces where other methods would have problems.

We then investigated whether the sample distributions produced by our algorithm fulfilled the the blue noise criterion. We analysed the radially averaged power spectrum density of distributions produced using our algorithm and the comparison distributions already introduced. Fulfilling this criteria indicates that a distribution is uniformly sampled but does not add its own structure to what is being sampled. The Hilbert-jittered-fff distribution showed good results, indicating that it fulfilled the blue noise criteria. The well known low-discrepancy distributions tested demonstrated a very highly structured sampling, showing a power spectrum very different to that of blue noise. The fulfilment of the blue noise criterion indicates that distributions produced by our method have no global density variation, and no grid like structure, making them good for visual representation.

Finally, we investigated the spatial coherence of the two space-filling curves used in this chapter, the Hilbert and Peano II curves. The Hilbert curve showed a better overall correlation between distances along the space-filling curve and straight-line distances, useful for vertex indexing, compression and fast neighbourhood look-ups.

Chapter 6

Mesh Sampling

Polygonal meshes have become a common and simple shape representation for storing, manipulating and rendering 3D models. In order to utilise this representation for sampling, we propose a technique, based on the parametric surface point-sampling algorithm described in Chapter 4, to resample arbitrary polygonal meshes. The emphasis is on the sampling 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 mesh surface that has low discrepancy with respect to a user-specified density function, and demonstrates a visually appealing 'blue noise' behaviour.

Our mesh sampling technique is described in Section 6.1. Taking a triangle mesh as input, we first cut it (if necessary) such that the result has the topology of a disk. Then we parameterise it over $[0,1]^2$. We 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 with a correlated uniformly distributed set of points according to a user-specified density (see Section 2.1). We then sample points along the curve taking into account the local geometry of the mesh.

The pre-processing steps of cutting and parameterising the mesh, and the generation of the Hilbert curve in $[0,1]^2$ are time-consuming (taking a few seconds). However, after these pre-processing steps, generating a new set of point samples on the mesh is extremely fast (less than 10 milliseconds for all meshes tested, see Table 6.2), allowing us to resample the surface at interactive rates in response to user required changes of the density, e.g. as a virtual camera zooms in or out. If the density changes significantly, we must subdivide the curve locally (see Sect. 6.2.4) in a background process. For larger density changes everywhere, it may be cheaper to increase the recursion depth of the Hilbert curve globally, requiring about a tenth of the time taken to compute the initial curve as no area measurements are required (see Section 7.1.3).

Experimental results are given to demonstrate the distributions produced using our technique in Section 6.2, and to compare them to other approaches. We first investigate a

mesh discrepancy measure in order to validate the distribution that is being generated on the mesh. We show that point distributions produced by our algorithm have low discrepancy consistent with results for parametric surfaces (see Section 5.1.4). We then compare the quality of meshes produced by our remeshing implementation to other competitive techniques by measuring the Hausdorff distance using Metro [28], showing results competitive with one of the most popular remeshing techniques.

In Chapter 7, we apply our mesh sampling algorithm in three ways: as a sample generation method for point-based graphics, remeshing and engineering prototype painting. We also demonstrate the flexibility of our technique and the advantages of using the Hilbert curve as the underlying technique for resampling, demonstrating real-time resampling for levels of detail (LoD) and view dependent rendering (VDR) (see Section 7.1.3).

6.1 Sampling Meshes at Interactive Rates

In this section, we describe our approach for sampling polygonal meshes (see Figure 6.1). It is fundamentally the same as the algorithm described in Chapter 4, although we also need methods to cut and parameterise an input mesh surface, and compute discrete area and curvature properties of the surface and produce mappings between the parameter domain and $\mathbb{R}^3$. The algorithm takes, as input, an arbitrary polygonal mesh M_s along with a user-defined density function, $\delta : M_s \rightarrow \mathbb{R}_0^+$. The mesh is first cut in order to turn it into a topological disk as a prerequisite for parameterisation over $[0,1]^2$ (see Section 6.1.1). In order to compute the output point distribution P efficiently, we generate an adaptive Hilbert space-filling curve in the parameter domain, $[0,1]^2$, and map it onto the mesh M_s (see Section 6.1.2). The curve simplifies placement of sample points on the surface, and is generated adaptively to compensate for distortion introduced by the parameterisation of the surface, and its pre-computation allows for resampling at interactive rates. Integrals to allow the accurate sampling of the Hilbert curve are then computed (see Section 6.1.3). The curve is sampled, the output of which is a set of high-quality uniformly distributed points P on the mesh M_s.

6.1.1 Mesh Cutting and Parameterisation

We now discuss how we approach the problem of sampling an arbitrary genus mesh. Our sampling algorithm relies on a unit square parameterisation, to which our Hilbert curve sampling algorithm is applied. Therefore, we must first compute a planar parameterisation for the input mesh $f : M_p \subset [0,1]^2 \rightarrow M_s \subset \mathbb{R}^3$. However, a $[0,1]^2$ parameter domain

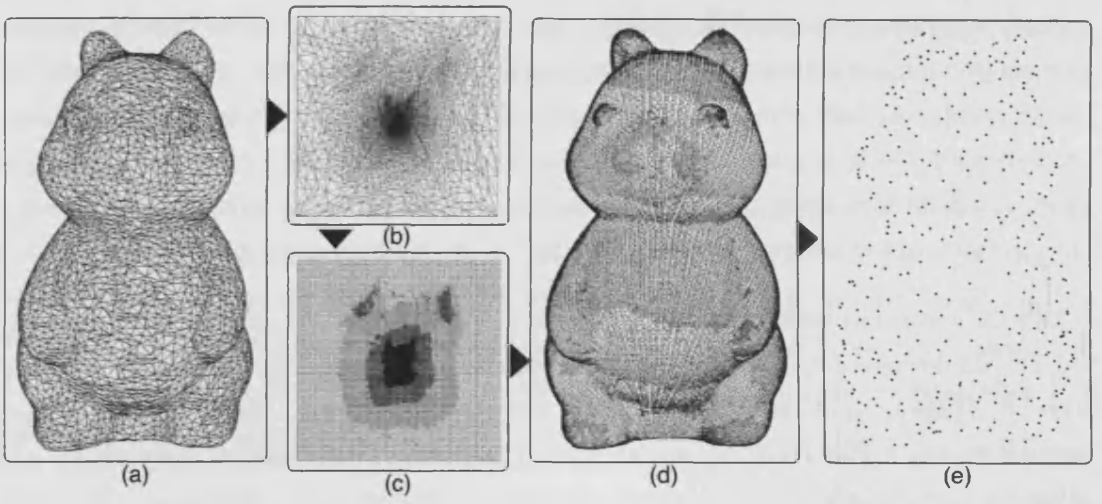


Figure 6.1: Mesh sampling pipline: (a) The input mesh is first cut into a topological disk and (b) parameterised over the unit square. The parameterised mesh is (c) sampled with an adaptive Hilbert curve, which is (d) mapped onto the original mesh surface, and (e) sampled, giving density-controlled low-discrepancy distribution.

cannot, homeomorphically, be mapped onto a surface that is not a topological disk. Yet, in order to make our algorithm as broad as possible in its application, we allow that the input mesh M_s be of arbitrary genus. Therefore, in order to compute f for an arbitrary genus mesh, the mesh M_s must either be cut into a disk topology (i.e. have a single, definite, boundary), or must be cut into a set of patches which can then be independently parameterised. Due to the adaptive nature of our Hilbert curve generation compensating for the area differential between the surface and the plane, we are predominantly concerned with maintaining angle conformity during the mapping f, which we can achieve without local patch parameterisation. Thus, we employ the approach of cutting the mesh into a topological disk followed by a global parameterisation.

A cutting method, employed in [112], creates a topological disk by cutting toward extremities of the mesh using a curvature-based local extremity detection. However, this approach does not necessarily find global extremities. Gu [46] employs a more advanced approach, which uses an iterative cut-parameterisation approach. A cut is made, and the mesh is parameterised. Global extremities are then found by measuring stretch across the parameterised mesh. The surface cut is then directed into the high-stretch extremity areas of the mesh. This process is iterated until a global minimum stretch cut is found. This allows for a stretch-minimising global parameterisation, but is computationally very expensive. As stretch minimisation is not required due to the adaptive sampling nature of our Hilbert curve, we employ the naive cutting approach described in [107], which

simply cuts the mesh into disk topology, and is computationally much faster, not requiring an iterative optimisation. Note however, that this naive method may suffer from the same problems noted with parametric surfaces if an extreme stretch is involved in the parameterisation (see Section 4.5). The output of this algorithm is a genus-zero mesh with duplicated vertices along its border (or seam); a necessary result in order to avoid a gap in the mesh that would prevent sampling.

Once the mesh has been cut into a topological disk, it must then be parameterised over $[0,1]^2$. As discussed in Section 3.2.3, there are two main approaches to surface parameterisation, equiareal (area preserving), and conformal (angle preserving). To avoid artifacts caused by angle distortion, we desire a conformal parameterisation of the surface; area distortion can be corrected using our adaptive Hilbert curve (see Section 6.1.2). In the discrete case of a mesh M_s, ensuring that the angles between edges are consistent when mapped between M_s and M_p for all polygons in the mesh ensures a conformal mapping. The conformal mesh parametrisation algorithm described in [42] is used to generate such a mapping $f : M_p \subset [0,1]^2 \rightarrow \mathbb{R}^3$ from a parameter mesh $M_p \in [0,1]^2$ to the surface mesh M_s. The output of the mesh cutting and parameterisation stages of this algorithm is a genus-zero mesh that fills $[0,1]^2$ (i.e. it has a square boundary).

6.1.2 Mapping the Hilbert Curve onto the Surface

Once the mesh M_s has been cut and parameterised, producing a planar disk-topology mesh M_p, we generate an adaptive Hilbert curve in $[0,1]^2$, filling the area covered by M_p. The Hilbert curve is used to reduce the complexity of surface sampling in $\mathbb{R}^3$ as before. It is generated using the quad-tree algorithm described in 4.2, the output of which is a discrete, adaptive Hilbert curve, represented using a set of vertices $\{h_l : l = 1,\ldots,L\} \subset [0,1]^2$. The adaptive construction of the curve is important as the parameterisation does not guarantee local area preservation with respect to the surface mesh M_s. Figure 4.5 demonstrates the advantages of using an adaptive Hilbert curve over a uniform Hilbert curve when using the global parameterisation approach on a squirrel mesh (see Section 6.1.1).

When adaptively generating the Hilbert curve, locally, the number of curve vertices required is some constant ω times the ratio $\int_U \delta\, dA / \int_{M_s} \delta\, dA$, for every subset U of the mesh M_s. As discussed in Section 4.2.1, in practice, $\omega \geq 10$ provides a sufficient density of curve vertices with respect to the density δ. In order 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 T in M_s as $N_T = \int_T \delta \, dA / \int_{M_s} \delta dA$. We thus require O_T Hilbert vertices inside a surface triangle with $O_T \geq \omega N_T$. 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 of the Hilbert curve at any given level. In practice, we require that $O_Q \geq O_T / (\text{area}(f^{-1}(T)) / \text{area}(Q))$, where O_Q is the number of vertices a quad Q of the quad tree, and $f^{-1}(T)$ is the triangle T mapped to the parameter mesh O_p (where f^{-1} is the inverse of the parameterisation). Note that here we generalise f to operate on sets in the sense that $f(Q) = \{f(x) : x \in Q\}$.

Once we have generated the Hilbert curve, we must then map it to the surface using the parameterisation f. If $M_p = \{t_l : l = 1, \ldots, n\}$ and $M_s = \{T_l : l = 1, \ldots, n\}$, where t_l is a triangle in $\mathbb{R}^2$, and T_l a triangle in $\mathbb{R}^3$, there is a bijective mapping $f : M_p \leftrightarrow M_s$ which can be used to map of the curve from $\mathbb{R}^2$ to $\mathbb{R}^3$. Note therefore that $f : t_l \leftrightarrow T_l$. Thus, the function to map a Hilbert vertex from M_p to M_s simply becomes $f(h_l) = H_l$ for $h_l \in t_l$, $H_l \in T_l$. In order to compute this mapping between the parameterised mesh M_p and and the surface mesh M_s, we must convert the (u, v) co-ordinates of the vertex h_l in M_p, to local barycentric co-ordinates within a triangle t_l. Knowing that t_l maps directly to a single surface triangle T_l, due to the bijective mapping between the sets of triangles, we then use the same barycentric co-ordinates in T_l for H_l. These must then be converted into (x, y, z) co-ordinates on M_s, giving us the final vertex position. In order to do this, however, we must first find the triangle on the plane that contains the vertex h_l.

To find the triangle containing a particular vertex, the algorithm must potentially deal with millions of vertices, and tens or hundreds of thousands of triangles (see Section 6.2.3). Thus, a fast approach must be employed to find these containing triangles. However, even a spatial data structure such as a kD-tree (or some other BSP tree variant), quad-tree or R-tree, will not provide fast results for data sets of this size. Instead, we solve this problem by using a simple GPU method. We first rasterise the planar triangulated mesh M_p as a high resolution square texture in the frame-buffer. Due to the adaptive nature of the curve, ideally, the texture resolution should be a square of 4^g pixels, where g is the upper bound of the recursive depth of the generated Hilbert curve. If the frame buffer is not large enough, then the maximum size should be assigned, dependent on the GPU, providing the best resolution possible. When rasterising M_p, each triangle t_l is uniquely coloured, with the colour being a hash key for the triangle index l. If both the width and height of the square texture is $d = \sqrt{4^g}$, for a vertex h_l we then sample the colour at the frame-buffer location $(\lfloor ud \rfloor, \lfloor vd \rfloor)$. The triangle index is then retrieved using the hash-table. Optionally, the algorithm can then perform a containment check on the found triangle to

ensure the accuracy of the process (in particular if the frame buffer is not large enough). If the point is not contained within the triangle, the algorithm will then check all of its one-ring neighbours, and so on outwards until the correct triangle is found. A possible improvement to this approach would be to divide the Hilbert curve up into patches based on the local recursion level and apply the GPU method to each patch. This GPU process is extremely fast, as demonstrated in Table 6.2.

Once we have found the containing triangle t_l for a Hilbert vertex h_l, we must then convert the (u, v) parameters for the vertex to a local co-ordinate system for the triangle. Barycentric co-ordinates are a method of describing a point h_l within a triangle $A\hat{B}C$ using a local co-ordinate system for that triangle, giving us a barycentric point p (see Figure 6.2). To compute p, we simply compute the intersection of the line that passes through A and h_l with the line BC, giving a point A'. The distance $\|A' - h_l\|$ gives a mass, or weight, m_A for the vertex A. The distance $\|B - A'\|$ then gives the mass m_C for the triangle vertex C, and $\|C - A'\|$ the mass m_B for the triangle vertex B.

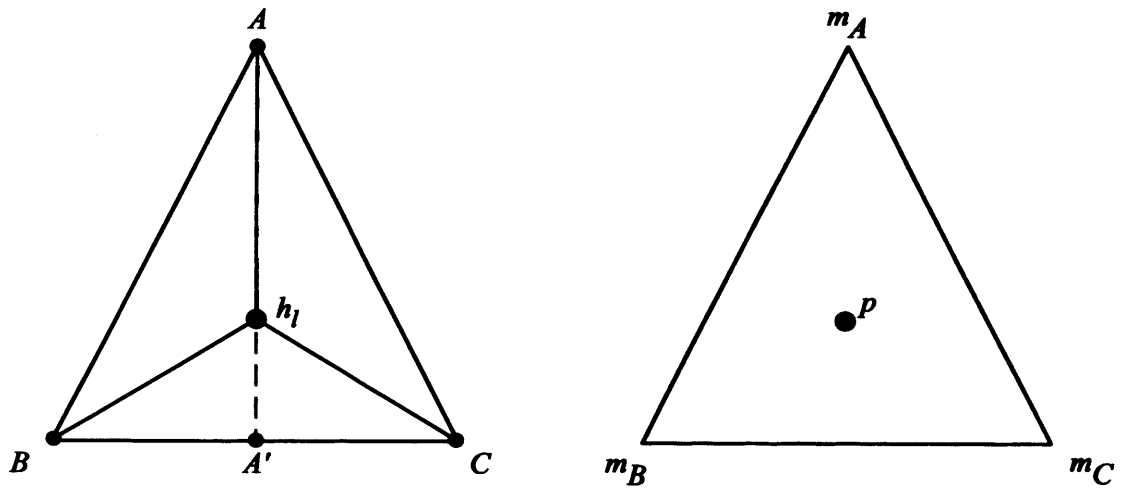


Figure 6.2: Barycentric co-ordinates in a triangle.

We now have a point p, described by three masses at A, B and C: m_A, m_B and m_C, the ratio between which defines the point p. The position of p can be thought of as being relative to the gravitational effect of the three masses on the point. This gives us a trilinear co-ordinate representation which is invariant to the embedding space of $A\hat{B}C$. To convert p back to Cartesian (u, v) or (x, y, z) co-ordinates, assuming a normalised total weight of $m_A + m_B + m_C = 1$, we must first compute the point A', which lies on the line segment BC. A' can be calculated by

$$A' = \left(B\frac{m_B}{m_B + m_C} + C\frac{m_C}{m_B + m_C} \right). \tag{6.1}$$

We then calculate the new vertex $H_l \in \mathbb{R}^3$, where A^* and A'^* represent the respective triangle vertices on the mesh M_s in $\mathbb{R}^3$, using:

$$H_l = \left(A^* \frac{m_A}{m_A + m_{A'}} + A'^* \frac{m_{A'}}{m_A + m_{A'}} \right) \tag{6.2}$$

Here, $m_A = m_B + m_C$. This can be done for any initial point (i.e. we must not necessarily first compute A'). We perform this mapping process for every Hilbert vertex h_l, and end up with the Hilbert curve vertices $H_l, l = 1, \ldots, M$ on the mesh M_s.

6.1.3 Sampling and Density Computation

In this section we explain how discrete properties are computed at each surface Hilbert vertex H_l along the curve in order to produce density controlled sample distributions.

Once the Hilbert curve has been generated in the parameter domain and mapped onto the original surface mesh using the parameterisation f, the curve is then sampled with a 1D distribution to produce a low-discrepancy distribution on the mesh M_s. To distribute the 1D points uniformly on the surface, along the Hilbert curve, we approximate the surface integral $\int_{M_s} \delta \, dA$ with a discrete (at each H_l), cumulative, surface sampling density S_l, and then sample points according to this density. When working with smooth, explicit or implicit surfaces, differential surface properties (e.g. area and curvature) can be easily calculated using the fundamental forms of the surface obtained from the exact parametrisation formula. However, computing similar properties on a mesh approximating the continuous shape must be done carefully in order to get accurate results.

Meyer et al. [84] describe a robust set of methods for computing differential properties on triangulated two-manifolds. Their methods avoid the use of polynomial reconstruction of the surface, reasoning that this may introduce unexpected surface behaviour, and in fact may often overestimate properties. They demonstrate competitive, or better, error rates when approximating Gaussian and mean curvatures on regularly and irregularly sampled meshes. In order to calculate the local density at a Hilbert vertex H_l on the mesh M_s, we therefore compute the differential properties using their approach. The points must first be triangulated for this process. However, this is simple, as a local triangulation is simply constructed for each vertex using the eight neighbours of each Hilbert vertex.

In order to distribute a set of points with low discrepancy on the mesh, we must first compute the local area A_l associated with each Hilbert vertex H_l. Let $\mathcal{N}_l$ be the triangles forming the one-ring neighbourhood of H_l by linking H_l to its neighbouring Hilbert vertices. For each triangle $T \in \mathcal{N}_l$, we compute the fractional area $\mathfrak{A}(T)$ of T that is assigned

to H_l (the remaining fractional area is assigned to T's other vertices). The sum of these areas is then an approximation of the area A_l for the region of H_l:

$$A_l = \sum_{T \in \mathcal{N}_l} \mathfrak{A}(T). \tag{6.3}$$

There are two cases when computing the fractional triangle area $\mathfrak{A}(T)$ belonging to H_l, distinguished by the type of the triangle T. The first case is formed for non-obtuse triangles, for which we can compute the Voronoi area. The Voronoi area gives the most accurate discrete approximation for the fractional area of the triangle which belongs to H_l [84]. Given a triangle $\triangle QH_lR$, with Q and R two consecutive one-ring neighbours of H_l, then the area $\mathfrak{A}(T)$, based on the Voronoi region, is (for simplicity we refer to the angles in the triangle as $\angle_T Q$ for the internal angle at point Q and similar for R):

$$\mathfrak{A}(T) = \frac{1}{8}(\|H_lR\|^2 \cot \angle_T Q + \|H_lQ\|^2 \cot \angle_T R) \quad \text{(non-obtuse case).} \tag{6.4}$$

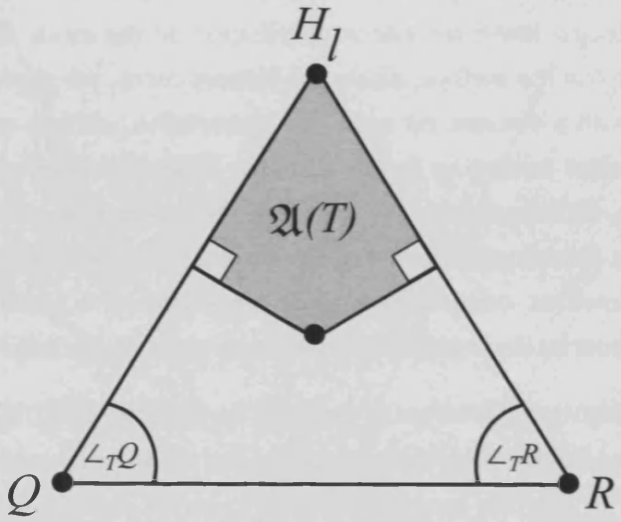


Figure 6.3: Voronoi area $\mathfrak{A}(T)$ for a triangle T.

In the second case, if the triangle $\triangle QH_lR$ is obtuse, we approximate the area by computing the barycenter instead of the Voronoi centre, taking the mid-point of the side opposite to H_l, and scaling the resulting area depending on whether the angle $\angle_T H_l$ is the obtuse angle or not [84]. Thus compute the scaled area $\mathfrak{A}(T)$ of an obtuse triangle is

$$\mathfrak{A}(T) = \begin{cases} \text{area}(\triangle QH_lR)/2 & \text{if } \angle_T H_l > \pi/2, \\ \text{area}(\triangle QH_lR)/3 & \text{otherwise,} \end{cases} \quad \text{(obtuse case)}$$

where area(QH_lR) is the area for that triangle.

This approach for calculating A_l relies on the mesh having no boundary. In our approach, we cut the mesh to allow a parameterisation over $[0,1]^2$. Thus, we must consider how to compute differential properties for vertices lying on the boundary. If the vertex H_l does not have a complete one-ring neighbourhood, the area A_l of the region for a vertex that lies on the boundary of a mesh can simply be approximated by

$$A_l = 2\pi \sum_{T \in \mathcal{N}_l} \frac{\mathfrak{A}(T)}{\angle_T H_l}. \tag{6.5}$$

This allows us to better approximate the area element for the vertex H_l, preventing a lower sampling density at the boundary. Note that if the original mesh had no boundary and was cut to a disc topology, we can instead simply compute the differential properties on the original mesh M_s before cutting, thus eliminating the boundary problem.

Surface curvature is commonly used to control the density of a distribution of points. A higher density is generally desirable in areas of high curvature in order to preserve features. In order to compute curvatures for each H_l, we must first compute the normals. Building on the methods described in [84], we compute the normal n_l for a vertex H_l, where α_{lm} and β_{lm} are the angles opposite the edge $H_l H_m$ in the two triangles containing that edge (see Figure 6.4),

$$n_l = \frac{\frac{1}{2A_l} \sum_{m \in \mathcal{N}_l} (\cot \alpha_{lm} + \cot \beta_{lm})(H_l - H_m)}{\left\| \frac{1}{2A_l} \sum_{m \in \mathcal{N}_l} (\cot \alpha_{lm} + \cot \beta_{lm})(H_l - H_m) \right\|}. \tag{6.6}$$

This gives an accurate estimate of n_l. If the value of n_l before normalisation is zero due to zero-curvature at H_l, we instead compute a normal for each triangle $T \in \mathcal{N}_l$ of which H_l is a member and take the average of these normals. This computation, unlike Equation 6.4, results in a vector not a scalar quantity, as we are no longer computing distances between H_l and its one-ring neighbours, but vectors.

In order to provide curvature-based density control of sampled points, we first compute the principal curvatures, k_1 and k_2 for each point on the mesh. We compute the principal curvatures in this way first in order to provide more flexibility in choice of scalar curvature to use. We use the technique described in [85]: determine the one-ring neighbourhood for a point, project these points onto the tangent plane of the current vertex, fit a quadric polynomial to this set of points and calculate the principal directions and curvatures from the shape of the polynomial at the original point. However, to ensure accuracy, instead of computing the curvature for the mesh M_s, and interpolating this for each Hilbert vertex H_l, we instead calculate it independently for each Hilbert vertex. Thus, we instead compute the curvature using a one-ring neighbourhood of each Hilbert vertex using the quad-tree structure of the Hilbert curve. We then compute the Gaussian curvature $K = k_1 k_2$ or mean curvature $H = (k_1 + k_2)/2$ (depending on which is needed for the density).

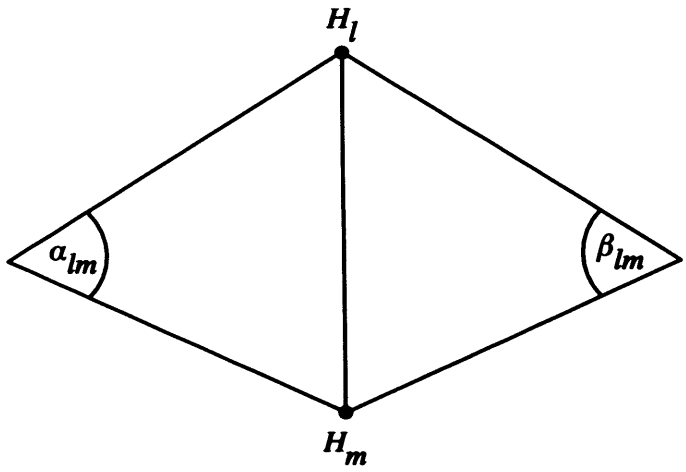


Figure 6.4: Angles used to compute the surface normal n_l at vertex H_l.

Once we have calculated the area, and density (e.g. based on curvature above), we then compute a cumulative density function, where the discrete density φ_l for each vertex may represent any desired density:

$$S_l = \sum_{k=1}^{l} \varphi_k A_k. \tag{6.7}$$

We then generate a set of 1D jittered samples $\{q_k : k = 1, \ldots, N\}$ 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, $l = 1, \ldots, L$, we move along the Hilbert curve, and whenever S_l becomes larger than the threshold described by the 1D sequence q_k (multiplied by S_L), we sample a point p_k at a vertex along the surface Hilbert curve, producing a distribution of jittered points lying on the surface of the input mesh M_s.

6.2 Evaluation of Mesh Sampling Technique

In this section, we evaluate the sample distributions produced using the algorithm described in Section 6.1. We first discuss a mesh discrepancy measure, assessing how well the new point samples are distributed on the original mesh (see Section 6.2.1). We then investigate the geometric distance between the input mesh and the new sampled surface using the Hausdorff distance (see Section 6.2.2). Finally, we discuss timing results for our sampling algorithm (see Section 6.2.3).

6.2.1 Discrepancy of Point Distributions on Meshes

First we assess the quality of resampled distributions on several example meshes by measuring their discrepancy. Then we consider how different distributions can have an impact on the number of samples required to cover a surface in particular for point-based graphics, comparing our technique to other popular methods.

For distribution quality assessment we consider the star discrepancy measure [132]. For a given number of points N, the lower the discrepancy the better. In the following tests, generalising those introduced in Section 5.1.1, we demonstrate that the point distributions produced by our algorithm behave as expected for a low-discrepancy distribution.

In this section, we generalise the numerical discrepancy experiments introduced in Section 5.1 to arbitrary triangle meshes. We compute the discrepancy of point distributions sampled on triangles meshes using contiguous sets of triangles as sample areas. We generalise the triangle-mesh star discrepancy to

$$D^{*,M_s}(P) = \sup_{\mathcal{Q}\subset\Omega(M_s)} \left| \frac{|P \cap \mathcal{Q}|}{|P|} - \text{area}(\mathcal{Q}) \right| \tag{6.8}$$

where $\Omega(M_s)$ is the set of all contiguous triangle sets in the mesh M_s, $|P \cap \mathcal{Q}|$ is the number of points in P inside some contiguous subset of mesh triangles $\mathcal{Q}$, and area($\mathcal{Q}$) is the total area of all triangles in the subset $\mathcal{Q}$.

In order to get uniformly distributed triangle subset areas, we first compute the total area A_{M_s} of M_s. We then compute a random number in $[0, 1]$ that represents a fraction of the total area A_{M_s} and choose a vertex at random from M_s. Then, one triangle at a time, we grow the selection in rings around the initial vertex. Every time a triangle is added to the selection, we add the area of that triangle, to the total selection area A_{total}. When $A_{\text{total}} \geq A_{M_s}$, we stop growing the selection, the output being a contiguous set of triangles $\mathcal{Q}$. When the discrepancy is measured using Eq. 6.8, the area of the contiguous subset of triangles area($\mathcal{Q}$) = A_{total}.

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 $A_\triangle / A_{M_s}$, where $A_\triangle$ is the triangle's area. We then generate each point by randomly bi-linearly interpolating the three vertices of that triangle. Note that this is a locally uniform, un-correlated distribution within each triangle, but there is a correlation when considering all triangles, actually improving the distribution compared to a purely random distribution.

We compute the discrepancy for point distributions 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 $\omega \geq 10$ (see Section 4.2.1). We sample N points on this curve for each test using a constant density of $\delta = 1$ to give an even distribution of points on the surface.

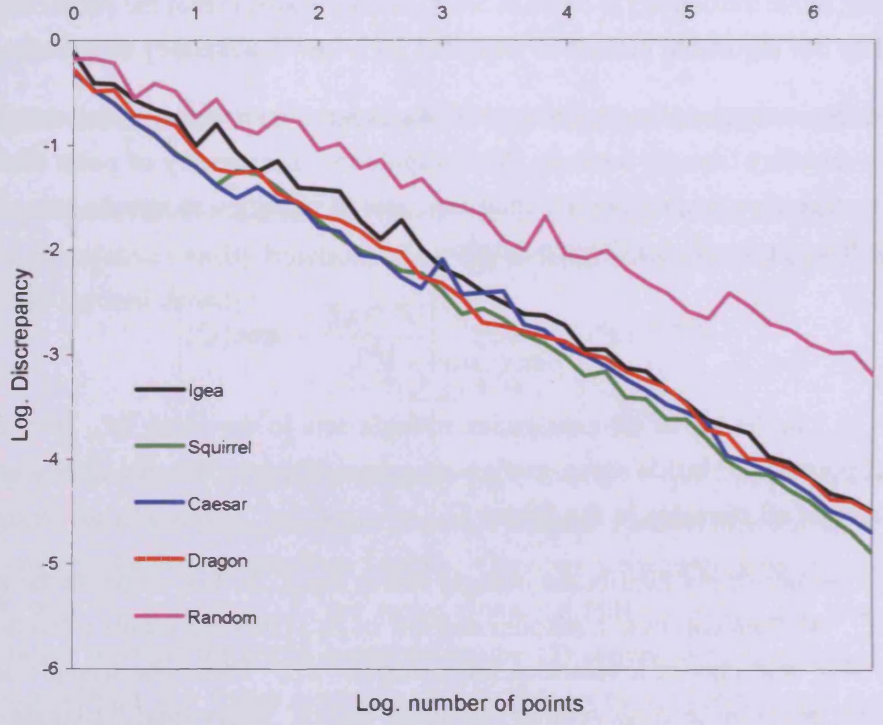


Figure 6.5: The discrepancy of our resampling approach on various meshes and a random control distribution.

Figure 6.5 shows the results for the Julius Caesar, Squirrel, Igea and Chinese Lion triangle meshes, all of which are from the Aim@Shape repository [1]. All graphs are on a logarithmic scale. In each case, our approach demonstrates discrepancy which scales better with N than the random distribution (which gave very similar results for each mesh, so is only shown once). Gradients for the slopes shown in Table 6.1 back up this consistency, demonstrating a mean gradient of about -0.72 compared to -0.5 for the random sequence. These results mirror those shown in Section 5.1 using low-discrepancy point samples generated on a parameterised sphere; they also correspond very closely to the

[1] http://www.aimatshape.net/

known Hammersley distribution on the sphere [34]. This indicates that our approach generates low-discrepancy point distributions on meshes.

Mesh	*Discrepancy*	*Metro*		
	Gradient	75%	50%	25%
J Caesar	−0.70	0.009	0.014	0.020
Squirrel	−0.74	0.008	0.010	0.018
Igea	−0.73	0.0032	0.013	0.017
Lion	−0.73	0.009	0.016	0.022

Table 6.1: Gradients of Discrepancy Tests, and Hausdorff Distances for Test Meshes.

6.2.2 Hausdorff Distance

In this section, we consider a measure that determines the distance between two surfaces. Measuring the discrepancy of a set of points gives us a good indicator of the quality of the distribution, and ensures that we have sampled the surface uniformly and non-regularly. However, practically, it does not give us any information regarding how well the distribution of generated points preserves the shape of the original mesh. Thus, in order to investigate how well a resampled surface approximates the original, we compute the geometric distance between the original surface and a triangulation of the resampled point set, using the symmetric Hausdorff distance [28].

As explained in Section 1.1.1, a measurement based on the Hausdorff distance between two surfaces S_a, and S_b, has been defined using the average distance from a surface S_a to a surface S_b [28]. This average can be defined by the average of the distance between all points p_a on the surface S_a, and their closest point p_b on S_b:

$$D_{\text{avg}}(S_a, S_b) = \frac{\int_{S_a} \inf_{p_b \in S_b} d(p_a, p_b)\, dA}{\text{area}(S_a)} \tag{6.9}$$

where area(S_a) is the surface area of mesh S_a. In practice this is approximated by discretely sampling S_a with points p_a, and computing the closest point on S_b, p_b using an efficient data structure. This distance is then averaged to approximate the Hausdorff distance from one surface to another. Given two surfaces S_1 and S_2, computing $\max(D_{\text{avg}}(S_1, S_2), D_{\text{avg}}(S_2, S_1)$ gives us the symmetric average Hausdorff distance between the two surfaces.

A popular algorithm, Metro [28], first computes the bounding box of the target mesh, and partitions it into a regular arrangement of cubes. It then performs a scan-line sampling

of the target mesh in order to simplify area computation. For each uniform sampled point p_l on the surface, the nearest mesh triangle T_l is found by first searching in cube c (containing p_l), then in adjacent cubes c_{ij}, increasing in distance from point p_l. Once the nearest triangle T_l has been found, rays are cast from p_l onto T_l, computing the distance d_l. The smallest distance is then recorded.

In order to generate a surface mesh for this measure, we simply generate a set of sample points on the surface M_s, and triangulate them in the parameter domain using conformal Delaunay triangulation in $O(n \log n)$ time [113]. Whilst Steiner points can be inserted to improve the shape of triangles, we do not insert them, as these would affect the prescribed density, and we do not alter the triangulation in any way. We discuss further mesh adjustment when discussing the remeshing application (see Section 7.2).

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 6.1 demonstrates the results, showing a Hausdorff distance of at worst, 0.009% for 75% of the points, 0.016% for 50% of the points, and 0.022% for 25% of the points. A popular remeshing technique by Surazhsky and Gotsman [119] 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%.

There is another approach to computing this measure of distance. We could simply take the input mesh, and compute the distance between the sampled points and the mesh, thus avoiding the process of triangulating the samples [129]. Whilst a Delaunay triangulation is fairly standard, triangulating the point set also changes the surface somewhat as a different triangulation of the same point results in a different surface, and a different distance measure. However, the advantage of triangulating the generated point set is that we can then also therefore compute the distance from the input mesh to the new surface (as well as the distance from the sampled points to the input mesh), giving us the symmetric distance measure. By computing both distances like this, we can take the larger of the values (whether locally or globally), which gives us the maximum difference between the two surfaces and thus a more accurate overall error.

6.2.3 Runtimes

In this section we consider the run-times for the stages of our mesh sampling algorithm. Table 6.2 gives run-times for each stage of our algorithm. Parameterisation using Floater's

method [42] takes 70 seconds for the 268K triangle Igea. Complex mesh cuts, however, can result in longer parameterisation times, e.g. for the Chinese Lion. Generating the adaptive Hilbert curve 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 takes about 1 second. After this pre-processing has been done, the mesh can be resampled in a fraction of a second regardless of the number of samples required (see Section 6.1 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. Note that triangulation time is considered in Section 7.2.

Mesh	**J Caesar**	**Squirrel**	**Igea**	**Lion**
No. of faces	49K	20K	268K	40K
No. of vertices	24K	10K	134K	20K
Parameterisation	5.3s	1.3s	70s	152s
Curve Generation	7s	4s	10s	5s
Mapping	1s	0.8s	1.1s	1s
Surface Sampling	0.001s	0.001s	0.0015s	0.001s
Total	13.3s	6.1s	81.1s	158.1s

Table 6.2: Timing for different stages of our sampling approach for some meshes.

6.2.4 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 or the number of samples N 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.

The numerical discrepancy of the resampled points behaves similarly to points generated on parametric surfaces using space-filling curve methods 5.1.4, demonstrating gradients of the discrepancy's logarithm of ≤ -0.7. All models tested demonstrated similar results which are consistently better than for random sampling. Whilst not investigated due to the unavailability of their implementations, the output from popular remeshing approaches such as [119], results in a Poisson disc distribution, which demonstrates a discrepancy that is consistently worse than the jittered output from our approach [115].

When measuring the average symmetric Hausdorff distance of the triangulated resampled (remeshed) surfaces, we obtain a low distance error between the surfaces for all meshes, which scales well with decreasing sample points. The average Hausdorff distance approximated with by Metro gives an indication of 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, e.g. near sharp edges.

The approach discussed in this chapter looks at mesh resampling, where all output samples lie on the surface of the original mesh. A higher-order surface approximation could easily be used if felt desirable or appropriate, e.g. if the output were a super-sampling of the input mesh. When the Hilbert curve vertices H_l are mapped back to the mesh using the barycentric co-ordinate approach (see Section 6.1.2), a local cubic Bézier approximation could be computed [122] for the containing triangle and its neighbours. The Hilbert vertices could then be projected along their normals onto this local Bezier surface, giving a new set of vertices H_l'. This would be a computationally simple approach, though has the drawback of ignoring sharp edges if done naively.

6.3 Summary

In this chapter, we have demonstrated a novel mesh sampling algorithm that is adapted from the algorithm introduced in Chapter 4, taking as input an arbitrary genus polygonal mesh, and outputting a high-quality set of sample points. The output points approximate the original surface, with similar accuracy when measured using Metro to the best results in the literature. Experimentally they exhibit good discrepancy behaviour suggesting the distributions are of low discrepancy. We have also demonstrated quick generation times for the initial distribution of points, including parameterisation and space-filling curve generation. The sampling can then be completely adjusted in about a millisecond (see

Section 7.1 for further details), though if the density becomes significantly greater, then the Hilbert curve must be further subdivided. In the next chapter we build on the sampling method introduced in this chapter, tailoring it for the applications of point-based graphics, remeshing and prototype painting.

Chapter 7

Mesh Sampling Applications

In this chapter we employ the algorithms described in Chapter 6 to demonstrate their usefulness in a selection of applications and compare them to existing approaches. The algorithms are used to generate low-discrepancy point distributions that accurately approximate the input mesh geometry, and are used as a basis for point-based graphics (see Section 7.1), remeshing (see Section 7.2) and robotic painting of prototype models (see Section 7.3). All three applications require similar algorithms for handling and sampling an input mesh, though their outputs and uses are quite different. Our point-based graphics approach considers point coverage for sample density and demonstrates real-time levels of detail and silhouette-enhancing view-dependent rendering. The remeshing approach generates high-quality triangulated meshes and considers triangle quality. Finally, our robotic painting method samples textures to produce high-quality dithered point samples for painting. In this chapter, due to the nature of the applications, the quality of the output is assessed visually in the context of the application (see Section 6.2 for quantitative analysis results).

7.1 Point-based Graphics

Whilst meshes are the most widely used method for rendering and storing models, point-based representations of surfaces have become a viable alternative, especially for complex, smooth or dynamic surfaces [65] as explicit topological information does not have to be computed (see Section 3.1). However, simply rendering the vertices of a mesh as a set of points is problematic, as they are often irregularly spaced due to limitations of the capture process, and due to the fact that large flat areas can be described using a small number of large polygons. Also, the vertices describe polygons that are the main rendering element, so the sampling requirements are different; a good representation of the surface shape by the triangles is desired, but not necessarily a good point coverage. Vertices of such meshes are thus not necessarily suitable for use as primitives for point-based rendering. There is, therefore, a need to resample meshes to produce point sets that are more

suitable for point-based rendering with properties such as good equidistribution and good shape preservation. 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, or possibly to impose a limit on the number of points used.

Surface splatting is a popular method to render point-based surfaces using oriented disc-like shapes; purely point-based methods, rendering a point using one or more camera-oriented pixels, run into problems due to their poor image-space representation (see Section 3.1). As the camera moves closer to a model, the sample density must increase considerably to maintain the surface coverage without holes when using only points. An alternative is to represent a point by a disc-like shape, orthogonal to the surface normal, which has a shape in image space (a splat) referring to object space properties, simplifying surface rendering considerably. However, continuity must be considered (see [37]), and generating a surface distribution that leaves no holes is still difficult to achieve.

Whilst we do not devise a new rendering technique, in order to provide a good surface coverage for point-based rendering, the sampling must take into account surface coverage. The criteria that we employ are as follows: the points should be spaced evenly, usually with respect to some function of curvature. By distributing the points according to curvature, and varying the size of the splat, we can make efficient use of the points: fewer, larger splats in areas of low curvature, and more, smaller splats in areas of high curvature. The points must not, however, be arranged in a regular pattern, in order to avoid artifacts and aliasing problems [133], and to prevent structure being added to the existing surface (see Section 5.3). 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 relative to the splat size used to represent the point. A distribution of points with a low discrepancy is a good indicator of the quality of the sampling method, with respect to how accurately we might represent the input surface. However, a distribution having low discrepancy does not guarantee that neighbouring points will be equidistant in all directions, a practical requirement when rendering a surface with points. In this section, we describe our approach for generating point distributions for point-based graphics (see Section 7.1.1), demonstrate visual results (see Section 7.1.2), and describe levels of detail (LoD) and view dependent rendering applications (see Section 7.1.3).

7.1.1 Sampling Approach for Point-based Rendering

In this section, we discuss the adaptations made to our standard mesh sampling algorithm for use in point-based graphics (see Section 6.1). We first look at improving the uniform

distance between sample points on a surface, followed by our fast neighbourhood look-up approach.

When rendering a point set, either using splats or simple point primitives, achieving complete surface coverage, so that no gaps between points exist, is very important. Thus, for the application of point-based rendering, we are interested in keeping the distances between points as constant as possible, minimising the gaps in the distribution. By keeping the distance constant, fewer points are required to fill holes, and fewer points become redundant at the rendering stage. As introduced in Section 7.1, the density, and thus the distance between points, should be determined by the surface curvature, such that the splat size increases with the spacing between the points. Alexa et al. [3] describe a moving-least-squares approach to computing a regular point set from a point cloud: a least squares surface is iteratively matched to the point set, which is then regularly sampled to the desired density. Whilst they do not focus on the quality of the sample distribution, they suggest an approach to removing holes between samples using Voronoi diagrams. A Voronoi diagram, the dual of a Delaunay triangulation, is a planar tessellation, with a cell belonging to each centre-point, where, for a given centre-point, the cell is defined as every possible point in the metric space that lies closest to this centre-point. Where more than two such cells meet at a point gives a Voronoi vertex. Due to the complexity of building a Voronoi diagram of the whole surface in their approach, they project local neighbourhoods of a point into its tangent plane and compute a local Voronoi diagram of these points. For each Voronoi vertex, the circle, centred on the vertex, and defined by the closest three neighbouring sample points is computed. If the radius of this circle is larger than a user defined parameter, then a point is sampled at the Voronoi vertex, essentially filling the hole.

After the Hilbert curve has been generated, mapped to the surface, and sampled (see Section 6.1), we adopt a similar approach to fill gaps in the distribution. By filling gaps in the coverage, we can ensure a hole-free sampling. However, we build a global Voronoi diagram in $[0,1]^2$ as we already maintain the (u,v) co-ordinates of the points in $[0,1]^2$ as a result of our parametric approach. The global Voronoi diagram is faster to compute than a set of local approximations, and as a point is added, the Voronoi diagram can then be incrementally updated. Thus, in our approach, the diagram need only be computed once, rather than iteratively computing local diagrams. Following Alexa et al. [3], we then consider the radius of each circle used to construct the Voronoi diagram, and place a point at its vertex if it is larger than a specific (user-defined) parameter. Due to parametric stretch introduced by parameterising the input mesh to $[0,1]^2$, the distances in the parameter domain are not consistent with distances on the mesh: a large circle radius in the parameter domain (one that would otherwise be sampled at the vertex), may not be

large on the surface. To compensate for this, we compute each radius by considering the position of the three points in $\mathbb{R}^3$ rather than in $[0,1]^2$. This results in a circle radius that is accurate with respect to the gap between the points on the mesh in $\mathbb{R}^3$. We then use this radius value to decide whether to add a point at the vertex. If a point is to be added, it is sampled at the nearest Hilbert vertex, and is then added to the existing Voronoi diagram. This process continues until there are no Voronoi circles that have a radius larger than the average or user-defined value.

Point neighbourhoods are used in many algorithms in geometry and graphics, and are given explicitly in mesh-based surface representations. Due to the lack of this explicit connectivity between points in a point-based surface representation, local neighbourhoods must be computed, often dynamically, when they are required. As discussed in [65], two main approaches to neighbourhood computation in point-based graphics exist: Euclidian neighbourhoods, and k-nearest neighbours. The k-nearest neighbours approach works regardless of local density and feature size if the point distribution is sampled with respect to these properties [8], and scales with a linear complexity with respect to the number of points. The adaptive Hilbert curve, constructed using the algorithm given in Section 4.2, is built using a quad-tree. This quad-tree functions as a partition of 2D space on the mesh surface, providing a fast, implicit, k-nearest neighbours computation method. Whilst this is already a fast approach, neighbours can then be cached per vertex by storing the 1D Hilbert curve index of each neighbouring point. Alternatively, if an approximate neighbourhood is required, for example, for the fast computation of surface normals, the previous and next k neighbours along the curve can be chosen due to the Hilbert curve's good spatial coherence.

7.1.2 Visual Results

We now demonstrate the results visually for some meshes converted into point representations using our sampling approach. Our research does not aim to improve the rendering process of point based models, thus we render the point representations using a simple technique described in [1], which produces an output similar to Gourad shading.

Figs. 7.1–7.3, show uniform splatting, splatting according to density given by Gaussian curvature, and rendering using simple splats according to Gaussian curvature for the Squirrel (1.5K, 1.5K and 15K points respectively), Chinese Lion, and Igea meshes (both 2K, 2K and 25K points respectively). The distribution of splats for the uniform sampling of all meshes appears regularly spaced for all models. The curvature controlled sample distributions for each mesh show higher densities of points in high curvature areas, and

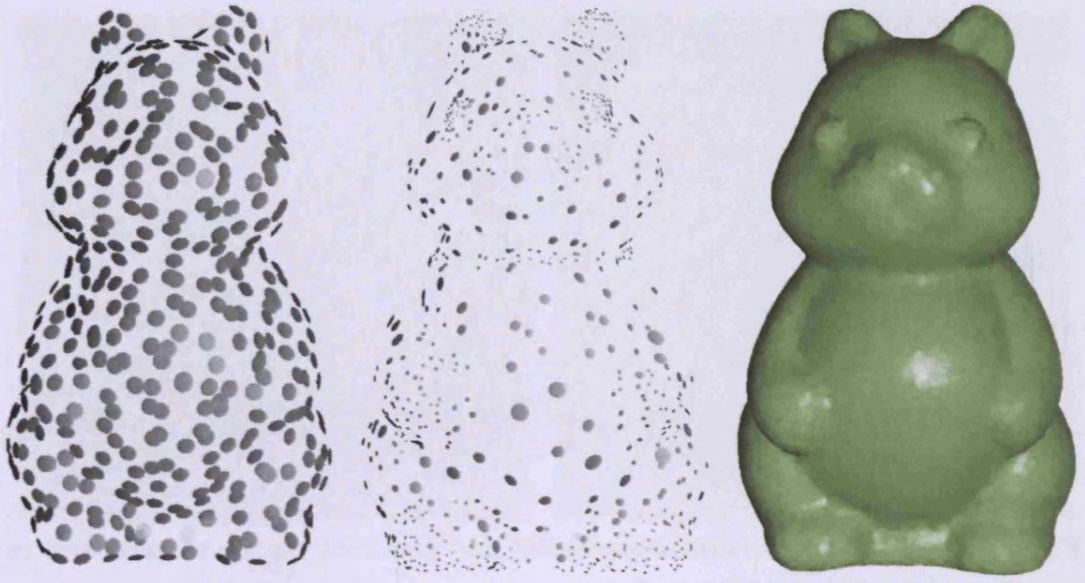


Figure 7.1: Uniform, curvature-controlled and rendered curvature-controlled resampling of the Squirrel model.

lower densities in low curvature areas. The rendered Squirrel model shows very good silhouettes, although slight inconsistencies in shading between splats is clearly visible. This is due to the the quality of the rendering, similar to that achieved using Gouraud shading. The Igea and Chinese Lion meshes, sampled more densely, show similarly good silhouettes, especially highlighted by the straight edge on the base of the Chinese Lion mesh. Inconsistencies in the shading are not obvious due to the higher sampling density.

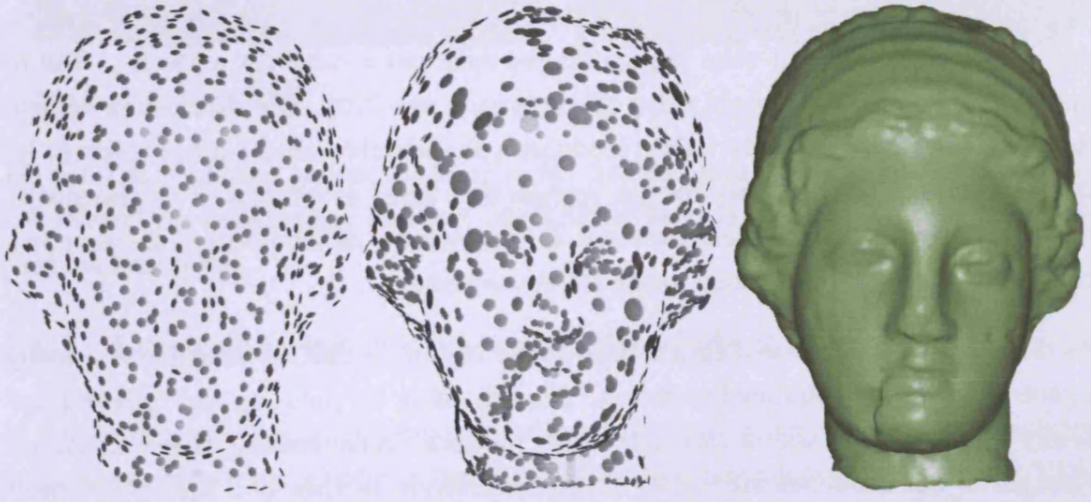


Figure 7.2: Uniform, curvature-controlled and rendered curvature-controlled resampling of the Igea model.

Figure 7.3: Uniform, curvature-controlled and rendered curvature-controlled resampling of the Chinese Lion model.

Low-discrepancy and stratified distributions have been widely used in computer graphics, e.g. for applications such as radiosity [63] and ray tracing [31]. In [106], 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. Most techniques employ splatting or surfel rendering techniques [65] or perform direct rendering (see Section 3.1) to provide a contiguous surface coverage.

In Figure 7.4, we have sampled 3000 points in the unit square, drawn using solid circles, first using a random distribution, then the Niederreiter distribution, followed by our technique. Finally, we show our technique with Voronoi hole-filling using 2000 samples, and holes were filled that were larger than the splat radius used for rendering. The final number of points generated using this approach was 2290. Our distribution without the Voronoi technique already visibly produces a considerably better surface coverage for this application. The boundary for the Voronoi hole-filled distribution is rougher due to the reduced number of samples. However, it improves our method further, resulting in a completely hole-free distribution, using 700 fewer points.

As discussed in [117], 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 similar results. However, the advantages of our technique are: (i) to achieve a uniform density in a square parameter domain, the number of samples need not be a perfect square, (ii) we can maintain an evenly stratified distribution on an arbitrary surface, (iii) parametric distortion does not affect the neighbourhood and connectivity of the samples.

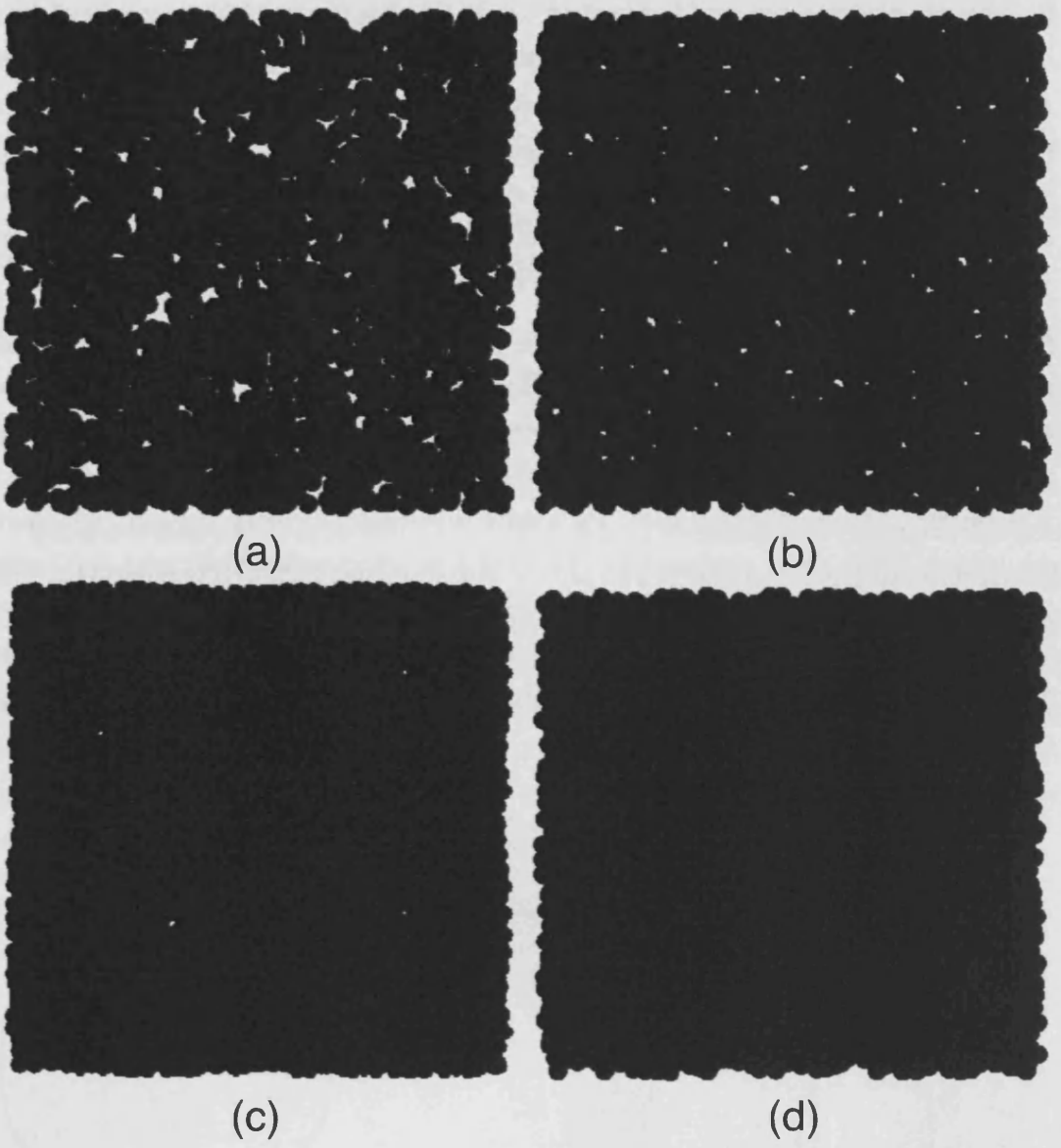


Figure 7.4: Splatted points in $[0,1]^2$: (a) 3K random, (b) 3K Niederreiter, (c) 3K Hilbert curve sampling method, (d) 2.3K Hilbert curve sampling method with Voronoi hole filling.

7.1.3 Levels of Detail and Viewpoint Dependent Rendering

In this section, we look at two more specific ways of using our sampling approach for point-based rendering, for levels of detail (LoD) and viewpoint dependent rendering (VDR).

Our sampling approach provides a real-time solution to continuous LoD, as points can be resampled quickly, as long as the density does not increase beyond limits imposed by the initially generated Hilbert curve (see Section 6.2.3). This is not a problem as

in LoD the maximum density is usually known. We first discuss LoD representation of surfaces with our approach. We compute the distance d between each vertex on the surface Hilbert curve, and the camera. Some simple function (linear or non-linear) of d is used to determine the number of points N 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 Section 6.1.3). This step typically takes less than 10 milliseconds, depending on the depth of the curve, allowing for interactive frame rates.

Figure 7.5 shows local LoD control for the cow model (from the Aim@Shape repository [1]), 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. Normally in LoD approaches, an average of d is computed for the whole object, and as d changes, the whole object is resampled. Using our approach, the model is sampled according to local distances, resulting in a fine grain, continuous LoD. To reduce computational overhead, due to the good spatial coherency of the curve, d may be computed for a subset of the curve vertices and interpolated the the remaining vertices. The computation of d can also efficiently be divided between multiple CPU cores as each calculation is independent.

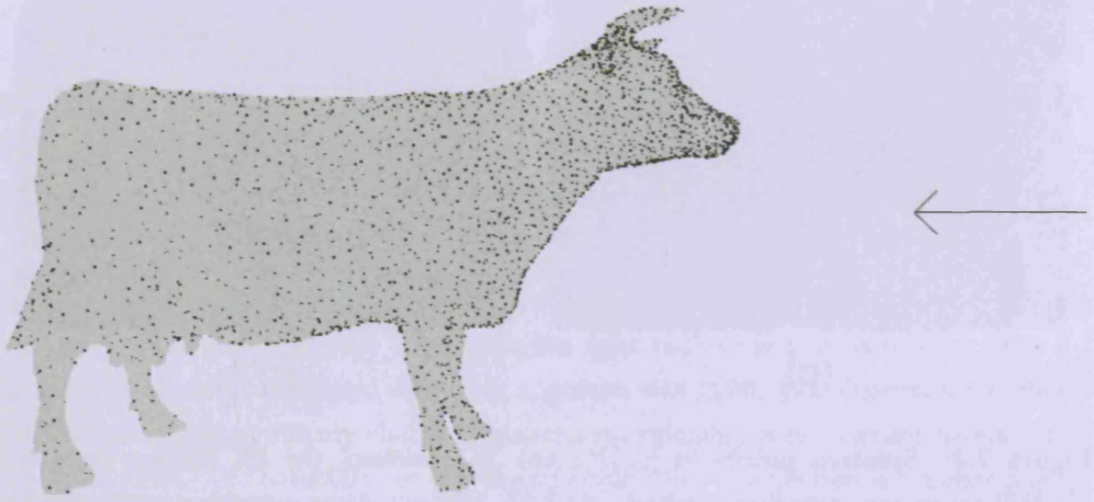


Figure 7.5: 4K points with localised LoD where the arrow represents the viewing direction from the scene camera.

We also apply our approach to the problem of VDR, such that we can increase the density locally at the silhouette of the object, whilst maintaining a lower density over the rest of the object. A low-density for the non-silhouetted parts of an object are often hidden by a surface texture. However, low-density silhouettes are, visually, very obvious, and thus increasing the density at the silhouette in this manner can greatly improve the visual

[1] http://www.aimatshape.net/

quality of the rendered object. For VDR of a given surface, we compute the angle θ_l between the vector from the camera to a Hilbert vertex H_l and the surface normal at H_l:

$$\theta_l = \cos^{-1}\left(\frac{(C - H_l) \cdot N_l}{|C - H_l||N_l|}\right), \tag{7.1}$$

where C is the camera position and N_l is the surface normal at H_l. We adjust the local discrete densities s_l, which are later accumulated into the cumulative densities S_l (see Section 4.3), according to θ_l:

$$s_l = \begin{cases} 1 - \cos(\theta_l) + \sigma & \text{if } \theta_l > 0, \\ 0 & \text{otherwise,} \end{cases} \tag{7.2}$$

where σ is a constant used to specify the minimum density. 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 Figure 7.6). Note that this density function is only used to demonstrate the effect. 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. Figure 7.6 demonstrates this result for a sphere shown from different angles relative to the camera direction to make the effect clearer to see. Our VDR implementation can recompute 50K sample points on an arbitrary surface approximately 40 times per second, and could be further optimised.

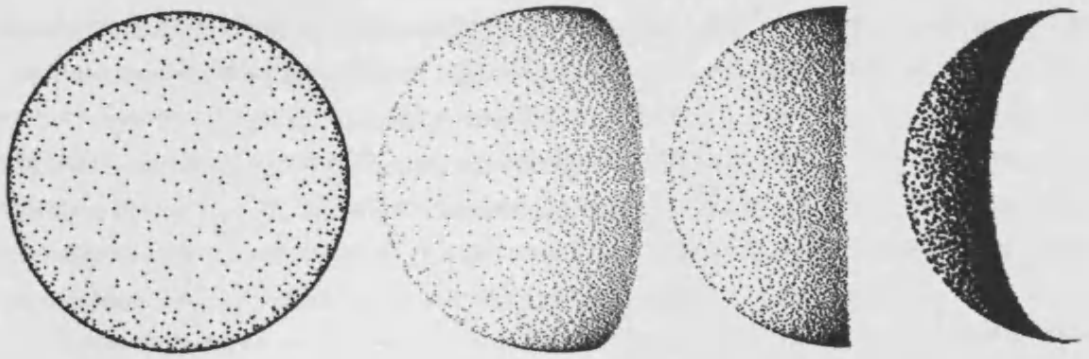


Figure 7.6: Silhouette enhancement with viewpoint dependent rendering: The left-most image shows a sphere from the rendered viewpoint. The remaining three images show the sphere from different angles, whilst fixing the view dependent rendering direction to that of the first image (in order to demonstrate the varying point density).

Once the Hilbert curve has been generated, and the quad-tree stored in memory, our approach allows for fast resampling of a surface. However, for both the LoD and VDR applications, if the local density changes significantly there may be insufficient pre-computed

curve vertices to sample the curve accurately, thus it may be necessary to subdivide the curve in real-time. If the maximum density is known in advance, then such subdivision is not necessary.

To achieve a high sampling fidelity, we require a local density $\omega \int_U \delta \, dA / \int_{M_s} \delta \, dA$, for each subset U of the mesh M_s, where $\omega \approx 10$ (see Section 6.1.2). If a large change in density occurs, and the curve does not adequately achieve the required local density, then the sample distribution is initially approximated using the existing curve. We then create a new thread to update the tree, 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 small changes, the quad-tree can be locally subdivided. In this case, the local densities can be derived from the surface, the surface integral updated, and the cumulative integral re-computed. For larger changes, each leaf node can be subdivided, globally increasing the depth. In this case, the surface integrals must be completely updated. Finally, the updated curve is resampled to the required density with the correct accuracy. Note that this approach can be easily applied to our parametric surface sampling approach (see Section 4.2.1).

7.1.4 Discussion

In this section, we discuss the results of our mesh resampling method for point-based graphics applications. We have demonstrated 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 our distributions. Examples of the variance exhibited by some sampling methods are shown in Section 5.1.3, 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. We also show that we can further reduce the holes in the distribution using the parameterised Voronoi method, further improving the equidistribution of points, and ultimately improving the point-based rendered output.

Our approach to LoD representations allows the rendered surfaces shown in Section 7.1.2 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. To further speed up these approaches, we suggest utilising the hierarchal structure of the Hilbert curve to spatially divide the points on the surface, and reduce the amount of directional (VDR) and distance (LoD) computation.

Different levels of of the quad-tree can be used to provide different levels of coarseness and speed. For example, at the second level of the tree, the distance towards the centre of four quads would be computed, rather than all points. For the LoD approach, the distance towards the centre of each quad provides the vertices within each quad with a constant density. This greatly reduces the number of necessary distance computations, yet still allows for *local* levels of detail control on a surface. For VDR, an initial coarse angle computation could be first employed. This initial computation can then be recursively expanded down the Hilbert-curve quad-tree, allowing for fast pruning of nodes that are either too far away or out of sight of the camera. By utilising the data-structure for these methods, the computational cost can be significantly reduced. Further enhancement to the VDR approach could be done by adopting fast look-up computation for visibility, such as the method described in [45]. These examples of LoD and VDR show that the ability to chose an arbitrary density, in our sampling technique, allows the user to define various different sample distributions, dependent on the requirements of the application.

7.2 Remeshing

Remeshing is an important area of research, given the ease of producing high resolution meshes using 3D scanners and modelling software [74]. Often the original meshes are too dense to handle easily, render quickly, transmit over a network efficiently, or perform computations on; they may also be highly irregular due to limitations of the capture process. There is a need for remeshing such high-resolution models to a more manageable size, whilst producing a more regular sampling which also preserves shape well. A secondary objective may be to adjust the mesh density so that it meets some user criterion, e.g., that the mesh has triangles of approximately uniform size, or that the points on the surface have a locally specified density, e.g., proportional to surface curvature.

Remeshing is important in many fields such as computer graphics, modelling and simulation. For these applications, requirements of the output mesh vary considerably. Engineering applications such as finite element analysis and computational fluid dynamics often desire regularly connected almost equilateral triangles. In real-time visualisation, we may simply want the simplest construction that results in few noticeable artifacts as the scene changes. In modelling, we might require the most accurate representation of the original surface within specified storage requirements or processing time limits. However, as discussed in Section 3.2, an important requirement for all methods is that the original mesh is sampled in a way that preserves shape, and provides control over the distribution of either triangles or vertices. Our approach, when used with a constant density, attempts

to evenly (but not regularly) sample the surface, by utilising a low-discrepancy point distribution on the surface, which allows us to capture the original shape well, but at the same time, avoids any aliasing effects of regular-grid sampling.

Remeshing involves resampling points on the existing surface and connecting them to generate a new mesh. As shown in Section 6.2, low-discrepancy distributions are very good for capturing features accurately when sampling geometry. In this section, we investigate the use of low-discrepancy distributions to generate two-manifold simplicial meshes with respect to a density function.

We now describe the process of generating a remeshed surface with our approach. A set of points lying on the surface of the original mesh in $\mathbf{R}^3$ is first computed using the algorithm described in Section 6.1. Using the algorithm described in [113] for speed, we then compute a conformal Delaunay triangulation of the (u, v) co-ordinates of the points in $[0, 1]^2$. This triangulation is used for the (x, y, z) co-ordinates in $\mathbf{R}^3$.

If the mesh does not already have a disc topology, a cut is defined on the mesh (see Section 6.1.1). The cut is defined by a set of vertices connected by a sequence of edges, resulting in a poly-line on the mesh, V_e. This set of connected vertices is duplicated, resulting in a second, identical, poly-line V_e'. These two poly-lines are joined at the first vertex of each poly-line and the last vertex of each poly-line, resulting in a single loop. This loop is treated as a zero-area hole in the mesh. The mesh is then parameterised to $[0, 1]^2$, with the poly-line loop used as a boundary. A Hilbert curve is then generated in $[0, 1]^2$, covering the parameterised mesh (see Section 6.1). The Hilbert curve is then sampled, resulting in a set of low-discrepancy points. The boundary vertices of the parameterised mesh, V_e and V_e', are then added to this set of low-discrepancy points, and all the points are triangulated into a new mesh in $[0, 1]^2$. This mesh is then mapped back to R^3. The duplicate vertex poly-line V_e' is then removed from the new mesh, and any edges in the mesh referring to these vertices are instead referred to their equivalent vertices in V_e. This process closes the hole in the mesh, and results in triangle mesh with the same topology as the input mesh. Note, however, that this is not an optimal solution, as the sampling rate and quality of the poly-line vertices V_e is unlikely to be consistent with the low-discrepancy samples, and thus the triangles in the mesh connected to this poly-line may vary in size and shape considerably compared to the rest of the mesh. Thus, whilst a topologically consistent remesh is produced, quality issues may exist along the poly-line where the original cut was made. In practice, a better solution would be required.

Parameterising patches independently may result in much better parameterised planar meshes (better in this case meaning less distance or angle distortion inferred by the parametrisation). For the application of remeshing, where new samples produced as an

output must be triangulated, the problem of stitching the patches back together to achieve a globally smooth and uniform triangulation becomes a complex issue. This stitching problem occurs because the triangulation of sampled points is generally performed in the parameter domain. Triangulating the point set in $\mathbb{R}^2$ like this simplifies the computation greatly (see Section 7.2), and ensures an accurate construction of the surface (a triangulation of a surface in $\mathbb{R}^3$ is considerably slower, and may not give consistent or robust results [8]). A potential problem with a planar triangulation, is that in extreme circumstances it can lead to a self-intersecting triangulation. Distributing points on the surface according to the local curvature generally avoids this problem. Potentially, however, a triangulation of the surface in $\mathbb{R}^3$ could be employed, circumventing this problem, and the issue of global surface reconstruction (see Section 7.1.4). Gu and Yau [47], and Khodaskovsky et al. [64] demonstrate solutions to this problem of global surface reconstruction, locally parameterising the surface in order to minimse both distance and angle distortion.

7.2.1 Results

We now consider the visual results from the remeshing process. We first look at a triangulation of the samples produced by the algorithm described in Section 6.1. Figs. 7.7–7.9 show a uniform remesh, a remesh according to Gaussian curvature and a rendering of a remesh according to Gaussian curvature for the Squirrel (4.5K vertices, originally 20K), Julius Caesar head (15K vertices, originally 49K), and Igea (15K vertices, originally 286K) meshes. Computation times for these models are given in Table 6.2, and triangulation times for the Squirrel, Julius Caesar head and Igea meshes were 0.1s, 0.3s, and 0.6s respectively.

Visually, especially for the curvature based remesh, the features of the meshes are well preserved. As we further discuss in Section 7.2.2, the focus of our method is on the quality of the point distribution, not on the quality of the triangles. As a result of this, long and thin triangles may result, and the valence of the vertices in the mesh is not considered.

7.2.2 Discussion

We now discuss the remeshing results. We can produce visually high-quality remeshed surfaces that maintain the original topology of the input mesh, with an error when compared to the original surface, measured with the Hausdorff distance, competitive with the best in the literature (see Section 6.2.2). The discrepancy results for the point distribution

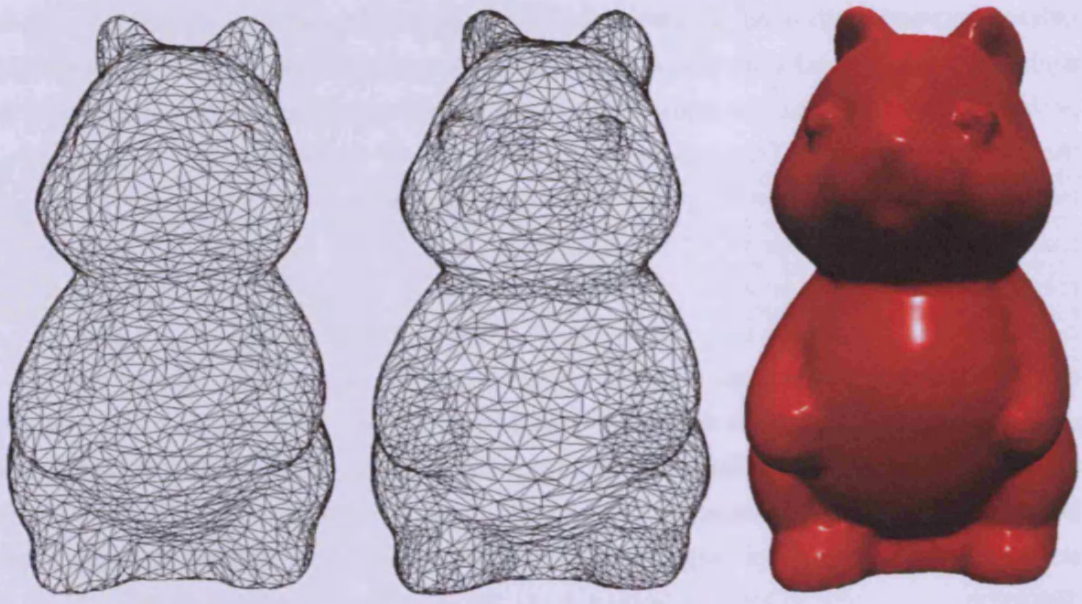


Figure 7.7: Uniform, curvature-controlled and rendered remesh of the Squirrel mesh.

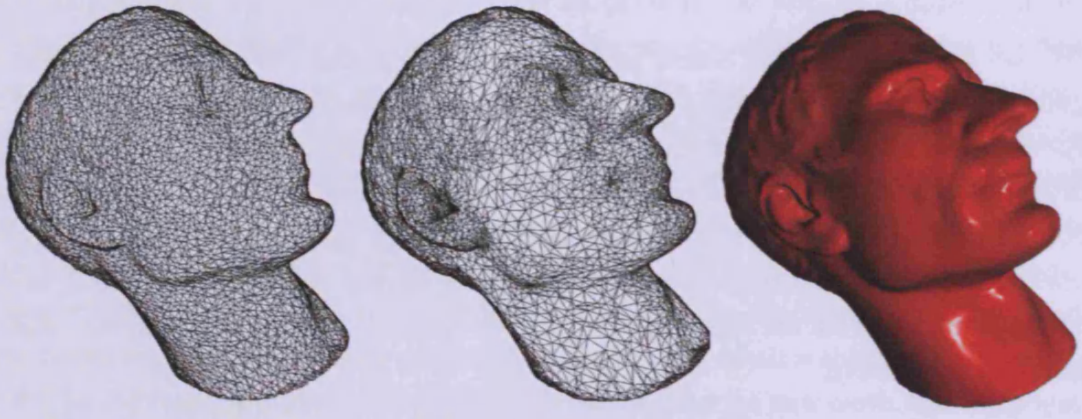


Figure 7.8: Uniform, curvature-controlled and rendered remesh of the Julius Caesar mesh.

from which the output mesh is constructed in our approach (see Section 6.2.1) indicate low discrepancy. However, the discrepancy results do not specifically indicate the quality of the triangle mesh, or indeed, how well the new set of triangles approximates the original mesh. We note that a low-discrepancy mesh may not always be the best or even desirable solution when remeshing, and that in some situations, nearly equilateral triangles may be more desirable, e.g. for finite element analysis. Yet a mesh based on low-discrepancy samples does allow us to sample the surface of the original input mesh very evenly, and provide consistent and predictable errors according to the Hausdorff distance. In Sec-

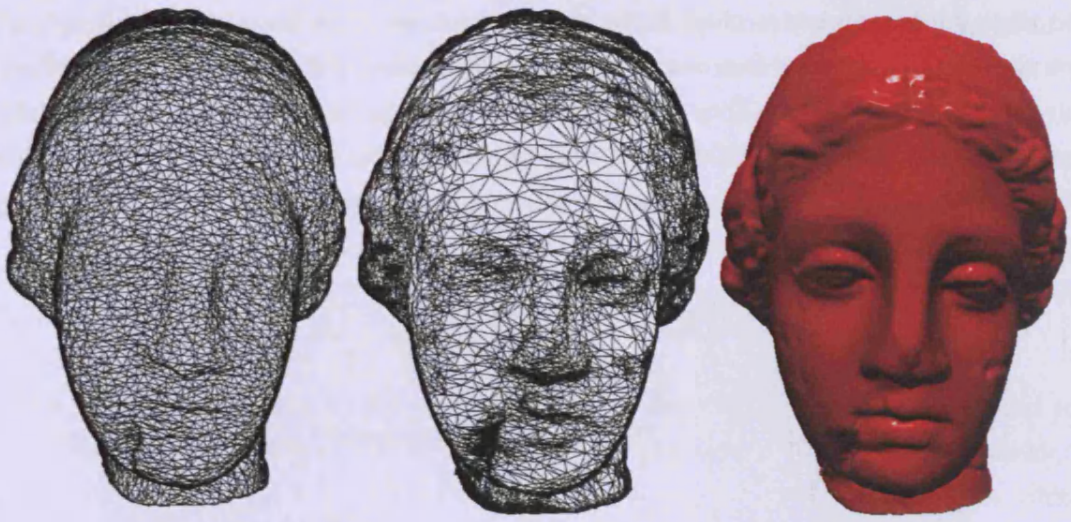


Figure 7.9: Uniform, curvature-controlled and rendered remesh of the Igea mesh.

tion 8.3 we discuss further measures that might be investigated, looking at triangle and surface distribution quality.

7.3 Robotic Prototype Painting

In this section, we introduce a prototype for robotic painting developed with Jonathan Corney and Finlay McPherson, using our high-quality sampling algorithm. Our contribution is a mesh sampling method, utilising surface textures, to generated high-quality dithered point distributions. Motivated by the need for a rapid, automated painting system for prototyping models, a non-contact optical painting system has been developed, which uses photosensitive paint, exposed by a tri-coloured laser [83, 118]. A laser is used to reduce scattering. Intensity-controllable red, green and blue laser beams are combined into a single beam. As the system is still a prototype, the colour of the laser cannot be changed during the painting process, and its intensity is limited in terms of the variation in colour that it can reproduce. This beam is transmitted via a multi-modal optical fibre through a focusing lens, mounted on the end of a robot arm, to expose a single point on the model's surface. The focusing lens is mounted on a four-axis robot with an additional two-axis rotary tilting table which holds the work piece (see Figure 7.10). The robot thus has the ability to position itself such that the laser beam can be positioned orthogonal to the tangent plane of a point on the surface, and thus expose points along their normal. The robot may also be positioned up to $\pm 45°$ from the normal in order to expose a point that may otherwise be occluded (rotation limited by the robot). The aim of the positioning is

to align the laser beam as close to the normal as possible, so that the point of exposure on the model is as close to a circle as possible; increasing the angle of incidence results in a more elliptical point. The system can currently apply only single colour images with varying intensity, of up to 300dpi resolution to work pieces, using as input VRML models that define both the surface and surface colour (i.e. a texture map) of the object being painted.

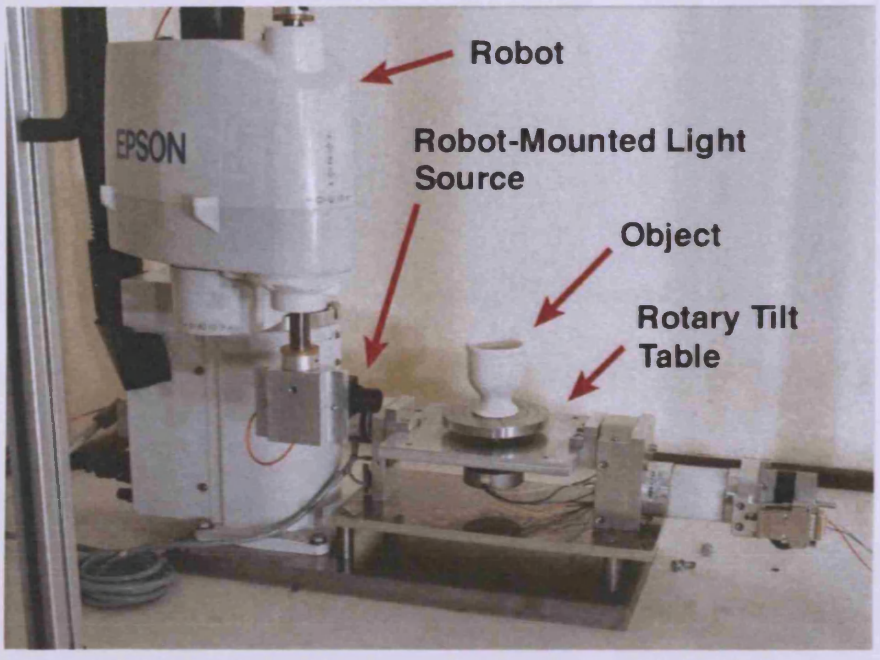


Figure 7.10: Layout of robotic cell (*courtesy of Heriot Watt University*).

A surface sampling, including point co-ordinates, surface normal and colour, describing a set of *splats* is first computed using our sampling approach. As input to the robotic painting system, the sampling is used to compute a series of robot end-effector positions, and laser intensities. From this data, a path for the robot to follow is generated so that the laser transfers the texture onto the model by exposing the light sensitive paint. The planning stage attempts to minimise the amount of sequential robot movement between positions in order to reduce the settle-time before each exposure. The robot must be allowed to settle after movement in order to achieve a sharp point of exposure on the object, and (up to a limit) the greater the movement, the more time must be allowed. Reducing settle time between exposure reduces overall painting time (see [83] for more details). This painting process requires no physical contact, so images can be applied to any shape limited only by acessibility, i.e. the ability of the robot to position and orient the laser suitably to expose each part of the surface. The robot has six degrees of freedom to facilitate exposure of overhanging and re-entrant areas.

Splats can be generated based on a laminar (layered) approach similar to that used in additive manufacture [78]. In this approach, the robot works from top to bottom of the object in small vertical increments, and for each height, a fixed number of steps is taken around the model, exposing each point to a burst of light. This layered approach to splat generation has the following shortcomings:

- Raster artifacts: the laminar approach creates a scan-line image on the model, the regularity of which the human eye can easily detect.
- Distortion: if a part of the surface being sliced is orthogonal to the axis used for laminar sampling (the z-axis), a sampling of even density is produced. However, if the surface is not orthogonal, the resulting sampling density will not be even, which can produce artifacts in the resulting image (see Figure 7.11).

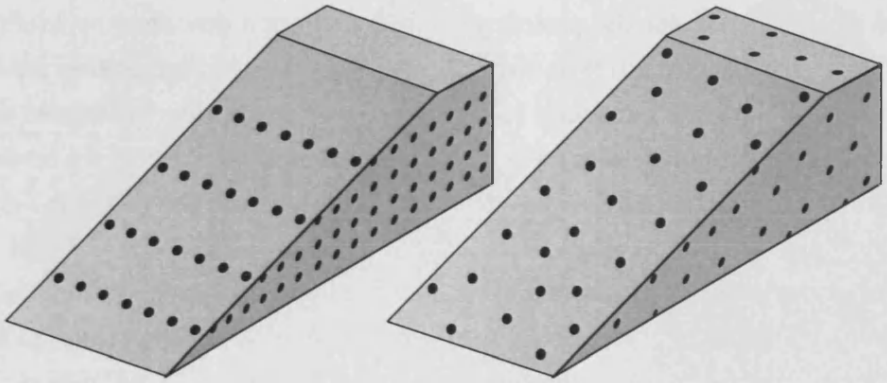


Figure 7.11: Splats using laminar slicing (left) and our low-discrepancy sampling (right).

Various solutions to these problems exist, such as varying the exposure time to increase the intensity of oblique exposed splats. Nevertheless, whilst complete coverage may be achieved in this manner, sampling fidelity is sacrificed. The importance of high quality sampling for the application of painting prototypes is two-fold. An even sampling of the input mesh (see Figure 7.11) is desirable to produce a high quality output image on the object. Parts of the surface that are almost parallel to the slicing planes are sampled very poorly (see Figure 7.11), making it difficult to produce a high quality surface exposure of a high resolution image. The equidistant laminar sampling therefore only works well on cylinder-like objects. Our approach solves this by sampling the input mesh uniformly with a low-discrepancy distribution.

The second advantage of our sampling method is the avoidance of aliasing problems in the output. Grid sampling covers a domain very evenly, but when sampling a continuous

signal, suffers from aliasing problems [87]. Aliasing occurs when a continuous function is not sampled densely enough, which is very apparent with evenly-spaced samples because of their regular, visible pattern. Sampling with non-gridded patterns of low-discrepancy samples greatly reduces the aliasing effect, as demonstrated for the application of texture filtering in [115]. Thus, we apply our approach to high-quality mesh sampling to produce a set of evenly distributed, low-discrepancy point samples, that are independent of the surface shape, to improve the output quality of the prototype painting technique. In Section 7.3.1 we describe the changes introduced into our mesh sampling approach, followed by a demonstration (see Section 7.3.2) and discussion (see Section 7.3.3) of the results.

7.3.1 Sampling Approach for Robotic Painting

In this section we expand on the mesh sampling algorithm described in Section 6.1, the output of which is used for the prototype painting solution described by McPherson et al. [83], in order to provide a robust and integrated approach to high-quality model painting. Our method extends our mesh sampling approach, allowing for texture sampling, texture-based adaptive curve subdivision, and density-controlled sampling based on texture and surface detail, and surface point dithering. Figure 7.12 demonstrates the pipeline of our sampling method, the output of which is then sent to the robot and path planning algorithms, before the model is finally painted. The figure shows the input object, which is then scanned to produce a triangle mesh. Two parameterisations are produced from this, a partial parameterisation where the texture is to be painted, and a full parameterisation for the mesh. The two parameterisations are then used to generate a Hilbert curve that is adaptive according to both. This Hilbert curve is mapped to the surface, and sampled according to the image texture provided in order to produce the input for the robotic painting process.

Colour Sampling

As an input to our algorithm, we take the input mesh M_s, and either (i) a user generated texture with texture parameterisation M_t, or (ii) colours provided in the input VRML file per triangle vertex. In case (ii) the colours can simply be linearly interpolated to compute a colour value for each vertex of the Hilbert curve H_l on the surface. In case (i), the texture is supplied explicitly via the texture parameterisation M_t, which indicates how the texture image is mapped onto the surface. M_t is a full or partial parameterisation of M_s, providing a mapping for the supplied texture onto M_s. Note that the texture parameterisation M_t is used in addition to the computed mesh parameterisation M_p (see Section 6.1.1) in order

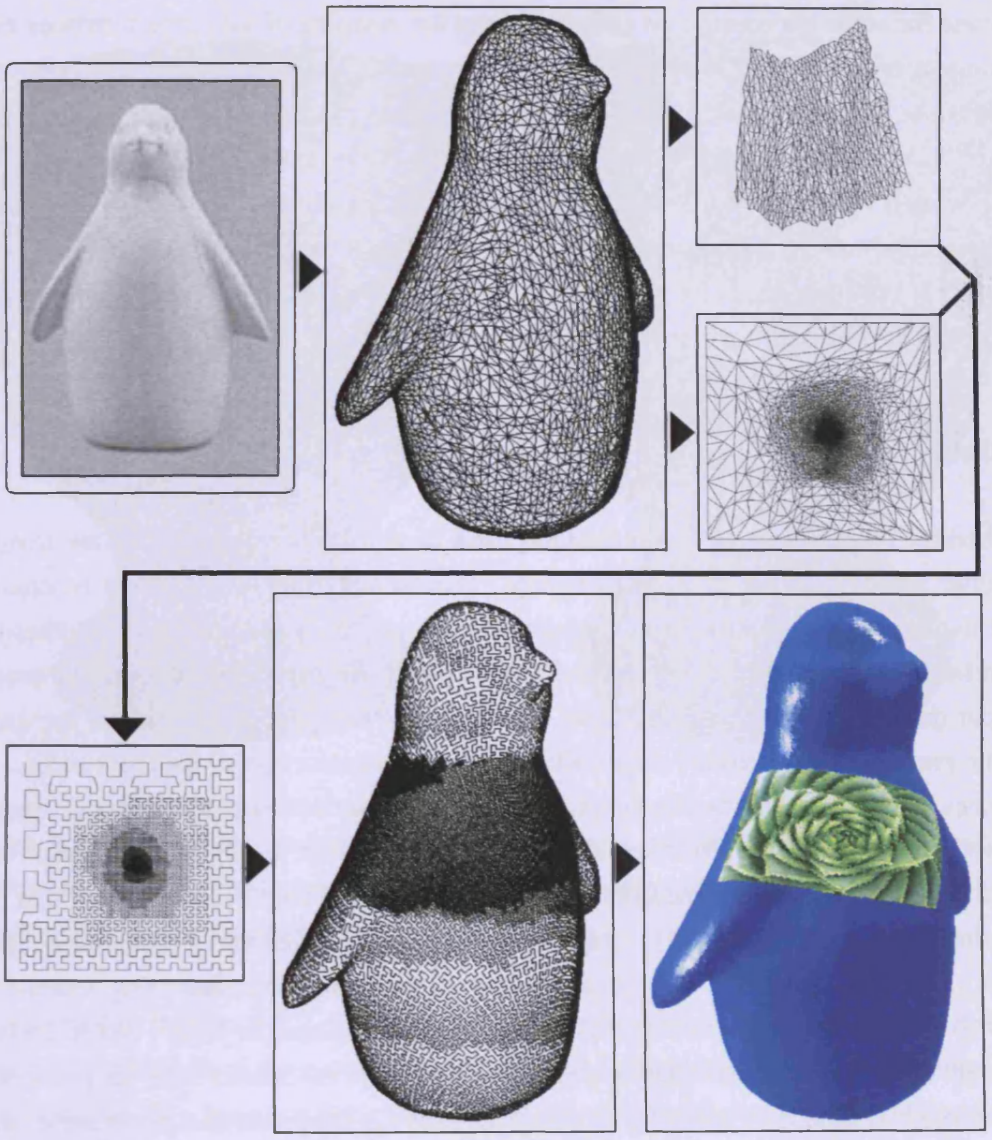


Figure 7.12: Generating point samples for the robotic painting process, in order, starting from the top left: the input model, scanned mesh, texture parameterisation (upper), mesh parameterisation (lower), adaptive Hilbert curve in the parameter domain, adaptive Hilbert curve mapped onto the original mesh, point based resampling with per-point colours and normals.

to give the user of the system more flexibility in the positioning and stretch of the texture. Colour data is extracted from the the texture image, which is then used to determine colour values for the vertices h_l of the Hilbert curve during its generation in the parameter domain of the mesh. We calculate a mapping $g(H_l) = t_l \in [0, s] \times [0, t]$, so that we can sample the parameterised texture, where s and t represent the dimensions of the texture.

We use the same barycentric co-ordinates from the mapping of the Hilbert vertices via the parameterisation f (see Section 6.1.2), but then convert them to Cartesian co-ordinates in the texture parameterisation supplied with the surface, $M_t \subset [0, s] \times [0, t]$, where M_t is the triangulated set of texture co-ordinates. A square area is assumed for each Hilbert vertex H_l, defined by the quad-tree structure of the curve, resulting in a contiguous coverage. For each vertex H_l, if the area lies within a pixel, then the vertex is assigned the pixel's colour. If multiple pixels lie within a vertex's area, the weighted average of those pixels is computed based on their coverage of the area.

Dithering

Dithering, or half-toning, is a technique used to represent a continuous tone image or graphic with a reduced palette of colours. The colour of the laser cannot be changed, and its intensity is limited in the variation in colour that it can reproduce. Dithering is therefore important due to the limited colour pallet, and also gives us control over the output quality and total painting time. To be able to reproduce an image on the surface of the prototype, we produce a spatially dithered reconstruction of the input image. The process of dithering can be considered simply as a quantisation of the colour or intensity depth of an input image. Thus, techniques generally attempt to minimise the quantisation error by assessing the local neighbourhood of a point, and alter the intensity of that point to minimise a local error [121]. Doing this for every point results in a minimised global error. In [121], a method is described to perform binary dithering using space-filling curves. Firstly, an image-covering Hilbert curve is generated in $[0, 1]^2$, and in an initial pass along the curve, a cumulative sum of intensities is calculated from the pixel values of the original image. A second pass along the curve is executed, and pixels are coloured black based on the accumulated intensity computed on the first path. The advantage of this approach is that the quantisation error is intrinsically accounted for by the first pass intensity accumulation. We use a similar approach, generating a dithered sampling of points on the output mesh to improve the image quality for our painting approach.

Our approach uses a tri-colour laser to expose a single colour point on the surface with limited intensity control. In order to provide flexibility in the output and total painting time and to improve image quality, dithering can be performed to give the appearance of a larger range of shades when painting a texture on the model. Our algorithm provides a simple solution to this: The local colour sampled from the texture stored at a Hilbert vertex H_l is converted to greyscale, which is used as an intensity. The inverse of this intensity is then used as the discrete density φ, so that more points are sampled in dark areas, and fewer in brighter areas. The result can be seen in Figure 7.13.

Figure 7.13: Dithering example using the Hilbert curve; a slight grey gradient (darker on the left, lighter on the right), and a central texture (the high density at the silhouette is due to projection).

As described in Section 6.1.2, when generating the adaptive Hilbert curve, at each level of adaptive expansion of the quad tree, we compute the number of vertices required in a particular quad, O_Q, based on that quad's coverage of the surface M_s. However, we must also consider the density functional, δ, for the surface. We compute the discrete density φ for a point on the surface of the mesh, which determines the level of subdivision of the Hilbert curve. φ is computed as the variance, σ^2, of the pixel intensity in a specified neighbourhood of a point. We now describe how we compute this discrete density, and how this is then used to determine the adaptive subdivision of the Hilbert curve.

For each quad in the Hilbert curve quad-tree, we compute its centre point η_Q, and the polygon defined by the four corners of the quad, γ_Q. Using the mesh parameterisation f, and the texture parameterisation g, we apply the compound functions $g(f(\eta_Q)) = \eta_t$ and $g(f(\gamma_Q)) = \gamma$, mapping the centre point and the quad polygon from the parameter domain M_p of the mesh, to the texture space M_t. A polygon γ is used to ensure that the proper area on M_t is represented by the Hilbert quad. γ_Q is then rasterised, resulting in the set of pixels $R_l = \{r_k : k = 1, \ldots, L\}$ around η_t, determined by γ_Q. The pixels

r_l form a neighbourhood for the central pixel sampled at η_t. A pixel r_k has n colour components for the pixel (e.g. greyscale colour model, $n = 1$), dependent on the colour model of the texture. However, in this simplified case, we have already converted the texture to greyscale. In order to compute the difference in intensity between two pixels in the neighbourhood, we first define the following function for two pixels $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_n)$,

$$\text{diff}(a, b) = \sum_{k=1}^{n} (\max(a_k, b_k) - \min(a_k, b_k)). \tag{7.3}$$

The colour difference is calculated in this way to ensure that the maximum difference is taken. The variance σ^2 of the intensity in the set of pixels R_l is then approximated by

$$\sigma^2 = \frac{1}{|R_l|} \sum_{k=1}^{|R_l|} \text{diff}(r_k, \bar{R}_l)^2 \tag{7.4}$$

where $\bar{R}_l$ is the arithmetic mean value of the pixels in R_l. The variance σ^2 of the pixels for the quad gives the discrete density approximation φ. Thus, for a quad Q, we approximate $\int_Q \delta \, dA / \int_{M_s} \varphi \, dA$ with $O_Q \geq \varphi(O_T / (\text{area}(f^{-1}(T)) / \text{area}(Q))$ vertices (see Section 6.1.2). To generalise this method to a colour, e.g. RGB, system, an approach such as that described in [96] could be employed.

7.3.2 Results

In this section, we demonstrate early results from the prototype painting system. Our sampling process is complete, but the process of getting the robot painting system fully working requires further work by the project partners. Figure 7.14 shows the output from our Hilbert curve sampling approach on an eggcup model, compared to the laminar slicing approach. The laminar slicing approach shows a clear and regular structure which the Hilbert curve approach does not. The laminar slicing sampled eggcup also displays an inconsistent vertical spacing between points, (vertical gaps can be seen between points on the base, whereas point overlap is visible near the top of the eggcup). This problem of inconsistent spacing for the laminar slicing approach highlights the advantages of our method, which can sample surfaces uniformly.

Figure 7.15 demonstrates graduated levels of gray on a piece of white card orthogonal to the laser, using varying exposure times rather than varying colour of the samples at 50dpi. Note that a low dpi is used here to demonstrate the point distributions more clearly. In a normal painting process, the sampling density used would be 300–500dpi. The laminar

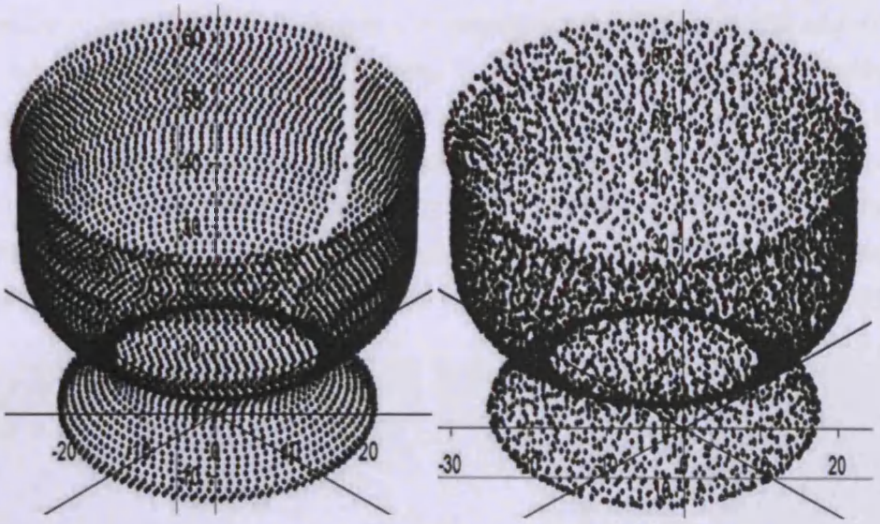


Figure 7.14: Sample point generation using laminar slices (left) and our low-discrepancy sampling (right).

slicing approach demonstrates fewer holes in the distribution due to the raster-scan approach. However, an undesirable structure is also far more apparent in the laminar slicing approach. Another problem which this image illustrates are the additional visual artifacts introduced by inaccuracies in the robot arm positioning; a vertical offset is apparent across the strip. This problem is not visible with the space-filling curve sampling approach.

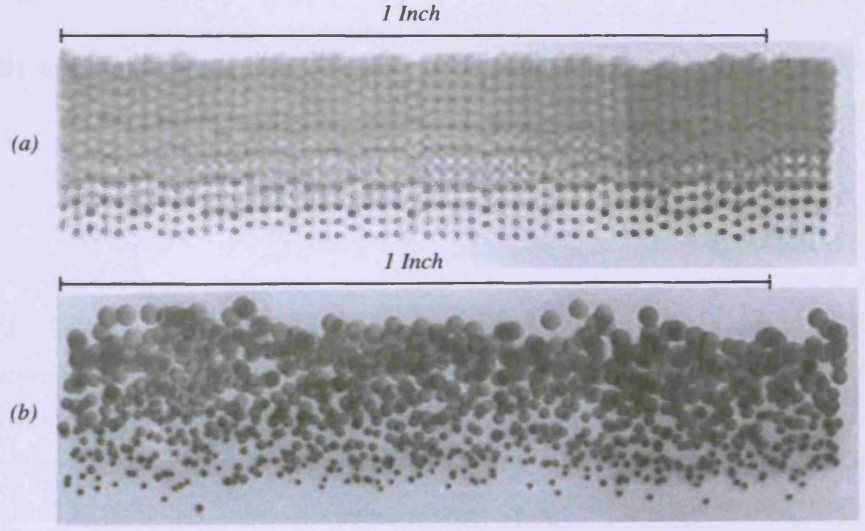


Figure 7.15: Painting of graduated gray levels using Laminar sampling (top) and our space-filling curve sampling (bottom) at 50dpi.

One main objective is to generate an even covering of the surface regardless of its orien-

tation. To test this a texture map consisting of a series of evenly spaced, horizontal strips was applied to the eggcup model mesh. Figure 7.16 shows the eggcup model, sampled at 50dpi and 100dpi for the laminar slicing, and Hilbert curve sampling approaches. The sampling generated by the laminar slicing approach can be clearly seen to produce an uneven separation towards the bottom of the egg cup (i.e. the lines get closer together as the curvature increases). However, the same texture reproduced using the Hilbert curve sampling method results in an even separation of bands regardless of the surface curvature.

(a) *(b)* *(c)* *(d)*

Figure 7.16: 2D test strips using (a) laminar slice 50dpi, (b) laminar slice 100dpi, (c) Hilbert curve 50dpi, (d) Hilbert curve 100dpi sampling.

7.3.3 Discussion

We now discuss the results obtained from our prototype painting method. The adaptive Hilbert curve sampling approach offers several advantages over the laminar slicing algorithm. For perfectly vertical surfaces and axis aligned textures, the laminar slicing approach produces better coverage (i.e. fewer holes in the distribution), though at the expense of structured rendering artifacts. Whilst requiring more samples in this case, our high-quality Hilbert curve sampling method does not introduce structure into the painted image, reducing artifacts in the output. It also demonstrated far better, even coverage with respect to varying curvature in a model, indicating that for more complex surfaces, our sampling approach is far more practical. We have also demonstrated the algorithm's ability to dither images, giving the user of the system far better control of image quality and

painting time.

7.4 Summary

We now summarise the results presented in this chapter. We have demonstrated that our sampling algorithm has advantages for point-based graphics, remeshing and prototype painting when compared to other sampling approaches. The point distributions produced for our point-based graphics application show high-quality surface representations, with efficient surface coverage, improved further with our parametric Voronoi hole-filling method, yielding better rendering results. We also demonstrate that after an initial preprocessing step, we can provide real-time levels of detail and view dependent rendering methods. These methods allow for fine-grain continuous LoD, and silhouette enhancement. We also demonstrated a remeshing approach, maintaining the topology of the original mesh, with a surface error as good as the leading approaches in the field. However, improvements in triangle quality of the output mesh may be required for some applications. Finally, we have demonstrated an approach to providing high-quality point distributions, that sample and represent a surface texture well, as part of a practical, robotic prototype painting process.

Chapter 8

Conclusions

The goal of two-manifold sampling, whilst often dependent on the application, is to produce a high quality distribution that samples the function or geometry with a correlated even distribution, whilst capturing and preserving features. Practically, this is governed by computational cost and complexity and often visual appearance. In this thesis, we have described an approach for fast generation of high-quality low-discrepancy point sample distributions for parametric surfaces and polygonal meshes. The underlying idea of our approach relies on a surface parameterisation, with considerable computation being done in the parameter domain. Due to our parameterisation approach, we reduce the computational complexity of the sampling problem, which allows us to very quickly generate high-quality density-controlled surface distributions. We further simplify the sampling problem by generating a space-filling curve to induce an ordering on the parameter domain. The space-filling curve is then mapped onto the surface, and is sampled to produce the point distribution. Differential properties of the parametric surfaces are computed to ensure accurate distribution of points, accounting for parametric stretch and density. In order to employ the same approach for polygonal meshes, robust discrete properties have been used. Overall, a robust framework has been developed for generating high quality distributions of points. The usefulness of this framework has been demonstrated for a selection of concrete applications.

Discrepancy is used throughout the thesis to assess the sampling quality of the distributions produced using our approach and compare them to alternative well known sampling methods. Visual quality is also assessed, both numerically using the blue noise criterion, and through the representation and rendering of various surfaces. A popular use of mesh sampling methods is to adjust the sampling density, often to reduce the number of samples on the surface. We therefore also assess the down-sampling error of our mesh point-sampling method relative to the original surface.

In Section 8.1 we discuss our contributions to the field of surface sampling. In Section 8.2 we discuss the problems involved in surface sampling and our approach to solving them, both in terms of generation of sample distributions and the subsequent evaluation. Finally,

in Section 8.3 we discuss our plans and suggestions for further development of this work.

8.1 Contributions

(I) We have introduced a high-quality low-discrepancy parametric surface sampling algorithm (see Chapter 4). Our novel sampling approach generates low-discrepancy sequences on parametric surfaces by sampling points along a space-filling curve. We are able to control the local sampling density, which gives greater flexibility in generating sample distributions for various applications. To devise this approach, and investigate its sampling properties, we first investigate the properties of various unit-square space-filling curves, choosing the Hilbert curve due to both construction advantages (see Section 4.2) and geometric properties (see Section 2.2.2), as shown experimentally (see Section 5.4). We then describe an algorithm to adaptively generate the Hilbert curve in order to generate a uniform coverage of the final surface (correcting for parametric stretch), with respect to the density function (see Section 4.2.1). We describe the computation of surface integrals (see Section 4.3), and employ a method similar to histogram equalisation in order to sample the curve according to area and a user-defined density function. We investigate the 1D sequences used to sample the curve (see Section 4.4). We also investigate the effects of precision loss due to the use of the Hilbert curve to map a 1D sequence to 2D, and the problems resulting from severe and non-conformal parameterisations (see Section 4.5). To address these problems, we consider a polynomial reparameterisation method and apply it as an example to a super-quadric surface that suffers from very high parametric stretch.

(II) To experimentally assess the quality of the point distributions produced using our parametric surface sampling algorithm, we compute the discrepancy of the distributions (see Chapter 5). We have extended the standard star discrepancy measure to include various non-rectangular geometric subsets in order to more thoroughly assess the distribution of points (see Section 5.1.1). We assess the discrepancy, using this extended measure, of the Hilbert and Peano curves along with different options for the 1D sequences (see Section 5.1.2). The overall best performing method, the Hilbert curve with a 1D jittered sequence is then compared to various well known low-discrepancy distributions (see Section 5.1.3). In the unit square, we show that the popular low-discrepancy distributions perform very inconsistently as the sample shapes change, highlighting the need for a wider range of shapes in discrepancy measures. We also show that our approach produces consistent results throughout all tests. We further present a discrepancy measure on a sphere, and again show the consistent low-discrepancy behaviour of our approach (see

Section 5.1.4).

(III) Moreover, we introduce a mesh sampling algorithm using adaptive Hilbert curves, extending our sampling algorithms to arbitrary genus polygonal meshes (see Chapter 6). In order to evenly distribute points on a mesh using our histogram-equalisation approach, we utilise a robust method for computing discrete approximations of surface integrals for the curve path on a triangle mesh (see Section 6.1.3). Our method builds upon accurate Voronoi area computation methods described in the literature, and further adds ways to compute the area of points with obtuse triangle neighbours, and points on the edges of the Hilbert curve. The result is a robust method for computing differential properties along the space-filling curve for arbitrary meshes, which are then used to distribute points along the curve on the mesh, to produce low-discrepancy and density controlled point distributions. In order to experimentally evaluate our method, we devise a numerical triangle mesh discrepancy measure (see Section 6.2.1), derived from the star discrepancy. We show that our method of mesh sampling generally produces point distribtuions with low-discrepancy behaviour, considerably better than a random distribution, across all of the meshes investigated (see Section 6.2.1). Discrepancy for our jittered sampling is also consistently lower than that of Poisson disc sampling, which is the output of many mesh sampling methods in the literature. We also demonstrate that our remeshing approach, when used to decimate a mesh, shows errors in line with the best remeshing methods in the literature (see Section 6.2.2). We do not, however, investigate the quality of triangles in the remeshed surface, important in, for example, finite element methods.

(IV) We demonstrate our mesh sampling method for the applications of point-based graphics, remeshing and robotic prototype painting (see Chapter 7), to show that it is also practically useful in various contexts. For our point-based graphics sampling method (see Section 7.1), we introduce a fast, global hole-filling algorithm to improve surface coverage (see Section 7.1.1). We show that the point distributions produced using our approach are very evenly spaced, providing a vastly better coverage when compared to a random distribution and the Niederreiter sequence (see Section 7.1.2). We also demonstrate rendered examples of point distributions produced using our approach, which demonstrate well-defined edges, and no rendering artifacts. We describe a real-time method for high granularity local levels of detail, such that the density of points can be modified locally on the model depending on the distance from the viewer in real-time (see Section 7.1.3). We also demonstrate an approach to view dependent rendering, allowing for higher detail along generalised silhouettes, and allowing for view-independent fixed model complexity (see Section 7.1.3). We discuss an algorithm that utilises the quad-tree structure of the Hilbert curve to provide a hierarchical surface decomposition in order to greatly speed up angle and distance computation for these methods. Further to this, we describe a remesh-

ing method using our low-discrepancy sampling method. Our approach maintains the original mesh topology (see Section 7.2), and results in high quality, density-controlled output meshes that have a very low error with respect to the original meshes, demonstrate no artifacts when rendered, and show no visible regular patterns. Finally, we describe a prototype painting method, utilising our mesh sampling algorithm, that demonstrates large improvements over the existing laminar slice sampling approach, providing an evenly distributed set of points, invariant of surface shape, capturing surface texture and colour (see Section 7.3). Our approach, due to its blue noise waveform, also avoids the highly-regular output of the laminar approach, resulting in point distributions that do not have their own underlying visible structure. We also demonstrate a density controlled dithering method that allows for improved output quality based on the limited control of the painting process (see Section 7.3.1).

8.2 Evaluation

The main focus of this work is to produce high quality point distributions on surfaces for visual applications in graphics and geometry. In Chapter 2, we discuss what we mean by high quality, focusing mainly on the property of discrepancy, the theoretical results of what can be achieved, and the distributions that minimise it. Such low-discrepancy distributions tend to be very well distributed. We therefore conjectured that a low-discrepancy point distribution would be advantageous for surface sampling and representation. However, a grid is a low-discrepancy sequence, so the goal is to find a distribution that has these properties that is not a grid, and does not therefore also suffer from aliasing effects and visual artifacts.

However, low-discrepancy distributions with these properties are not that easy to find. We therefore investigate the key construction methods of a selection of such distributions that are popular in the literature. Whilst these distributions have a low discrepancy, and do not form a regular grid, we have demonstrated, through several experiments, the shortcomings of such sampling methods: Firstly, the generalised star-discrepancy of the popular distributions investigated varied significantly as the sample shape was changed. Secondly, the popular distributions showed a power spectrum very different to that of blue noise, and a very high level of structure. Finally, the coverage of these distributions when rendered as discs in the plane was not good. These properties demonstrate the unsuitability of such point distributions for our goal of high quality sampling for visual computing. In addition to this, our review of existing applications in Chapter 3 demonstrates the importance of high quality sampling in various applications. For these reasons, and the lack of an exist-

ing ideal solution in the literature, we therefore investigated and developed an approach to produce high-quality point distributions on surfaces, applying it in the fields of point based graphics, remeshing and robotic prototype painting.

In Chapter 2 we also build up a definition of the star discrepancy, a popular numerical discrepancy analysis method. This measure is defined only for axis aligned rectangles with one point at the origin, which are somewhat limited in that they inherently favour point sequences that are based on a lattice structure. For this reason, in Chapter 5, we generalise this measure, including other sample shapes, in order to get a better understanding of the distribution of sample points. Whilst low-discrepancy point sequences generally do not alias as badly as a grid structure, it is not strictly a measure of this property, and thus we also investigate the fulfilment of the blue noise criterion by the distributions (see Section 5.3). By analysing the fulfilment of the blue noise criteria, and through visual assessment, we were able to measure the quality of point distributions for rendering and surface representation applications.

In Chapter 4, we describe our novel approach to high quality parametric surface sampling using space-filling curves. Sampling an arbitrary surface uniformly is a complex problem. Our solution provides a method of reducing the complexity of surface sampling by reducing the dimension of the domain that needs sampling to 1D. Whilst many space-filling curves exist, we use the Hilbert curve due to its better spatial-coherency, both theoretically and experimentally, its simplistic generation and low rate of subdivision (2^{2d}), and its superior discrepancy results. We do, however, also investigate the Peano curve in experiments to assess the distribution quality as point of comparison, and to verify the theoretical considerations leading to choosing the Peano curve experimentally as well. Our adaptive algorithm description focuses on the Hilbert curve, although it is widely applicable to other space-filling curves that fill the unit square. In Chapter 5 we compute the discrepancy of samples produced using the Hilbert and Peano curves on the plane, sampled with with various 1D sequences. We found that random 1D samplings, as expected, produced random samplings on the plane, and that the jittered 1D sampling of the Hilbert curve (Hilbert-jittered) produced the best results. However, a deterministic approach, such as the Niederreiter sequence might be more desirable for certain applications, such as a watermarking process for example, where the same distribution of points need to be computed every time.

We compare our best results, using the Hilbert-jittered sampling method, to existing approaches. Results indicate strongly that the distributions produced using our algorithm when measured using standard, axis-aligned rectangular subsets, whilst not performing quite as well as the well known methods, have a low discrepancy, and perform signifi-

cantly better than a random distribution. When measuring the discrepancy, we expanded the standard star discrepancy measure, using not only axis aligned rectangles as subsets, but also triangles and quarter circles. These experiments demonstrate interesting results, indicating that the well known distributions tested have low discrepancy when measured using axis-aligned rectangular test shapes, but do not behave as well using other test shapes. When measured with quarter-circles and triangles, the distributions performed equal to or worse than our method, with our method producing very similar results throughout each test. The test shapes are there to determine whether there is any significant gap in the sampling, roughly relating to the actual test shape used, i.e. to verify whether there are larger holes in the sampling with respect to these shapes. Our results indicate, primarily, that for general sampling applications, using only a single test shape does not give a clear picture of the overall distribution of the samples. This further suggests that our sampling method, which produces consistently good results across a range of test shapes, is more suitable for applications that are not specifically dependent on rectangular shapes.

In order to test this further, we devise a definition for spherical triangle discrepancy on the surface of a sphere in 3D with a fixed parameterisation. This allows us to investigate the quality of distributions on a surface that, whilst having no irregular features, does have significant parametric stretch. Results show that point distributions produced using our approach, show a discrepancy very similar to distributions that have been specifically designed for the sphere parameterisation. Testing on the sphere was useful, however, it is a simple surface, and with that, easy to deal with. A more general definition of discrepancy that not only allows us to robustly define discrepancy with a non-uniform density on arbitrary surfaces, but also to numerically evaluate such a distribution using varied density functions may be very useful. This would enable us to investigate the effect of complex density functions on the local discrepancy of the points, on a variety of surfaces. In Section 6.2.1, we investigate a mesh discrepancy measure (a linear generalisation of this approach), providing some indication for these results, although only for a uniform density. The sample shapes used may also need to be investigated, as the holes in the distributions may be of quite a different nature.

There are various ways for us to compute surface area along the curve to correctly distribute point samples using our approach. In Chapter 5, the discrepancy of distributions generated using both the most (first fundamental form) and least (triangular area) accurate area measurement methods are shown. Results showed that the accuracy of this measurement had a direct effect on the discrepancy of the point set, highlighting the importance of computing these values as precisely as possible. The lower accuracy method may be appropriate if a minimisation of pre-computation time is desired. Whilst using the first

fundamental form of the surfaces gives us the most accurate measurement, its automated symbolic computation is not the focus of this work. However, a good approximation of these computations on meshes is discussed in Section 6.1.3.

The similarity of our approach with the standard jittered sampling method is quite clear. A standard 2D jittered approach splits the domain into a series of squares, in each of which a random point is placed. Our approach in this simplified case of the plane uses a similar principle, only that the shape of the strata is a contiguous section of curve, not necessarily a square, and the degree of freedom when placing a point within such strata is more limited. Whilst a proof of the upper bound for the discrepancy of the jittered distribution is known, we do not have a *proof* of an upper bound for our curve sampling approach. Experimental discrepancy measured on the plane and on surfaces using our approach indicates that our sampling method is very robust, and produces a low-discrepancy distribution regardless of sample shape. Our approach is also superior to standard 2D jittering, as it does not require a perfect square sample distribution size, allows for full density control over the points, and works for arbitrary surfaces.

One drawback of sampling a 2D surface along a 1D curve is that for each original 1D co-ordinate, the fixed precision is halved when it is mapped onto the curve in 2D. This means that sampling in higher-dimensions directly is advantageous with respect to precision. This problem is somewhat inherent to this dimensional reduction approach. For 3D or higher dimensional space-filling curve this problem only gets worse. Whilst fundamentally this problem cannot be solved due to the precision limit on the discrete representations, multi-precision methods can provide a solution to this problem, but are inherently slow. We present the adaptive Hilbert curve as a partial solution to this, and demonstrate that it still relies on a sensible parameterisation. We also demonstrate a solution for polynomial reparameterisation for more severe cases. However, these solutions really only touch on the problems of angle and area stretch, in parameterised surface sampling, which we highlight as a limitation of our approach. In Section 8.3 we look into this problem further, suggesting approaches that address some of the problems with parameterisation in our work.

To assess the quality of our point distributions for rendering, we measure the radially averaged power spectrum density, and assess the fulfilment of the blue-noise criterion. The blue noise criterion essentially highlights how visually appealing a distribution is, and how much structure the distribution itself adds to what is being rendered. We found that our distribution did closely match that of blue noise. However, the other low-discrepancy sequences tested do not fulfil the criterion, showing a very different power spectrum, indicating a large amount of structure. This structure is undesirable for rendering, as it

detracts from the original structure or texture of the image. Hence, our approach reduces visible patterns in the distribution more than other low-discrepancy sampling approaches.

Our mesh sampling approach relies on a parameterisation of the input surface to the unit square. A unit-square parameterisation requires the mesh to have a disc topology, often requiring a cutting process. In our work we use a simplistic cutting method; by selecting a more appropriate cut, we could reduce the stretch of the mesh when parameterised, resulting in a more uniform-density adaptive curve. Whilst the output would not change, this would certainly reduce the computational cost of generating the Hilbert curve and avoid issues with extreme parametrisations. The adaptive Hilbert curve can correct for area-scaling stretch well. Angle-stretch, resulting from non-conformal parameterisations however, may have an undesirable effect on the Hilbert curve. In our mesh sampling algorithm, we therefore use a conformal, or angle preserving, parameterisation. Using local parameterisations, or parameterising the mesh to a different domain, may provide a better solution to this problem, and may simplify local-reparameterisation of the surface (see Section 8.3). However, our approach can deal with normal cases, even with these simple adaptations. Also, our work does not focus on mesh cutting and parameterisation, and as demonstrated, our approach works well with such algorithms. The output of which does not depend greatly on the particular algorithms used.

In order to sample the mesh with high-quality point distributions, robust methods are employed to compute the area, per-vertex, covered by the space-filling curve. Our method for calculating mesh curvature, however, relies on the construction of local polynomials, which are then used to calculate the local curvature. More recent literature recommends avoidance of methods such as this, due to errors that can be introduced [84], however, visual results may only be marginally improved using their approach.

We compute the discrepancy of our point distribution on various meshes using random, contiguous selections of triangles as the area subset. The distributions produced using our method demonstrate a discrepancy that behave far better than the random distribution tested, and very similarly to results for our method on the plane and the sphere. This test demonstrates that the point distributions produced by our method on a variety of triangle meshes behave with a low discrepancy. It would be useful to compare the performance of our method to competing approaches, but the poor availablity of implementations for such sampling methods makes this very difficult. However, as discussed in Section 8.1, alternative methods in the literature tend to produce a Poisson disc output, which demonstrates a worse discrepancy than the jittered output of our method.

We also measure the approximation error, based on the Hausdorff distance, between the resampled meshes that we generate using our sampling approach, and the original meshes.

Results from these experiments show that the error for the triangulated output from our point sampling approach when reducing the complexity of the model was very low; performing very similarly to the best remeshing methods in the literature. Whilst this measure provides useful insights, it would, however, be more useful to measure the approximation error with respect to fundamental surface properties, such as area, normal direction, surface curvature, and possibly salient surface features. These properties would give a better indication as to how much the shape of the surface has changed in the remeshing process. In Section 8.3 we discuss a possible extension of the Hausdorff distance to consider these properties.

We build upon our sampling method to produce a high-quality distribution with good surface coverage for point-based rendering. As discussed in Chapter 2, our sampling approach splits the domain into a series of strata, randomly placing a point within each, and thus reducing the variance of the distribution when compared to an uncorrelated random distribution. Whilst this method enforces good uniformity in the distribution, by nature it is still probabalistic, and thus, whilst an upper-bound of the maximum distance betweeen points may be computed, each distribution will be different. For the application of point based rendering, it is important to generate a distribution of points that are uniformly spaced (without, however, being sampled on a grid). Thus we describe an approach to fill the holes in the distribution that may appear between points that are highly spaced apart: a global Voronoi hole-filling algorithm that is able to ensure a maximum hole size.

Experimentally, we investigate splat coverage for our sampling method in the plane, with and without the hole-filling algorithm, comparing the results to the Niederreiter sequence and a random distribution. Results show that our standard sampling approach achieves a much better coverage than the Niederreiter and random sampling methods. Our algorithm with Voronoi hole-filling produces even better results, with no visible holes in the coverage. However, other than comparing our approach to a broader selection of competitive methods, there are various other properties that would be interesting to investigate. Firstly, we currently scale splat sizes using the inverse of the surface curvature. I.e. on the plane, every splat has the same diameter. In order to take into account some variation in the distance between splats, it may be appropriate to scale the splat size based on the Voronoi radius to the neighbouring points, not just on the inverse of the surface curvature.

In Chapter 7, we also demonstrate real-time LoD and VDR techniques. Our approach to LoD provides fine granularity continuous LoD that can vary locally over an individual model. Our VDR approach provides real-time point reduction and silhouette enhancement. After an initial pre-processing stage, we demonstrate real-time frame rates for these applications. We also discuss an approach to utilise the quad-tree structure of the

space-filling curve to provide a heirarchical decomposition of the point distribution, allowing us to considerably reduce the number of distance and angle computations required in this approach. These results, along with the high-quality rendered examples indicate that point-based rendering and surface representation is a very practical and natural application of our sampling approach.

We also consider remeshing, whereby we triangulate the point distribution generated using our mesh sampling algorithm. We produce a point distribution on a mesh surface using our mesh sampling algorithm, that is then triangulated to produce a remeshed surface.

We present a remeshing approach based upon our mesh sampling algorithm. As the mesh may require cutting to a disc topology, in order to insure that the output mesh has the same topology as the original mesh, we use a basic stitiching approach that requires little computation, but does result in an insertion of additional points—supplemental to the set of points generated using our algorithm. One possible approach to avoid this stitching solution (and thus avoid the modification of the distribution) would be to locally parameterise the mesh, and produce a global map of the parameterisations. Whilst introducing more joins between the patches, it would simplify each joint significantly. However, generating a high-quality distribution of points that is not disrupted along the cut lines may not be a trivial problem.

One problem not addressed algorithmically in this work is that of triangle quality. Many approaches exist to improve the quality of a triangulation, including removal of undesirable triangles, flipping of triangle edges, and the addition of extra vertices in the triangulation. Whilst these modifications can certainly improve the quality of the mesh with respect to different application requirements (such as finite element analysis methods), they are not within the scope of this work. However, our results demonstrate that the remeshed surfaces produced by our algorithm very accurately approximates the original mesh, and produce very high quality renders.

Finally, in Chapter 7, we introduce our application of robotic prototype painting. The full sampling approach, based upon on our algorithm used for the robotic painting process, is not yet finished in terms of finalising the robot, and thus we have only demonstrated a limited number of painted objects using the original laminar slicing approach as well as our method. However, our sampling method is completed, and the quality of the output from renders and some initial tests with the robot can already be assessed. These early results show that the point distributions generated using our mesh sampling algorithm produces high-quality output. Due to its raster-scan like distribution, resulting in very evenly spaced samples (and therefore little overlap), the laminar slicing sampling approach employed previously has an advantage in terms of the DPI required. However, as the exposed

points are circular, not square, the raster-scan like output of the laminar slicing approach results in a regular grid of holes between the points. Thus, with the same DPI, whilst the laminar slicing approach will result in fewer holes, the regularity of these holes is highly structured, and thus clearly visible to the human eye. Structured distributions have been shown to result in high spatial-frequency patterns, detracting from the original texture, and are thus undesirable in this situation. Unlike the laminar slicing approach, our approach creates a uniform, correlated, sampling density across the entire model. The original laminar slice approach can only produce a uniform sampling on a cylinder. Our approach, however, considerably improves on the existing technique in terms of the complexity of surfaces that can be uniformly painted, allowing for high-quality sampling on arbitrary surfaces. The sequential path produced for the robot is optimised to reduce the amount of settle-time required after the arm moves, before painting. Both the laminar slice and our point sampling distributions require similar settle-time, however, due to alignment inconsistencies and the poor handling of high-curvature features by the laminar slice approach.

An improvement to the current approach would be to have better control over the size of each point, via the focus of the lasers and the exposure time of each individual point in order to provide better surface coverage. Further to this, applying the Voronoi hole-filling algorithm developed for the point based graphics application may provide better surface coverage, and thus higher quality output. In terms of assessment, it would be useful to be able to numerically evaluate the quality of the textured output on the final surface. However, due to the nature of the output, human reviewing of the output may actually provide the most appropriate feedback.

8.3 Future Work

In this section, we discuss the issues and questions arising from the results of this thesis to be addressed in future work. We have described algorithms to generate density controlled point distributions on surfaces. When we refer to density, we have thus far only considered it as an arbitrary scalar property on the sruface. However, to achieve the most efficient placement of points, or to describe certain features, such as sharp edges, it is advantageous to have anisotropic control over the point distribution.

The anisotropy of a surface property such as density is described by certain properties of its second derivative; typically limited to surface properties given by a quadratic form (as otherwise the second derivative may be more complicated). Controlling the anisotropy directly by prescribing a Hessian (the matrix of all partial second derivatives) for the surface may be very useful for some applications, e.g. to sample more densly along the direction

of a sharp edge. However, one issue is how well the anisotropy and direction dependency of the distribution is actually preserved by our discrete sampling. The square structure of the Hilbert curve is likely to impose a limit upon this. Thus, instead of solving a partial differential equation to compute a density for our sampling approach from a prescribed Hessian, we propose an alternative approach to density adjustment. In particular, this may be useful for more exact control over where the points are placed, for example, in order to avoid overlapping ellipses for point-based rendering. We suggest a solution to this problem, by means of computing an 'anisotropic' curve for sampling; the properties of the curve explicitly enforcing the local, anisotropic, density of the point distribution.

The approach would still be based on the Hilbert curve, generating an adaptive quad-tree based on the local area on the surface and according to surface curvature. Every quad on the surface is therefore approximately the same size. A set of particles should be generated within each leaf node based on the local density. For each particle, its principal curvatures are computed to determine anisotropic repulsion radii, which are then scaled relative to the local area of the parametric quad on the surface. Particles are then locally anisotropically relaxed within each quad, in a similar manner to that described by [114] and [73]. Relaxation forces between particles from neighbouring quads are computed, and inter-quad particle transition and scaling of the forces of each particle based on their new local area must take place. This process of particle relaxation would then be iterated until a certain minimal amount of relaxation occurred during an iteration. Particle transitions between quads must be considered. Once the particle simulation is complete, a Hilbert curve ordering of the quads may be used globally, and a local path within each cell computed to yield a spatially coherent global ordering of the particles. This curve would then be sampled as described in Section 4.3. This approach may then be accelerated with preprocessing and resampling methods. Further to this, this approach lends itself to GPGPU methods to accelerate the independent local relaxation force computations. The relaxation within each quad may be computed on an individual GPU processing element, with the set of all quads being divided up among the available processing elements.

We are also interested in further methods to numerically investigate the quality of the point distributions produced by our methods. Metro gives us an indication of how well a mesh, produced using a triangulation of points from our mesh sampling algorithm, preserves the overall shape of an input surface. 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 resampled surface. In [72], a *depth weighted Hausdorff distance* is proposed, using local surface curvature variance as weight. We expect that such a measure could provide useful numerical results for the assessment of

surface resampling, and may better highlight feature loss due to grid sampling. Another approach to analysing the distribution of points would be to assess the distribution of errors when resampling the surface, in terms of the variance, arithmetic mean and the type of distribution (e.g. Gaussian).

The numerical discrepancy measurements discussed in this work will not provide any useful results as to the quality of an anisotropic distribution of points. The general issue is how to assess how well an arbitrary density (which may be isotropic or anisotropic) has been discretised, assessing the size of holes in the sampling (measured by some sampling shape). This is a complex problem, even in the plane when using simple shapes. As a simplified solution to experimentally investigate the quality of distributions generated by an anisotropic sampling approach (such as that introduced in this section), we suggest the consideration of a local, elliptical neighbourhood of each point in the distribution. A Gabriel graph is constructed as a set of edges, whereby an edge is added between two vertices if the circle with the diameter of the length of that edge contains no other vertices. Park et al. [97] describe an approach for computing elliptical Gabriel graphs, whereby the circle is replaced with an ellipse, the radii of which are governed by the local anisotropy. We suggest utilising elliptical Gabriel graphs as a method to analyse the anisotropy of a point distribution based on surface curvature, though further development of this approach would be required to provide numerical results. An approach such as this may help to assess how well the directions and the two densities in these directions are realised in the discretised distribution of points. However, this does not provide a solution to a more generalised discrepancy for non-constant densities. A further, related, issue is the selection of a suitable density for a given set of requirements (e.g. to preserve salient features where saliency is not clear).

In our approach, we rely on a unit-square parameterisation of the input surface that we wish to resample. Using this parameter domain allows us to generate, and sample, a global unit square space-filling curve. Whilst this solution allows us to use a global sampling approach, it becomes a problem for high genus mesh surfaces; the mesh must be cut into a disc topology. The cutting process itself must be considered, varying greatly in computation based on the quality of the cut required. It also introduces seams into the sampling, where the cut-lines of the global parameterisation meet. If a remeshing output is required, complex or distribution-altering methods must be employed to remove the seams. The global parameterisation itself is then a computationally expensive process, highlighted in Section 6.2.3, where the run-time of the parameterisation is often orders of magnitude slower than the rest of the sampling process. Sampling a global parameterisation may also require extreme local density control to achieve a uniform point distribution on the final surface, or reparameterisation. A possible solution to this problem would

be to locally parameterise patches of the surface, generating space-filling curves for each patch, and simply linking the patches together. A method of defining surfaces locally using moving least square polynomials is described by Alexa et al. [2]. In this approach, the polynomials are sampled with evenly spaced points. We suggest using this approach to define local polynomials, which we would then sample using local space-fillling curves to provide density controlled low-discrepancy distributions. This approach would avoid the global parameterisation, and if the output required is a mesh, the process of stitching local patches together has been largely solved [64]. A further solution may be to avoid parameterisation all together, sampling in higher dimensions, and thus also avoiding the problem of precision loss.

Bibliography

[1] Bart Adams. *Point-Based Modeling, Animation and Rendering of Dynamic Objects*. PhD thesis, Katholieke Universiteit Leuven, 2006.

[2] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Point set surfaces. In: Thomas Ertl, Ken Joy, and Amitabh Varshney, editors, *Proc. Conf. Visualization*, pp. 21–28. IEEE Computer Society, Piscataway, NJ, 2001.

[3] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Computing and rendering point set surfaces. *IEEE Trans. on Visualization and Computer Graphics*, 9(1):3–15, 2003.

[4] Marc Alexa, Markus Gross, Mark Pauly, Hanspeter Pfister, Marc Stamminger, and Matthias Zwicker. In: Point-based computer graphics. *SIGGRAPH Course Notes*. ACM Press, New York, USA, 2004.

[5] Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Lévy, and Mathieu Desbrun. Anisotropic polygonal remeshing. Jessica Hodgins and John C. Hart, editors, *ACM Trans. Graphics*, 22(3):485–493, 2003.

[6] Pierre Alliez, Ucelli Giuliana, Craig Gotsman, and Marco Attene. Recent advances in remeshing of surfaces. Research report, *http://cgal.inria.fr/Publications/2005/AUGA05* [accessed 03–March–09], AIM@SHAPE EU Network, Springer, 2005.

[7] Pierre Alliez, Mark Meyer, and Mathieu Desbrun. Interactive geometry remeshing. *ACM Trans. Graphics*, 21(3):347–354, 2002.

[8] Nina Amenta, Marshall Bern, and Manolis Kamvysselis. A new voronoi-based surface reconstruction algorithm. In: *SIGGRAPH*, pp. 415–421. ACM Press, New York, USA, 1998.

[9] W. Anotiapiaboon and S. S. Makhanov. Tool path generation for five-axis nc machining using adaptive space-filling curves. *International Journal of Production Research*, 43(8):1643–1665, 2005.

[10] Maurice Stevenson Bartlett. *An introduction to stochastic processes, with special reference to methods and applications*. Cambridge University Press, 2nd edition, 1966.

[11] J. Beck and W. W. L. Chen *Irregularities of distribution*, Cambridge Tracts in Mathematics. Cambridge University Press, Cambridge, 1987.

[12] Marshall Bern and David Eppstein. Mesh Generation and Optimal Triangulation. In: F. K. Hwang and D. Z. Du, editors, *Computing in Euclidean Geometry*. World Scientific, 1992.

[13] Sushil Bhakar, Liang Luo, and Sudhir P. Mudur. View dependent stochastic sampling for efficient rendering of point sampled surfaces. In: *WSCG*, pp. 49–56. 2004.

[14] Z. Bi, M. Lang, and Y.T. Sherman. A framework for cad and sensor-based robotic coating automation. *IEEE Trans. on Industrial Informatics*, 3(1):84–91, 2007.

[15] T. Bially. Space-filling curves: Their generation and their application to bandwidth reduction. *IEEE Trans. Information Theory*, 15(6):658–664, 1969.

[16] A. Bogomjakov and C. Gotsman. Universal rendering sequences for transparent vertex caching of progressive meshes. *Computer Graphics Forum*, 21(2):137–137, 2002.

[17] Mario Botsch and Leif Kobbelt. Resampling feature and blend regions in polygonal meshes for surface anti-aliasing. *Computer Graphics Forum*, 20(3):402–410, 2001.

[18] Mario Botsch and Leif Kobbelt. A remeshing approach to multiresolution modeling. In: *Symp. on Geometric Processing*, pp. 189–196. 2004.

[19] Mario Botsch and Leif P. Kobbelt. A robust procedure to eliminate degenerate faces from triangle meshes. In: T. Ertl, B. Girod, G. Greiner H. Niemann, and H.-P. Seidel, editors, *Proc. Vision Modeling and Visualization Conference*, Aka GmbH, Berlin, pp. 283–290. 2001.

[20] Paul Bratley and Bennett L. Fox. Algorithm 659: Implementing Sobol's quasirandom sequence generator. *Journal of Transactions on Mathematical Software*, 14(1):88–100, 1988.

[21] Paul Bratley, Bennett L. Fox, and Harald Niederreiter. Implementation and tests of low discrepancy sequences. *ACM Trans. Modeling and Computer Simulation*, 2(3):195–213, 1992.

[22] Paul Bratley, Bennett L. Fox, and Harald Niederreiter. Algorithm 738: Programs to generate Niederreiter's low-discrepancy sequences. *Journal of Transactions on Mathematical Software*, 20(4):494–495, 1994.

[23] Greg Breinholt and Christoph Schierz. Algorithm 781: Generating hilbert's space-filling curve by recursion. *ACM Trans. Mathematical Software*, 24(2):184–189, 1998.

[24] Kevin Buchin. *Organizing Point Sets*. PhD thesis, Free University Berlin, 2007.

[25] A.R. Butz. Space filling curves and mathematical programming. *Information and Control*, 12:314–330, 1968.

[26] H. Chi, M. Mascagni, and T. Warnock. On the optimal halton sequence. *Mathematics and Computers in Simulation*, 70(1):9–21, 2005.

[27] P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms. *Computers and Graphics*, 22(1):37–54, 1998.

[28] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: Measuring error on simplified surfaces. *Computer Graphics Forum*, 17(2):167–174, 1998.

[29] B. Cipra. In math we trust. In: *What's Happening in the Mathematical Sciences (1995–1996)*, volume 3. American Mathematical Society, Providence, RI, 1996.

[30] J. Clark. Hierarchical geometric models for visible surface algorithms. *Communications of the ACM*, 19(10):547–554, 1976.

[31] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Trans. Graphics*, 5(1):51–72, 1986.

[32] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. *Computer Graphics*, 1803:137–145, 1984.

[33] J. G. Ven Der Corput. Verteilungsfunktionen I. *Nederl. Akad. Wetensch. Proc. Ser. B*, 38(38):813–821, 1935.

[34] Jianjun Cui and Willi Freeden. Equidistribution on the sphere. *SIAM Journal of Scientific Computing*, 18(2):595–609, 1997.

[35] R. Dafner, D. Cohen-Or, and Y. Matias. Context-based space filling curves. M. Gross and F. R. A. Hopgood, editors, *Computer Graphics Forum*, 19(3):209–218, 2000.

[36] T. J. G. Davies, Ralph R. Martin, and Adrian Bowyer. Computing volume properties using low-discrepancy sequences. *Geometric Modelling*, pp. 55–72. 1999.

[37] P. Davis. *Interpolation and Approximations*. Dover, 1975.

[38] Dobkin and Gunopulos. Computing the rectangle discrepancy. In: *Annual Symp. Computational Geometry*, 1994.

[39] David P. Dobkin, David Eppstein, and Don P. Mitchell. Computing the discrepancy with applications to supersampling patterns. *ACM Trans. on Graphics*, 15(4):354–376, 1996.

[40] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In: *SIGGRAPH*, pp. 173–182. ACM Press, New York, USA, 1995.

[41] C. Faloutsos and S. Roseman. Fractals for secondary key retrieval. In: *Symp. Principles of Database Systems*, pp. 247–252. 1989.

[42] Michael S. Floater. Parametrization and smooth approximation of surface triangulations. *Journal of Computer-Aided Geometric Design*, 14(4):231–250, 1997.

[43] Michael S. Floater and Kai Hormann. Surface parameterization: a tutorial and survey. In: N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in multiresolution for geometric modelling*, pp. 157–186. Springer, 2005.

[44] C. Gotsman and M. Lindenbaum. On the metric properties of discrete space-filling curves. *IEEE Trans. Image Processing*, 5(5):794–797, 1996.

[45] J. P. Grossman and William J. Dally. Point sample rendering. In: George Drettakis and Nelson L. Max, editors, *Rendering Techniques*, pp. 181–192. Springer, 1998.

[46] Xianfeng Gu, Steven J. Gortler, and Hugues Hoppe. Geometry images. *ACM Trans. Graphics*, 21(3):355–361, 2002.

[47] Xianfeng Gu and Shing-Tung Yau. Computing conformal structures of surfaces. *Communications in Information Systems*, 2(2):121–146, 2002.

[48] Xianfeng Gu and Shing-Tung Yau. Global conformal surface parameterization. In: *Symp. Geometry Processing*, pp. 127–137, 2003.

[49] Igor Guskov, Kiril Vidimce, Wim Sweldens, and Peter Schröder. Normal meshes. In: Kurt Akeley, editor, *SIGGRAPH*, pp. 95–102. ACM Press, New York, USA, 2000.

[50] J. H. Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerical Mathematics*, 2(2):84–90, 1960.

[51] J. H. Halton. A retrospective and prospective survey of the monte carlo method. *SIAM Review*, 12:1–63, 1970.

[52] J. H. Halton and G. B. Smith. Algorithm 247: Radical-inverse quasi-random point sequence. *Communications of the ACM*, 7(12):701–702, 1964.

[53] J. H. Halton and S. K. Zaremba. The extreme and l2 discrepancies of some plane sets. *Monatsh. Math.*, pp. 316–328. 1969.

[54] J. M. Hammersley. Monte Carlo methods for solving multivariable problems. *Annals of the New York Academy of Sciences*, 86:844–874, 1960.

[55] J. M. Hammersley and D. C. Handscomb. *Monte Carlo Methods*. Methuen, London, 1964.

[56] Paul S. Heckbert. Fast surface particle repulsion. Technical report, CMU Computer Science, October 26 1997. In: *SIGGRAPH Course Notes*, 1998.

[57] Fred J. Hickernell. A generalized discrepancy and quadrature error bound. *Mathematics of Computation*, 67(221):299–322, 1998.

[58] Hugues Hoppe. View-dependent refinement of progressive meshes. *Computer Graphics*, 31:189–198, 1997.

[59] A. K. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall, Englewood Cliffs, NJ, 1989.

[60] Cha-Soo Jun, Kyungduck Cha, and Yuan-Shin Lee. Optimizing tool orientations for 5-axis machining by configuration-space search method. *Computer-Aided Design*, 35:549–566, 2002.

[61] Aravind Kalaiah and Amitabh Varshney. Modeling and rendering of points with local geometry. *IEEE Trans. Visualization and Computer Graphics*, 9(1):30–42, 2003.

[62] Ibrahim Kamel and Christos Faloutsos. On packing R-trees. In: Bharat Bhargava, Timothy Finin, and Yelena Yesha, editors, *Proc. 2nd International Conf. Information and Knowledge Management*, pp. 490–499. ACM Press, New York, USA, 1993.

[63] Alexander Keller. Instant radiosity. In: *SIGGRAPH*, pp. 49–56. ACM Press, New York, USA, 1997.

[64] Andrei Khodakovsky, Nathan Litke, and Peter Schr"oder. Globally smooth parameterizations with low distortion. *ACM Trans. on Graphics*, 22(3):350–357, 2003.

[65] L. Kobbelt and M. Botsch. A survey of pointbased techniques in computer graphics. *Computers and Graphics*, 28(6):801–814, 2004.

[66] Leif Kobbelt, Thilo Bareuther, and Hans-Peter Seidel. Multiresolution shape deformations for meshes with dynamic vertex connectivity. *Computer Graphics Forum*, 19(3), 2000.

[67] Leif Kobbelt, Swen Campagna, and Hans-Peter Seidel. A general framework for mesh decimation. In: *Graphics Interface*, pp. 43–50. 1998.

[68] Ladislav Kocis and William J. Whiten. Computational investigations of low-discrepancy sequences. *ACM Trans. Mathematical Software*, 23(2):266–294, 1997.

[69] Gerhard Larcher, Wolfgang Ch. Schmid, and Reinhard Wolf. Digital (t,m,s)-nets, digital (T,s)-sequences, and numerical integration of multivariate walsh series. In: *1st Salzburg Minisymposium*, pp. 75–107. 1994.

[70] Jonathan Ledlie. Hilbert curves. *http://www.eecs.harvard.edu/ jonathan/p2p/hilbert.pdf* [accessed 03–March–09], Harvard University.

[71] D. T. Lee and B. J. Schachter. Two algorithms for constructing a delaunay triangulation. *International Journal of Computer and Information Science*, 9(3):219–242, 1980.

[72] Yeung-hak Lee and Jae-Chang Shim. Curvature based human face recognition using depth weighted hausdorff distance. In: *International Conf. Image Processing*, pp. 1429–1432. 2004.

[73] Florian Levet, Juliem Hadim, Patrick Reuter, and Christophe Schlick. Anisotropic sampling for differential point rendering of implicit surfaces. In: *WSCG*, pp. 109–116. 2005.

[74] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The digital michelangelo project: 3D scanning of large statues. In: Kurt Akeley, editor, *SIGGRAPH*, pp. 131–144. ACM Press, New York, USA, 2000.

[75] Xueqing Li, Wenping Wang, Ralph R. Martin, and Adrian Bowyer. Using low-discrepancy sequences and the crofton formula to compute surface areas of geometric models. *Computer-Aided Design*, 35(9):771–782, 2003.

[76] Yu-Shen Liu, Jun-Hai Yong, Hui Zhang, Dong-Ming Yan, and Jia-Guang Sun. A quasi-monte carlo method for computing areas of point-sampled surfaces. *Computer-Aided Design*, 38(1):55–68, 2006.

[77] David Luebke, Benjamin Watson, Jonathan D. Cohen, Martin Reddy, and Amitabh Varshney. *Level of Detail for 3D Graphics*. Elsevier Science, New York, USA, 2002.

[78] Ren C. Luo and Yawei Ma. A slicing algorithm for rapid prototyping and manufacturing. In: *Robotics and Automation*, pp. 2841–2846. 1995.

[79] B. Mandelbrot. *The fractal geometry of nature*. Freeman, San Francisco, 1982.

[80] Ricardo Marroquim, Martin Kraus, and Paulo Roma Cavalcanti. Efficient point-based rendering using image reconstruction. In: M. Botsch, R. Pajarola, B. Chen, and M. Zwicker, editors, *Symposium on Point Based Graphics*, Eurographics Association. pp. 101–108. 2007.

[81] Jerrold E. Marsden and Michael J. Hoffman. *Basic Complex Analysis*, 3rd edition. W.H.Freeman & Co Ltd, 1999.

[82] Jerrold E. Marsden and Anthony J. Tromba. *Vector Calculus*, 5th edition. W.H.Freeman & Co Ltd, 2003.

[83] F.N. McPherson, R.C.W. Sung, and J.R. Corney. Path planning for automated robot painting. In: *ASME International Design Engineering Technical Conferences*, Las Vegas, Nevada, USA, 2007.

[84] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. Discrete differential geometry operators for triangulated 2-manifolds. *VisMath III*, pp. 35–57. 2002.

[85] M. J. Milroy, C. Bradley, and G. W. Vickers. Segmentation of a wrap-around model using an active contour. *Computer-Aided Design*, 29(4):299–320, 1997.

[86] D. R. Mitchell. Generating antialiased images at low sampling densities. M. C. Stone, editor, *SIGGRAPH* Computer Graphics, 21(4):65–72, 1987.

[87] Don P. Mitchell. Consequences of stratified sampling in graphics. In: *SIGGRAPH*, pp. 277–280. 1996.

[88] G. M. Morton. A computer-oriented geodetic data base and a new technique in file sequencing. Technical report, IBM Ltd. Ottawa, Canada, 1966.

[89] Diego Nehab and Philip Shilane. Stratified point sampling of 3D models. In: Markus Gross, Hanspeter Pfister, Marc Alexa, and Szymon Rusinkiewicz, editors, *Symposium on Point Based Graphics*, pp. 49–56. Eurographics Association, 2004.

[90] H. Niederreiter. Quasi-monte carlo methods and pseudo-random numbers. *Bulletin of the AMS*, 84:957–1041, 1978.

[91] H. Niederreiter. Point sets and sequences with small discrepancy. *Monatshefte für Mathematik*, 104:273–337, 1987.

[92] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. SIAM, 1992.

[93] Harald Niederreiter. Low-discrepancy and low-dispersion sequences. *Journal of Number Theory*, 30:51–70, 1998.

[94] H. Nyquist. Certain topics in telegraph transmission theory. *Trans. A.I.E.E.*, pp. 617–644. 1928.

[95] Yutaka Ohtake, Alexander Belyaev, Marc Alexa, Greg Turk, and Hans-Peter Seidel. Multi-level partition of unity implicits. Jessica Hodgins and John C. Hart, editors, *ACM Trans. on Graphics*, 22(3):463–470. ACM Press, 2003.

[96] Victor Ostromoukhov and Roger D. Hersch. Multi-color and artistic dithering. In: *SIGGRAPH*, pp. 425–432. ACM Press, New York, USA, 1999.

[97] Joon C. Park, Hayong Shin, and Byoung K. Choi. Elliptic gabriel graph for finding neighbors in a point set and its application to normal vector estimation. *Computer-Aided Design*, 38:619–626, 2006.

[98] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In: *SIGGRAPH*, pp. 335–343. ACM Press, New York, USA, 2000.

[99] Loren K. Platzman and III John J. Bartholdi. Spacefilling curves and the planar travelling salesman problem. *Journal of the ACM*, 36(4):719–737, 1989.

[100] Joao Proenca, Joaquim A. Jorge, and Mario C. Sousa. Sampling point-set implicits. M. Botsch, R. Pajarola, B. Chen, and M. Zwicker, editors, *Symp. Point Based Graphics*, pp. 11–18. Eurographics Association, 2007.

[101] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, 1990.

[102] Lijun Qu and Gary W. Meyer. Perceptually driven interactive geometry remeshing. In: Marc Olano and Carlo H. Séquin, editors, *Proc. Symp. Interactive 3D Graphics*, pp. 199–206. ACM, 2006.

[103] Jonathan A. Quinn, Frank C. Langbein, and Ralph R. Martin, and Gershon Elber. Density-controlled sampling of parametric surfaces using adaptive space-filling curves. In: *Proc. Geometric Modeling and Processing*, LNCS, 4077, pp. 658–678. Springer, 2006.

[104] Jonathan A. Quinn, Frank C. Langbein, and Ralph R. Martin. Low-discrepancy point sampling of meshes for rendering. In: M. Botsch, R. Pajarola, B. Chen, and M. Zwicker, editors, *Symp. Point Based Graphics*, pp. 19–28. Eurographics Association, 2007.

[105] R. D. Richtmyer. The evaluation of definite integrals, and a quasi-monte carlo method based on the properties of algebraic numbers. Technical report, Los Alamos Science Labs, Los Alamos, 1951.

[106] J. Rovira, P. Wonka, F. Castro, and M. Sbert. Point sampling with uniformly distributed lines. In: *Symp. Point Based Graphics*, pp. 109–118. Eurographics Association, 2005.

[107] Laurent Saboret, Pierre Alliez, and Bruno Lévy. Planar parameterization of triangulated surface meshes. CGAL Editorial Board, editor, *CGAL-3.2 User and Reference Manual*. CGAL, 2006.

[108] Hans Sagan. *Space-Filling Curves*. Springer-Verlag, New York, 1994.

[109] L. A. Santalo. Integral geometry. Studies in Global Geometry and Analysis, *The Press of The Mathematical Association of America*, pp. 147—167. 1967.

[110] W. M. Schmidt. Metrical theorems on fractional parts of sequences. *Trans. American Mathematical Society*, 110:493–518, 1964.

[111] C. E. Shannon. Communication in the presence of noise. *Proc. of the Institute of Radio Engineers*, 37:10–21, 1949.

[112] Alla Sheffer. Spanning tree seams for reducing parameterization distortion of triangulated surfaces. In: *Shape Modeling International*, pp. 61–66. IEEE Computer Society, 2002.

[113] Jonathan Richard Shewchuk. Triangle: Engineering a 2D quality mesh generator and delaunay triangulator. In: Ming C. Lin and Dinesh Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, LNCS, 1148, pp. 203–222. Springer, 1996.

[114] Kenji Shimada, Atsushi Yamada, and Takayuki Itoh. Anisotropic triangular meshing of parametric surfaces via close packing of ellipsoidal bubbles. In 6th International Meshing Roundtable, pp. 375–390. 1996.

[115] Peter Shirley. Discrepancy as a quality measure for sample distributions. In: *Eurographics*, pp. 183–94. Elsevier Science, Amsterdam, 1991.

[116] I. M. Sobol. The distribution of points in a cube and the approximate evaluation of integrals. *USSR Comp. Math. Phys.*, 7(4):86–112, 1967.

[117] Mauro Steigleder and Michael McCool. Generalized stratified sampling using the Hilbert curve. *Journal of Graphics Tools*, 8(3):41–47, 2003.

[118] Raymond C.W. Sung, Jonathan R. Corney, David P. Towers, Ian Black, Duncan P. Hand, Finlay McPherson, Doug E.R. Clark, and Markus S. Gross. Direct writing of digital images onto 3d surfaces. *Industrial Robot: An International Journal*, 33(10):27–36, 2006.

[119] Vitaly Surazhsky and Craig Gotsman. Explicit surface remeshing. In: *Symp. on Geometry Processing*, pp. 20–30. 2003.

[120] R. A. Ulichney. Dithering with blue noise. *Proc. of IEEE*, 76(1):56–79, 1988.

[121] Luiz Velho and Jonas de Miranda Gomes. Digital halftoning with space-filling curves. *ACM SIGGRAPH Computer Graphics*, 25(4):81–90, 1991.

[122] Alex Vlachos, Jörg Peters, Chas Boyd, and Jason L. Mitchell. Curved PN triangles. In: *Symp. Interactive 3D Graphics*, pp. 159–166. 2001.

[123] Jens Vorsatz, Christian Rössl, Leif Kobbelt, and Hans-Peter Seidel. Feature sensitive remeshing. *Computer Graphics Forum*, 20(3):393–401, 2001.

[124] T Warnock. Computational investigations of low-discrepancy point sets. S. K. Zaremba, editor, *Applications of Number Theory to Numerical Analysis*, pp. 319–344. Academic Press, 1971.

[125] Tien-Tsin Wong, Wai-Shing Luk, and Pheng-Ann Heng. Sampling with hammersley and halton points. *Journal of Graphics Tools*, 2(2):9–24, 1997.

[126] Henryk Woźniakowski and Ian Sloan. When are quasi-monte carlo algorithms efficient for high dimensional integrals? *Journal of Complexity*, 14:1–33, 1998.

[127] J. Wu and L. Kobbelt. A stream algorithm for the decimation of massive meshes. In: Graphics Interface, pp. 185–192. 2003.

[128] Jianhua Wu and Leif Kobbelt. Optimized sub-sampling of point sets for surface splatting. *Computer Graphics Forum*, 23(3):643–652, 2004.

[129] Jianhua Wu, Zhuo Zhang, and Leif Kobbelt. Progressive splatting. In: Marc Alexa, Szymon Rusinkiewicz, Mark Pauly, and Matthias Zwicker, editors, *Symp. Point Based Graphics*, pp. 25–32. Eurographics Association, 2005.

[130] JI Yellott Jr Spectral consequences of photoreceptor sampling in the rhesus retina. *Science*, 221(4608):382–385, 1983.

[131] Shin Yoshizawa, Alexander G. Belyaev, and Hans-Peter Seidel. A fast and simple stretch-minimizing mesh parameterization. In: *Shape Modeling International*, pp. 200–208. IEEE Computer Society, 2004.

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

[133] Matthias Zwicker. *Continuous Reconstruction, Rendering, and Editing of Point-Sampled Surfaces*. PhD thesis, ETH Zurich, 2003.

[134] Matthias Zwicker, Mark Pauly, Oliver Knoll, and Markus Gross. Pointshop 3D: An interactive system for point-based surface editing. In: John Hughes, editor, *SIGGRAPH*, pp. 322–329. ACM Press, New York, USA, 2002.

[135] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. Surface splatting. In: *SIGGRAPH*, pp. 371–378. ACM Press, New York, USA, 2001.