Generalized Anisotropic Stratified Surface Sampling

J. A. Quinn, F. C. Langbein, Y.-K. Lai and R. R. Martin

Abstract—We introduce a novel stratified sampling technique for mesh surfaces that gives the user control over sampling density and anisotropy via a tensor field. Our approach is based on sampling space-filling curves mapped onto mesh segments via parametrizations aligned with the tensor field. After a short pre-processing step, samples can be generated in real-time. Along with visual examples, we provide rigorous spectral analysis and differential domain analysis of our sampling. The sample distributions are of high quality: they fulfil the blue noise criterion, so have minimal artifacts due to regularity of sampling patterns, and they accurately represent isotropic and anisotropic densities on the plane and on mesh surfaces. They also have low discrepancy, ensuring that the surface is evenly covered.

Index Terms—sampling, stratified, anisotropy, low-discrepancy, blue noise, spectrum analysis, non-photorealistic rendering.

1 Introduction

Sampling is used widely in computer graphics and geometry processing, for purposes such as rendering, remeshing and mass property evaluation. Many sampling approaches are available and the requirements vary considerably between applications. Generally, a manifold sampling should represent the manifold to a given accuracy with as few points as possible, but must also meet application specific requirements. For most visual computing applications, sampling should avoid artifacts: a regular sampling structure can interact with the sampled shape in an undesirable way. One measure to quantify this effect is the *blue noise criterion*: the radially-averaged power spectrum density of a sampling should have small low-frequency components and no high-energy spikes [1]. A blue noise distribution aims to avoid degradation caused by too unstructured a discrete sample distribution, and yet avoids too much global structure. Another effective measure of sampling quality is *discrepancy* [2]: unlike random samples, sample distributions with low-discrepancy optimally bound the maximal size of holes between samples. While a regular grid is undesirable, so is being too far from one.

Non-photorealistic rendering (NPR) research [3], [4] notes that patterns and regularities are quickly detected by humans, and unless the application specifically benefits from it (e.g. by providing improved clarity in medical imaging), they should be avoided. Slightly random placement of points typically has greater aesthetic appeal: in a review of NPR, Hertzmann [5] noted that sampling for NPR requires even surface coverage with some randomness to introduce an artistic non-determinism. Whilst the highly popular *Poisson disk* methods avoid a very regular grid structure [6], such methods approximate a *centroidal Voronoi tesselation* (CVT) [7], which converges to a hexagonal grid. Such algorithms are normally terminated before convergence, and dart throwing algorithms limit this structure by using random sample initialisation. However, remnants of the hexagonal structure are still present in the output, resulting in large mid-frequency peaks when considering the radially averaged power spectrum, or concentric rings in the mean periodogram [8]. Furthermore, the discrepancy from such methods is known to be worse than for other semi-structured methods, such as *stratified sampling* [9]. To overcome the limitations of Poisson disk methods such as [10], we propose a manifold sampling method that has less visible local structure and a more uniform global coverage of the domain. These properties are better suited for sampling in both NPR and Monte Carlo applications. Figure 1 compares (approximately) uniform stippling produced by (a)–(c) artists, (d) our method, and (e) Poisson disk sampling. The outputs of our method and Poisson disk sampling are quite different. The majority of stipplings produced by active artists, and the founding neo-impressionists, use patterns much more like those produced by our method (of course other artists may prefer patterns like those produced by Poisson disk sampling). Hence, the output from our method has artistic utility.

Control of both density and anisotropy are important when sampling. Local control of density enables efficient use of a given budget of samples: e.g. more points may be placed in areas with greater detail or higher curvature, and fewer in flatter regions. Figure 2 demonstrates that the regular structure of the Poisson disk method is still apparent in density-controlled sampling. Anisotropic, directional control of the spacing between samples can further benefit this sampling budget. It allows samples to be placed closely in one direction, and at a greater distance in the orthogonal direction, which may be useful, e.g., when the principal curvatures of a shape differ widely. Anisotropic sampling also allows for the use of anisotropic sampling primitives; e.g. long brush strokes as used in NPR may be efficiently placed so that overlap is minimized. Figure 3 shows two brush-stroked samplings generated using our method. An arbitrary, uniform tensor field was defined with an $8{:}1$ aspect ratio; uniform isotropic sampling and anisotropic sampling with respect to the tensor field were generated, each with 500 points, and both point sets rendered with a randomized set of brush strokes. Brush strokes vary in size, but maintain the same aspect ratio of $8{:}1$. Considerable stroke overlap and many large holes are visible in the rendering of the isotropic sampling. The anisotropic sampling demonstrates a better coverage with the same number of points due to a better stroke arrangement, whilst still not appearing overly structured. Li et al [10] also note that, to achieve an accurate isotropic sampling that is not subtly warped due to parameterization of the manifold, anisotropic control of samples is required.

Our novel approach is based on the space-filling curve sampling method of Quinn et al [11], generalized to produce anisotropic, density controlled, low-discrepancy blue noise samples on meshes. The basis of our method is to segment the mesh according to a user defined two-tensor field, and sample each segment according

• *All are with the School of Computer Science & Informatics, Cardiff University, UK.*
E-mail: {j.a.quinn, f.c.langbein, yukun.lai, ralph}@cs.cf.ac.uk

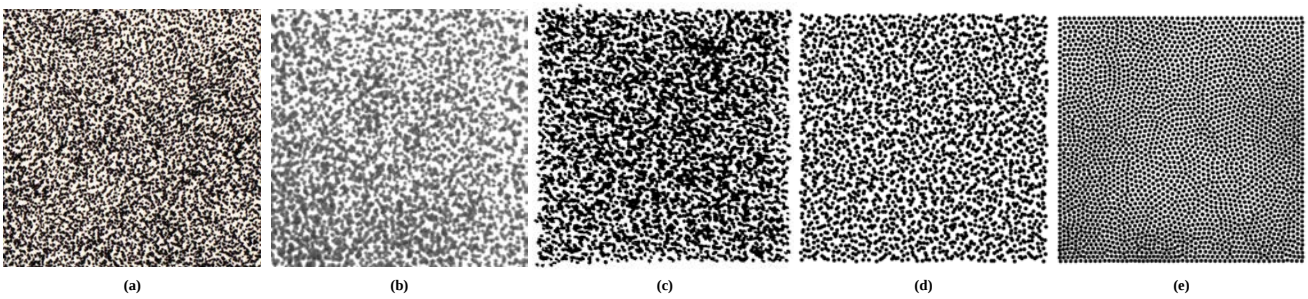

Fig. 1: *Examples of (approximately) uniform stippling produced by: (a)–(c) artists (Q. Rumbley, J. Visscher, C. Fulkerson, respectively), (d) our method, and (e) Poisson disk sampling, using about 2500 points.*

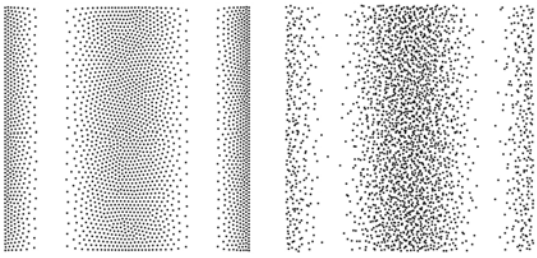

Fig. 2: $1,500$ *samples with density* $\delta = cos(x) + 1$ *for: (left) Poisson disk sampling, (right) our method.*

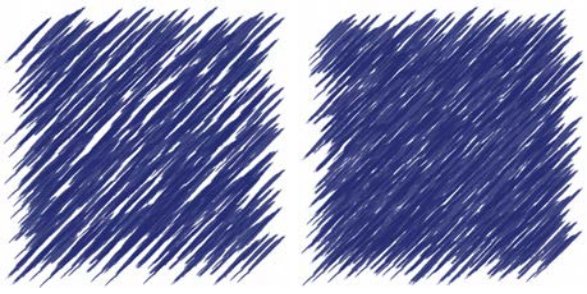

Fig. 3: 500 *acrylic brush-stroked samples: (left) isotropic placement, (right) anisotropic placement.*

to the local anisotropy. The solution to this problem requires significant additional steps compared to previous work:

- an extension to a random-walk-based method to segment a mesh according to variation of the prescribed tensor field;
- a robust method for detecting singularities on a mesh, and using them to control the segmentation;
- an extension of a well-known parameterization method to consider local stretch according to the tensor field;
- generation of space-filling curves accurately aligned to a user-defined tensor field on a mesh, which are locally adapted to handle boundaries between curves;
- an extended, anisotropic, definition of *star discrepancy*.

We refer to the output of our method as *generalized anisotropic stratified surface sampling*, following the terminology of *generalized stratified sampling*, introduced by Steigleder and McCool [12]. We believe this to be the first work addressing the generation of anisotropic stratified samples on meshes. With little preprocessing, millions of points can be sampled on large meshes in real-time, with density and anisotropy prescribed by a tensor field. The resulting sampling has a lower discrepancy than Poisson disk sampling (independent of the test shape used to measure

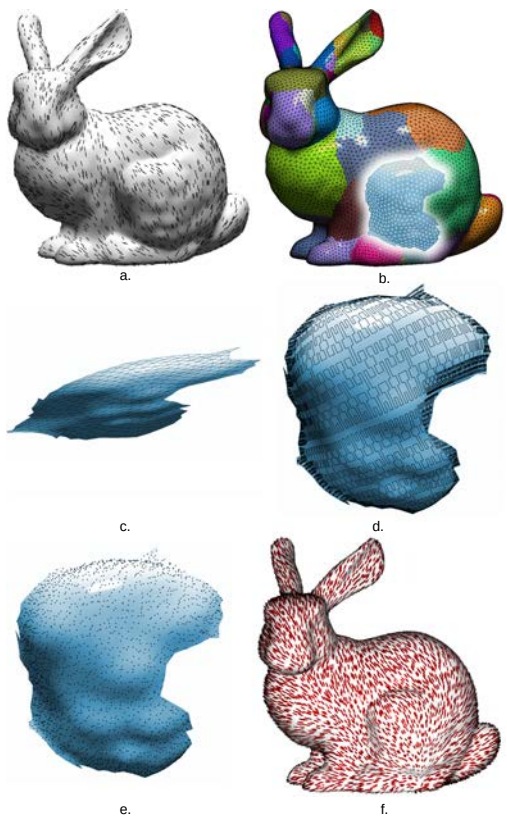

Fig. 4: *Steps in our anisotropic sampling algorithm: (a) input tensor control field, (b) mesh segmentation, (c) parameterization of one segment, (d) Hilbert curve on that segment (shown at reduced depth for visualization), (e) point samples for that segment, (f) point samples as splats.*

the discrepancy), and fulfils the blue noise criterion, using both conventional spectral analysis, and methods described in [8]. We provide comprehensive experimental evidence suggesting that although our process is approximate, problems do not arise due to imprecise alignment of curves, segmentation, or quantization.

2 Related Work

For many point-based rendering techniques, samples should be roughly equidistant (taking into account local sampling density), with spacing depending on the direction across the manifold, using a locally defined metric. Such equidistance from neighbors results in a uniform degree of overlap of sampling primitives. If points

are not regularly spaced, a higher density is required to ensure complete coverage. However, as distances become more regular, the distribution becomes more and more grid-like, causing sampling artifacts. Low-discrepancy distributions have been employed to avoid such artifacts, but the sampling pattern also matters: Dobkin et al [9] demonstrate that in distributed ray-tracing, sampling patterns with very low discrepancy defined in terms of *rectangular* holes (see Niederreiter [2]) may still produce artifacts. Stratified sampling, however, does not exhibit this problem. Whilst demonstrating a higher discrepancy when measured with axis-aligned rectangles, it has a more consistent and lower discrepancy when measured using alternative shapes [13].

Poisson disk sampling is widespread in computer graphics. Many methods can be used to produce Poisson disk distributions [6], [10], [14], [15], often employing a relaxation approach to distribute sample points. The results converge to a CVT, which fulfils the blue noise criterion [6], but are expensive to compute, and result in a rather too regular structure. Cline et al [16] give another method for Poisson disk sampling, converting a surface into fragments and sampling points over each fragment using a dart throwing approach. Inexact intersection tests for ellipsoids limit sample placement precision on triangle meshes. Unfortunately, computation times for ellipsoid placement, with or without a geodesic surface measurement, are not stated, nor is any analysis given of the quality of the ellipsoid distribution produced. Balzer et al [17] describe a variation of Lloyd's method that stops before a CVT is reached, such that each sample has the same area with respect to a density function. While this method demonstrates a superior termination criterion for Lloyd's method in the plane, contrary to its description, it still produces regular artifacts, especially when small regions are visualized. Bowers et al [18] describe a parallelized method for generating Poisson disk samples on surfaces using dart throwing. However, Lagae et al [6] note that dart throwing methods may require further expensive relaxation steps for large radii.

Other sampling methods, such as [19], use error metrics to produce simplified point distributions from high-resolution scanned point sets, avoiding the need to triangulate the surface. Errors introduced by simplification processes can easily be avoided. However, this sampling method is linked to the geometry of the surface, and using an arbitrary density function with this method is difficult. It is also unclear how to extend this method to anisotropic sampling. [20] describes a method to perform stratified surface sampling which relies on voxelization of the model, and places samples on the surface within each voxel. Sampling with a grid that is not regularly spaced with respect to the surface geometry may result in an unevenly sampled model. The method also does not allow for anisotropic control of the sample distribution.

Considerable work exists on point set rendering using artistic methods such as dithering, stippling and strokes. Methods such as [21] use a texture-based approach for rendering artistic patterns like hatching, employing *mip-mapping* to handle changes in viewing distance. While this approach is fast, it lacks the ability to control the density of the rendered elements. Other methods, such as [22], [23], [24], [25], use high quality point sampling methods and rendering 'brushes' to achieve artistic styles, but these methods cannot operate at interactive rates. Work by Umenhoffer et al [26] considers real-time hatching using low-discrepancy samples. However, as the method uses Halton points, the sampling is highly-structured and does not fulfil the blue noise criterion [27]; furthermore, their method does not support anisotropic point spacing.

Work such as [6], [28], [24] has started to use easily repeatable methods for assessing the quality of sample distributions. Frequency analysis and Voronoi cell traversals are useful to assess the anisotropy of uniform functions in the plane. More recently, the assessment of these properties for arbitrary density functions, anisotropy, and on surfaces has become a focus. Li et al [10] suggest warping a sampled function to a uniform, isotropic representation to assess quality. Bowers et al [18] describe a spectral analysis method to assess sampling quality on meshes using spectral mesh basis functions; it is limited to assessing a relatively small number of samples. More recently, Wei and Wang [8] describe a method that builds upon the assessment techniques of [10] and [18] to analyze the quality of samples in non-uniform, anisotropic and non-Euclidean domains. Their approach reformulates standard Fourier analysis methods to rely on differentials of positions rather than positions directly. This method provides a significant tool for analysis of complex samplings, and is straightforward to apply to most sampling methods.

Unlike Poisson disk sampling, our approach leads to a stratified sampling. This makes it easier to ensure sufficient randomness in the distribution, which is important in applications such as artistic rendering [5]. Further advantages are that our approach is much faster, as intersection tests and relaxation are not required, while the discrepancy of stratified samples is also lower. After a short precomputation, our method can generate high-quality, low-discrepancy, anisotropic samples in real-time.

3 Algorithm Overview

Our approach produces an anisotropic sampling of a manifold in which the spacing between neighboring samples varies depending on the direction. The results of each main step are illustrated in Fig. 4. The input is a triangle mesh with a smooth two dimensional tensor field defined on it: a positive-definite, symmetric, rank 2 tensor is given at each mesh vertex. This controls the distances between samples in two orthogonal directions, determining where samples should be placed. The tensor's eigenvalues give sample density in the direction of the corresponding eigenvector; the overall local sampling density is given by the square-root of the product of the eigenvalues. This tensor can thus be understood as an alternative metric on the surface.

In practice, supplying a tensor field is not easy for users. Approaches like those in [29], [30], [31], [32], [33] may be used to generate a smooth tensor field from simpler inputs. Often, anisotropic sampling is used to produce samples according to curvature. In this case, the principal curvatures and directions of the mesh themselves provide the tensor, and may be computed using, e.g., the method in [34] (principal curvatures computed from a mesh may be quite noisy, so it may be desirable to smooth them before use). Singularity control is widely available in the field design literature, and whilst careful positioning can lead to a reduced complexity in the application of our method, we leave this, along with definition of the field, to the user. Whilst smoothing the field before use can reduce the number of singularities in the mesh [35], it may have little effect.

The degree of quantization of the underlying continuous tensor field depends entirely on the quality of the input mesh. Much of the behavior of a complex tensor field may be lost if the mesh is too coarse. Whilst no specific quality of mesh or field is required

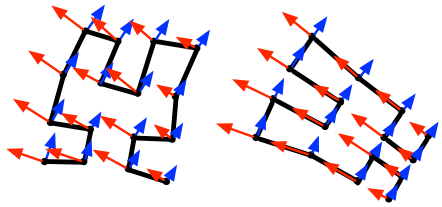

Fig. 5: *Surface Hilbert curve, left: aligned to (arbitrary) parameterization of geometry, right: aligned to tensor field.*

for our method to function, the user should consider the accuracy of both the tensor field and input triangle mesh in order to achieve the desired results from the anisotropic sampling.

Previous space-filling curve based surface sampling algorithms [13] have generated a space-filling curve in the parameter domain and mapped it onto the surface. The standard basis of the parameter domain is mapped onto an arbitrary orientation on the surface, as determined by the parameterization. In general, this is not aligned with the eigenvectors of the tensor, so the anisotropy cannot be readily controlled. Quinn et al [27] also show that the structure of the space-filling curve has an effect on the placement of points, and therefore on the structure of the sampling. Here, in order to produce a set of point samples with a desired anisotropy, we carefully generate a curve in the parameter domain that, when mapped onto the manifold, is aligned and stretched locally according to the tensor field (see Fig. 5).

An anisotropic space-filling curve could be generated that fills a space defined by an anisotropic metric in the parameter domain, ensuring that it is aligned to the eigenvectors of the tensor field after anisotropic stretching maps it to the surface. This would require a complicated curve construction to compensate for the distortion of the parametrization and to align it to the tensor field. Instead, we adopt an alternative approach. We parameterize the surface according to a tensor field, such that we can treat the parameter domain as having an isotropic metric and therefore can generate a straightforward space-filling curve. The anisotropy of the tensor field is then induced onto the space-filling curve by the parameterization. However, it is not feasible to do this for the whole manifold in one piece. Thus, we first segment the mesh into pieces each having locally similar anisotropy (see Section 4), ensuring that each segment can be parameterized appropriately. Segmentation also simplifies topological issues, as most segments naturally have a disc topology. Whilst this is not guaranteed, very few segments must be cut to give discs at a later stage.

Singularities in the tensor field must be handled with care (see Section 4.1); in general they cannot be avoided (except for genus 1 closed surfaces). At a singularity the eigenvalues coincide and the eigenvectors are no longer uniquely defined. Thus, if a singularity lies within a segment, we cannot cover it with a parameterization which is well aligned with the tensor field everywhere in that segment. We avoid this problem by forcing singularities to lie on segment *boundaries*. We first detect all singularities, then connect them using a minimal set of mesh cuts that follow the field lines of the tensor field. Segments containing singularities are then further divided using these cuts as boundaries. Triangles containing singularities are temporarily removed from the mesh, to be dealt with individually later.

Segments are further cut, if needed, to have the topology of a disc; cuts follow tensor field lines on the mesh (see Section 4.2) so that segments can be easily parameterized in the plane. Each segment is then parameterized such that the eigenvectors of the tensors in the segment lie along the parametric directions (see Section 5). During this process, each mesh segment is also stretched locally according to the eigenvalues of the tensors. The field lines in the parameterized segment are therefore quite close to parallel. If we cannot align the tensor eigenvectors sufficiently accurately, or a parameterized segment intersects itself, the segment is subdivided into smaller segments, and the process repeated, until a *valid* parameterization is produced. This process converges quickly, generally resulting in large segments, as long as the field is relatively smooth with respect to the resolution of the mesh.

We next generate a space-filling curve in the parameter domain (see Section 6.1), adaptively proceeding to different depths of recursion to take into account local variation in segment sizes on the mesh. The space-filling curve is then mapped onto the mesh in $\mathbb{R}^3$ using a barycentric mapping and fast point-within-triangle lookups. To provide a mapping between points in the parameter domain and the mesh triangles, raster images of the parameterized segments are created, in which each triangle is colored uniquely, to enable fast lookups.

During mapping, densities are computed as the square-root of the determinant of the tensor. (If the tensor describes the curvature, this is the usual surface area element). These densities are then integrated along the curve, reducing the problem to 1D sampling of the curve. The algorithm steps along each vertex of the curve and whenever the accumulated density reaches a certain value, derived from the tensor field, a point is sampled. The output of this process is an anisotropic, low-discrepancy point sampling on the input mesh. (If an *isotropic* tensor is input, the eigenvectors are not uniquely defined anywhere, and this process fails. However, previous methods can be used to handle this simpler case [11], for a whole mesh, or for any segment with isotropic tensors).

4 Mesh Segmentation

We now describe the tensor field controlled mesh segmentation. We aim to produce a set of disc topology segments, each of which can be parameterized over a subset of $[0,1]^2$. We wish to parameterize each segment such that it aligns to the tensor field; the direction of the tensors mapped to the parameter plane should not vary too greatly over the segment (otherwise, the parameterization will fail: producing a parameterization poorly aligned to the tensor field, or a parameterization containing an *invalid* self-intersecting parameter mesh). Thus, we first segment the input mesh into topologically simple pieces, each of which has tensors with similar eigenvectors. Note that a segment smaller than the accuracy achievable by the prescribed number of sample points need not be subdivided further.

Work by Delmarcelle and Hesselink [36] has a similar goal, for use in the plane. They first find singularities, and segment the surrounding area into hyperbolic and parabolic tensor regions. Whilst this method is simple, changes of direction of the field in hyperbolic regions requires further segmentation. It is also unclear how well it would generalize to manifold segmentation. An alternative approach would be to treat single triangles as segments, and construct charts of such segments that do not contain singularities [37]. However, this will often result in too few samples per segment, increasing sampling complexity.

Instead, to segment the input triangle mesh, we modify Lai et al's [38] random walk method. This fast method works well for

a wide range of objects with limited need for parameter tuning. Our modification lies in the distance measure used, and results in a mesh segmented according to the anisotropy of the tensor field. Initially, a set of seed triangles is automatically chosen, starting with a random first seed triangle. The algorithm then steps out in rings around this triangle. At each step, the average of the first eigenvectors (with largest eigenvalue) of the tensors at the triangle vertices is computed. If the angle between this vector and the vector in the previous triangle is larger than a parameter β, a seed is placed in that triangle. When a new seed is placed, the process is repeated, and the algorithm then steps out in rings around this new seed; expansion may not cross existing paths. We set β to $\pi/4$, although this may be adjusted by the user. Decreasing the value of β results in a more dense set of seeds.

For each non-seed triangle f_i, probabilities $p_{i,1}$, $p_{i,2}$ and $p_{i,3}$ are associated with each of its three edges $e_{i,1}$, $e_{i,2}$ and $e_{i,3}$, summing to 1 for the triangle. This probability corresponds to the likelihood that a random walk will cross the edge into a neighboring triangle. A function $d(f_i, f_k)$ must be defined, measuring the *difference* between two neighboring triangles f_i and f_k. Lai et al [38] use the dihedral angle to define this measure. However, here, we describe the difference in terms of the tensor field rather than mesh geometry. For each triangle f_i we compute the barycentric linear interpolant t_i', which corresponds to the tensor at the barycenter of f_i. The difference between two triangles f_i, f_k is then given by the angle between the eigenvectors with the largest eigenvalue $\mathbf{m}_i$, $\mathbf{m}_k$ for this pair of interpolated tensors:

$$d(f_i, f_k) = \cos^{-1}\left(\frac{\mathbf{m}_i \cdot \mathbf{m}_k}{\|\mathbf{m}_i\|\|\mathbf{m}_k\|}\right).$$

When comparing eigenvectors, we use an approximate notion of parallelism from the surrounding Euclidean space, to reduce computational complexity. If error is introduced, it will simply result in further segmentation during parameterization. To handle variations in the tensor field, we normalize d by its average $\bar{d}$ over all mesh edges,

$$d^*(f_i, f_k) = d(f_i, f_k)/\bar{d}.$$

This measures how fast the tensor field is turning: the greater the turn, the less likely both triangles should be included in the same segment. For each non-seed triangle f_i, the un-normalized probability of a random walk, originating at f_i, reaching each seed triangle s_l, is then computed, based on the probability of going from a triangle f_i to one of its neighbors f_k:

$$p_{i,k} = |e_{i,k}| \exp\left(-d^*(f_i, f_k)/\sigma\right),$$

where $|e_{i,k}|$ is the length of the edge shared by f_i and f_k and σ a parameter used to control how the difference between two triangles maps to variations in probability. Setting $\sigma = 1$, as in [38], also works well for tensor differences. Each non-seed triangle is assigned to the seed for which it has the highest probability of being reached first; this can be formulated efficiently as a sparse linear system. The result splits the original mesh into a set of contiguous segments. Fig. 4 shows segmentation results for the Stanford Bunny, using a user-defined tensor field.

4.1 Singularities

The topology of a tensor field is partially defined by its *singularities*. On a discrete mesh, we may locally treat the field at singularities as isotropic. As we linearly interpolate the tensor field inside each triangle from vertex values, a triangle may contain at most one singularity. If a singularity exists in a mesh segment, the parameterization method will fail, as alignment of the tensors is not possible. Therefore, we find the singularities in the tensor field, and further segment the mesh to ensure that singularities lie on segment boundaries rather than in segment interiors.

4.1.1 Singularity Detection

To locate singularities, we follow [39]. Rather than performing the computation in the parameter domain, we instead compute it directly on the mesh for each triangle f_i individually. For each tensor $t_{i,k}$ at each vertex v_k of f_i, we take the eigenvector with the largest eigenvalue, $m_{i,k}$ and project it into the plane of the triangle: $m_{i,k}' = m_{i,k} - (\mathbf{N}_i \cdot (m_{i,k}\mathbf{N}_i))$. The eigenvector is sufficient to represent the tensor so long as we have the normal $\mathbf{N}_i$ of the triangle f_i, and can therefore orient the surface. Following [36], [35], these projected vectors are then represented in barycentric coordinates. We set the linear combination of the eigenvectors to zero in barycentric coordinates (a, b, c) [39]. If a solution to this linear system exists with $a, b, c \geq 0$, a singularity lies within triangle f_i. An acceptable discontinuity error is introduced by projection of the eigenvectors into the tangent plane of the triangle. A possible resulting false singularity would simply cause slight over-segmentation, whereas a missed singularity would cause a problem in the parameterization, simply resulting in further segmentation later in the algorithm. Auer and Hotz [40] investigate virtual singularities over smooth transitions between discrete tensors. However, in this work, we simply require the singularities to find an appropriate segmentation, and thus it is actually beneficial to ignore any that do not pose a problem.

4.1.2 Further Segmentation

Connecting the singularities to each other and to existing segmentation boundaries appropriately allows us to produce a new segmentation in which all singularities lie on a segment boundary. We connect singularities on a per-segment basis, rather than considering all pairs of singularities, which greatly simplifies the problem while still providing results that meet our needs. When connecting singularities from a mesh segment, we use paths lying along the tensor field lines, which helps to ensure that the new segments have minimal directional variation.

To be able to find paths that are aligned as closely as possible with the field lines, for an edge $e_{i,j}$ connecting vertices v_i and v_j, we compute a distance $D_{i,j}$ with respect to the field: $D_{i,j}$ is the angle between the eigenvectors with the largest eigenvalues of the tensors t_i and t_j at vertices v_i, v_j, $\mathbf{m}_i$, $\mathbf{m}_j$, scaled by the edge length $|e_{i,j}|$ (in the Euclidean metric). We then compute the shortest path between each pair of singularities within a segment using Dijkstra's algorithm, with these distances. It is possible that two singularities within a segment are not connected by tensor lines, but without their evaluation, this is non-trivial to compute. If only one singularity lies within a segment, we instead compute the shortest path from the singularity to the boundary of the segment, and return this path. For two or more singularities, we construct a graph from the set of all pairwise shortest paths, in which the shortest paths between singularities are the edges of the graph. We then compute and return the minimum spanning tree for this graph (Fig. 6). The use of $D_{i,j}$ causes the resulting paths to follow the field lines between singularities as closely as possible, and the tensors to vary without undue rotation along the path.

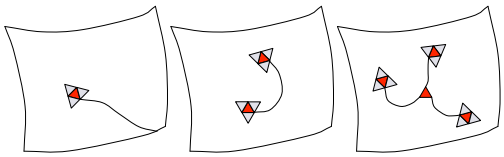

Fig. 6: *Cuts between tensor field singularities (shown in red) in a segment. Left: one singularity, middle: two singularities, right: three or more singularities. Grey triangles indicate new seed points for further segmentation.*

The path connecting the singularities within the mesh segment is then traversed, and only singularities lying at leaf nodes in the path are added to a set G of new seeds for this segment (see Fig. 6). We subdivide the mesh segment by performing a new segmentation. During this segmentation only triangles adjacent to singularities in G are used as seeds (shown in grey in the figure), with the restriction that random walks may not cross any edge in the path (also see Fig. 4). This strategy ensures that the region surrounding a singularity triangle is always segmented further, as the tensor field changes quickly around such triangles.

Due to the discrete nature of the tensor field, triangles containing a singularity can be treated as if they have an isotropic density, and thus do not have to be parameterized in alignment with the tensor field. Instead, they are flagged in the mesh. As explained in Section 6, triangles containing singularities are parameterized uniformly in the plane and are sampled isotropically according to the local density when generating output point samples.

4.2 Topological Consistency

In order to ensure that each segment can be correctly parameterized in the unit square, we compute its topology via its Euler characteristic, and cut it to give it a disc topology if it does not already have one. To preserve alignment, the cut should follow the tensor field lines. Thus, we again use edge lengths according to the metric as above: setting edge distances along boundaries to 0, we compute the shortest path between each pair of segment boundaries using Dijkstra's algorithm. We again compute the minimum spanning tree of the graph constructed from these paths, and cut the segment along this minimum spanning tree; this may result in multiple segments.

5 Mesh Segment Parameterization

For each segment, we compute a parameterization aligned with and locally stretched according to the tensor field. This allows us to sample the parameter domain of each segment with a standard space-filling curve, which, when mapped to the surface S, inherits the anisotropy of the field. We perform parameterization and segmentation using an iterative process, carrying out further segmentation as needed to ensure the *validity* of each parameterized segment, whilst avoiding over-segmentation. A *valid* mesh segment (i) does not intersect itself when parameterized, and (ii) has parameterized angular variance of the field below a user-defined threshold. Keeping the total number of segments low limits the number of boundaries between segments, in turn reducing the computation required to ensure consistent sampling.

5.1 Parameterization

We perform discrete parameterization of each segment, adapting Ray et al's [37] method, similar to the solution described by Auer et al. [41], to enforce a parameter spacing based upon the tensor field. The result is a low-distortion, field-aligned mesh parameterization, obtained by formulating the problem in terms of minimizing an energy function. This can be quickly solved via a sparse linear system. As we perform the parameterization, we introduce as scaling factors the square-root of the eigenvalues, $\sqrt{\lambda_{i,1}}$ and $\sqrt{\lambda_{i,2}}$, of the tensor t'_i averaged over the three vertices of each triangle f_i. Thus, with the parameters ∇u and ∇v (see [42]) and the orthonormal eigenvectors $\mathbf{m}_{i,1}$ and $\mathbf{m}_{i,2}$ of t'_i, the discrete energy functional to be minimized is

$$F^* = \sum_{i \in T} \left(\left\| \nabla u - \sqrt{\lambda_{i,1}} \mathbf{m}_{i,1} \right\|^2 + \left\| \nabla v - \sqrt{\lambda_{i,2}} \mathbf{m}_{i,2} \right\|^2 \right) A_i,$$

where $i \in T$ indexes triangles of the segment, and A_i denotes the area of triangle i.

5.2 Segment Validity

The output of the above step is a segment parameterized in the 2D plane, approximately aligned and locally stretched according to the tensor field. However, it may not be *valid*. We first test if it contains self-intersections using fast triangle overlap detection [43]. If none are found, we check the alignment of the field. We first map the tensor field into the parameter domain. The eigenvector of the tensor with the largest eigenvalue, $\mathbf{m}_{i,1}$, is projected into the tangent space of the underlying triangle (or closest triangle, if on the boundary of a segment). It is parameterized with respect to the edge vectors of the triangle. At each vertex, we compute the angle between the projected $\mathbf{m}_{i,1}$ and the basis vector in the parameter domain with which it should be aligned. In order to measure how well the field in the segment is aligned, we compute the average of the angle distribution. If the average is less than a threshold γ, we declare the segment to be *valid*. If the angular deviation is too high, the surface sampling will be poorly aligned with the tensor field. In practice, a value of $\gamma = \pi/16$ is used, ensuring adequate accuracy, whilst avoiding too many segments.

If a segment is *invalid*, two new seeds are generated in order to split the segment into two using random-walk segmentation (see Section 4). The seed positions are determined heuristically using a simple method. One tensor is chosen arbitrarily, and the angle between its eigenvector $\mathbf{m}_{i,1}$ and the eigenvector $\mathbf{m}_{j,1}$ of every other tensor in the segment is computed. The tensor t_a with the largest angular difference is kept. The angle between the eigenvector of t_a and every other tensor is then computed pairwise, and the tensor t_b with the largest angular difference is kept. The locations of t_a and t_b are used as seeds.

6 Mesh Sampling

Having obtained suitable parameterized mesh segments, we generate a space-filling curve for each segment S with its parameterized equivalent S_p. S_p is then uniformly scaled to fit within the unit square. A rasterization of S_p is generated to accelerate containment checks for space-filling curve generation; any curve outside S_p is discarded. The rasterized mesh is also used for fast point-within-triangle lookups when mapping the curve within S_p

to S. The pixel size of the rasterization is set smaller than the minimum triangle size in S_p, to avoid discretization errors.

Space-filling curves provide a continuous mapping from the unit interval onto a higher dimensional domain, in this case the unit square. We use a space-filling curve to reduce a complex sampling problem on a 2D surface in 3D to the simpler problem of sampling a 1D interval. The Hilbert curve is simple to construct and provides good spatial coherence; it produces better samples than other space-filling curves [27]. A first order approximation of the Hilbert curve can be constructed by dividing $[0,1]^2$ into four congruent squares, with the curve following a self-avoiding path through the center of each square. 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 to maintain a contiguous curve, resulting in 2^{2r} subsets, where r is the approximation order.

6.1 Space-Filling Curve Generation

The Hilbert curve should densely fill the area of the rasterization that contains the mesh segment, but have lower density in the remaining area of the rasterization, which is later discarded. To avoid gaps between points near segment boundaries, the curve should also be denser near the boundary of the mesh segment. We thus create an adaptive Hilbert curve, using a quad tree algorithm following Quinn et al. [13], [11]. When generating the curve, we determine if each quad's area contains part of the mesh, and if this part of the mesh is near the boundary. To do so efficiently, and to prevent gaps appearing near the boundary, we first generate a Hilbert curve to an initial approximation order $r \in \mathbb{Z}$. As the actual size of a segment S may vary considerably, r is computed per-segment using a step function of the area of S. Using a fast OpenGL fragment shader, the rasterized mesh in $[0,1]^2$ is dilated by $\lceil 1/2^r M \rceil$ pixels, where M represents the number of pixels along one axis of the square rasterization. This dilation is large enough to ensure that at least one vertex from any quad in the initial Hilbert curve that intersects the boundary of the mesh must lie within the dilated area.

We adaptively generate a discrete Hilbert curve in $[0,1]^2$, resulting in a set of vertices h_l. Locally, for a density $\delta = \sqrt{\lambda_{i,1}\lambda_{i,2}}$ where $\lambda_{i,1}$ and $\lambda_{i,2}$ are the eigenvalues of the tensor t_i, the number of curve vertices required is a constant ω times the ratio $\int_U dA / \int_S dA$, where dA is the area element according to the tensor field ($dA = \delta\, du \wedge dv$), for every subset U of S. 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 S_p and S, we cannot simply increase the depth of the Hilbert curve in proportion to the area of each triangle in S (a small triangle may map to a large triangle). Hence, first we define the number of point samples needed in a given surface triangle f_i in S as $N_{f_i} = \int_{f_i} dA / \int_S dA$. We require ωN_{f_i} Hilbert vertices inside a surface triangle. Given the initially constructed Hilbert curve to approximation order r, we approximate the required number of point samples N_Q that should be contained within a quad Q of the Hilbert curve at a given level: $N_Q = N_{f_i}/(\text{area}(p_i)/\text{area}(Q))$, where p_i is the pre-image of the triangle f_i in the parameter mesh S_p and the area is computed with respect to the tensor field metric. We therefore require at least ωN_Q Hilbert curve vertices within a quad Q. Quads with vertices lying within the dilated boundary are further iteratively refined twice to increase the density near the boundary and avoid gaps between neighboring segments. Hilbert curve vertices that lie outside the rasterized mesh are then discarded (see Fig. 7, left).

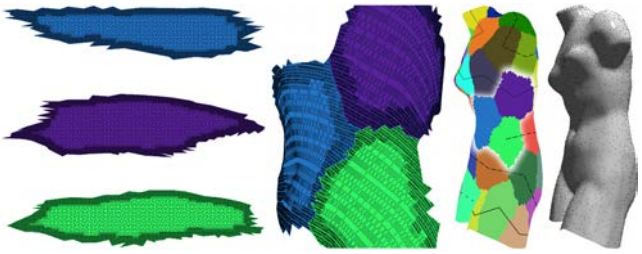

Fig. 7: *Left, planar Hilbert curve segments, middle left, surface Hilbert curve segments, middle right, Hilbert curve chains, shown as different line styles, right, sampling.*

Having generated the Hilbert curve, we must map it to the surface. Let $S_p = \{p_i : i = 1, \ldots, n\}$ be the parameterized mesh for surface mesh segment $S = \{f_i : i = 1, \ldots, n\}$, where p_i is a triangle in $\mathbb{R}^2$ and f_i a triangle in $\mathbb{R}^3$. The bijective mapping $g : S_p \leftrightarrow S$ is used to map the curve from $\mathbb{R}^2$ to $\mathbb{R}^3$. Therefore, $g : p_i \leftrightarrow f_i$. Thus, the function to map a Hilbert vertex from S_p to S simply becomes $g(h_l) = H_l$ for $h_l \in p_i$, $H_l \in f_i$. In order to compute this mapping between the parameterized mesh S_p and the surface mesh S, we convert (u, v) coordinates of the vertex h_l in S_p to local barycentric coordinates within a triangle p_i. Now, p_i maps directly to a single surface triangle f_i due to the bijective mapping between the sets of triangles, so we can use the same barycentric coordinates in f_i for H_l. These are then converted into (x, y, z) coordinates on S, giving the final vertex position. The triangle p_i containing a Hilbert vertex h_l is found by performing a lookup in the rasterization of the mesh segment, which returns a uniquely colored triangle from which the triangle can be determined using a hash table. The output from this algorithm is a set of independent Hilbert curves in $\mathbb{R}^3$, densely covering the input mesh (see Fig. 7, middle left).

6.2 Sampling

We next sample points along each independent Hilbert curve. In order to produce a consistent sampling on the surface, we must maintain a smooth change in density along the curve. Therefore, the distance between neighboring Hilbert curve vertices on the surface should be small compared to the rate of density change. Trying to combine the set of Hilbert curves into a single contiguous curve may result in a complex routing problem in order to maintain a smooth density change, and in general may not be solvable. If $t/N < 1/3$, where t is the number of segments, each curve is sampled individually. If not, then segments are linked together into chains, reducing possible quantization error (see Section 7.3). A seed segment is chosen at random (weighted toward the boundary) and, using a greedy algorithm, the smallest-area neighboring segment is appended to the current segment. This occurs until no further neighboring segments exist, resulting in a chain of pairwise-adjacent segments. A new seed segment is then selected and similarly grown, until the process cannot link any more segments. Only adjacent curves may be connected, as otherwise large changes in density may occur, resulting in potentially erroneous sample placement. The output of this algorithm is several chains of pairwise-adjacent segments (and possibly some individual segments). The curves within these segments are connected, forming fewer, longer curves (see Fig. 7, middle right).

We then compute the number of samples that must be placed along each chain of Hilbert curves. If the global number of required samples is N for the input mesh G, the number of points N_S required for a segment S is:

$$N_S = \left\lfloor N \left(\int_S dA / \int_G dA \right) + \epsilon \right\rfloor ;$$

ϵ is the accumulated remainder for each calculation of N_S.

Surface samples are generated by placing points along the Hilbert curve chains using 1D point distributions to produce a low-discrepancy sampling. To do so, we approximate the surface integral $\int_S dA$ with a discrete (at each H_l) cumulative surface sampling density W_l, and sample points according to this density. The discrete density φ_k at a Hilbert vertex H_k is bilinearly interpolated from the densities δ at the vertices, where δ comes from the tensor field (see Section 6.1). A cumulative density function is then computed for each Hilbert curve vertex: $W_l = \sum_{k=1}^{l} \varphi_k$. A set of 1D stratified samples $\{q_k : k = 1, \ldots, N\}$ in $[0, 1]$ is then generated: the elements of an evenly spaced sample set are moved a uniform random amount up to half the distance towards the next or previous sample. We then step along the curve, and whenever the cumulative density W_l becomes larger than the threshold described by the 1D sequence q_k (multiplied by W_L to account for the total density), we sample a point p_k at a vertex of the surface Hilbert curve chain. As the Hilbert curves are locally stretched according to the anisotropy of the tensor field, the output of this process is a density controlled, anisotropic, distribution of stratified points lying on the input mesh (see Fig. 7, right).

7 Experiments

In this section, we first provide examples of meshes sampled using our method, to allow for visual analysis. We further visually demonstrate our sampling method using a non-photorealistic stroke-based rendering technique. We then compute the discrepancy of sample distributions produced using our method in order to assess the global quality of the sampling, and compare these results to those from alternative sampling approaches. We then perform a series of experiments in order to validate the output of our algorithm, assessing the deviation of the curve from the input tensor field, the effect of boundaries on sampling quality, and quantization effects caused by a reduction in the number of samples per segment. We also perform a spectral analysis, telling us about the local neighborhoods of the samples and the local anisotropy. Finally, we perform a differential domain analysis, which gives similar information to the spectral analysis, but for isotropic and anisotropic density functions and on mesh surfaces.

7.1 Visual Examples

To allow visual analysis, we show our sampling results on various mesh surfaces and demonstrate the method applied to stroke-based rendering. However, to help the reader visually interpret the results in this section, we first give examples of anisotropic sampling. Fig. 8 shows three point distributions in $[0, 1]^2$, each with 750 samples. The image on the left shows the output from our method, isotropically sampled with a uniform density. The middle image shows the same uniform density, but with an anisotropic spacing of the points. It was generated by sampling a rectangular domain (with a height of 0.25, and a width of 1) using our method with a isotropic density, then stretching it to $[0, 1]^2$. This result serves as a control. The image on the right shows a square, sampled using our anisotropic method, with a uniform density, and the same 4:1 ratio of anisotropy as the middle image. Both the middle and right images display the same level of visual anisotropy. However, note that whilst the anisotropy present in these figures is indeed measurable (see Figs. 19, 22), to the human visual system the anisotropy is certainly less obvious than it would be in an equivalent anisotropic *grid* or Poisson disk sampling, where the *regularity* of samples greatly enhances the *visibility* of their directionally-dependent spacing.

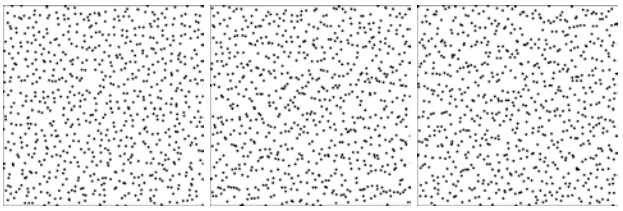

Fig. 8: *Sampling with 750 points. Left: isotropic sampling using our method. Middle: anisotropic curve sampling by manually stretching the parameter domain. Right: anisotropic curve sampling using our method.*

TABLE 1: *Timing.*

Mesh	**2-Torus**	**Wing**	**Bunny**	**Dragon**
No. of faces	5K	10K	18K	206K
No. of vertices	2.5K	5K	7K	103K
No. of samples	5K	4K	8K	10K
Segm. + param.	1.8s	2.6s	3.2s	20.8s
Curve gen.	3.2s	3.6s	4.0s	14.8s
Sampling	0.001s	0.001s	0.001s	0.01s
Total	5s	6.2s	7.2s	35.6s

Fig. 9 shows a 2-torus sampled using our algorithm. The first image shows the user-defined tensor field, containing two singularities and a constant anisotropy of 2:1. The field is rendered as a subset of the original mesh vertices (chosen to clearly show the field), rendering the largest eigenvector of the tensor field with unit length. The segmentation and parameterization steps created 25 segments, shown in the second image. The third and fourth images show $5,000$ samples, as points and as splats. Using previous methods such as [11], this model would be difficult to cut and parameterize with minimal stretch. Fig. 10 shows similar results for $4,000$ samples on an aircraft wing model; in this example, the tensor field comes from the principal curvatures of the mesh, so the anisotropy is consistent with the surface curvature. Curvatures were computed using the *integral invariants* method [44], and samples were placed according to mean curvature. Appropriately, few samples are present in the flatter regions of the wing. In contrast, the density increases, along with the anisotropy due to the almost-cylindrical shape, close to the leading and trailing edges of the wing. An additional denser sampling occurs in a ridge that runs the length of the wing close to the trailing edge. Fig. 11 shows a close up of part of the wing. The field contained 12 singularities, while segmentation and parameterization resulted in 42 segments. Using previous stratified surface sampling methods, a more complex cut would be required to unwrap the mesh, and considerable stretch would be introduced by the parameterization.

Fig. 4 shows the Stanford Bunny model sampled with respect to a user-defined tensor field with 4 singularities, with a uniform anisotropy of 3:1. Segmentation and parameterization resulted in 32 segments. Image (a) shows the tensor field on the model, (b) the segmentation, with a single segmented highlighted, (c) the

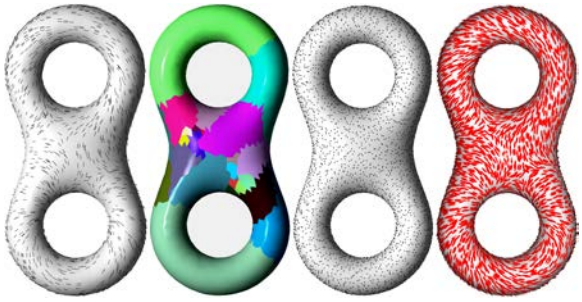

Fig. 9: *A 2-torus segmented and sampled with respect to a user defined anisotropic tensor field.*

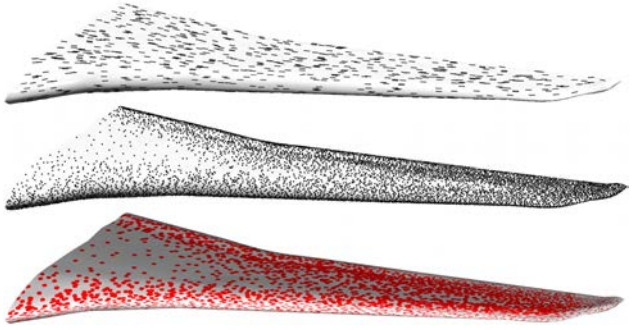

Fig. 10: *An aircraft wing sampled with respect to its principal curvatures.*

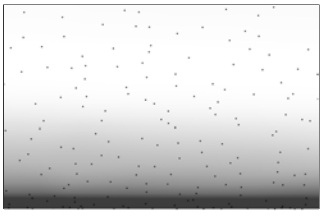

Fig. 11: *A close up of the aircraft wing sampled with respect to its principal curvatures.*

segment parameterised with respect to the field, (d) a reduced-level Hilbert curve generated on the segment, with respect to the field, (e) the segment, sampled anisotropically, and (f) $8,000$ splats on the model. Fig. 12 shows the Stanford Dragon model, sampled from a user-defined field with 4 singularities and a constant anisotropy of 4:1. For this complex model, segmentation and parameterization resulted in $1,622$ segments. The top image shows the tensor field, the remaining images show the model sampled with $10,000$ points and elliptical splats aligned to the field; left, isotropically sampled, right, anisotropically sampled. Note the large reduction in white space in the anisotropic sampling. Whilst this example demonstrates that the method works well with two orders of magnitude more segments than the other examples, it also highlights potential improvements that could be made to the algorithm. For example, cuts could be made to unwrap high curvature areas of the model, such as spines on the left hind leg and head, which would reduce the number of segments required.

In order to demonstrate use of our method for non-photorealistic rendering, we show examples of our sampling rendered using a stroke-based renderer. Fig. 13 shows the Stanford Bunny, sampled with $15,000$ points, rendered using acrylic paint, crayons and watercolor styles. Fig. 14 shows the Venus mesh, sampled with $15,000$ points, rendered using acrylic paint, pencils and inks. Our

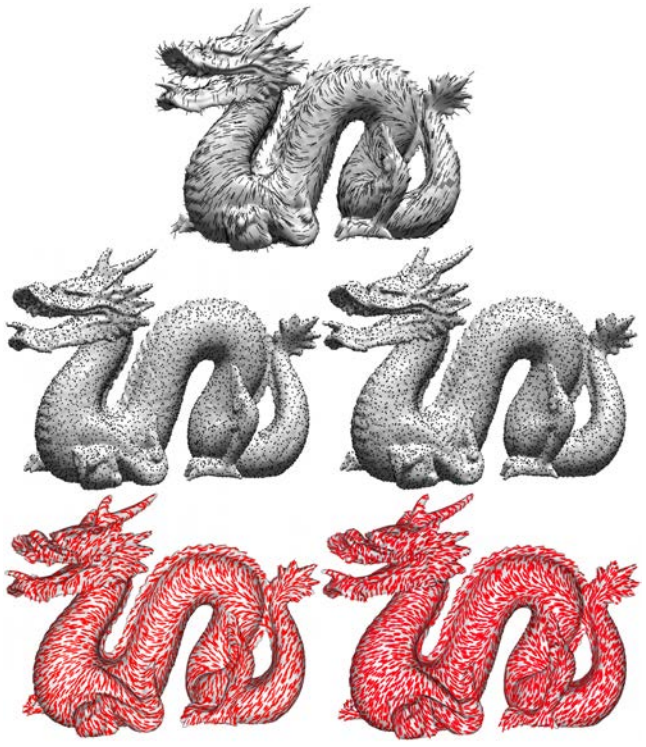

Fig. 12: *Anisotropic low-discrepancy sampling of Stanford Dragon. Top: input tensor field prescribing sampling density and spacing. Remaining images:* $10,000$ *samples as points and elliptical splats; left: isotropic sampling, right: anisotropic sampling.*

Fig. 13: *Bunny,* $15,000$ *points, rendered with crayons, inks, and watercolors.*

Fig. 14: *Venus,* $40,000$ *points, rendered with acrylics, pencils and inks.*

sampling method results in a globally-correlated sampling, but with local non-determinism, resulting in a well distributed set of strokes that look naturally placed.

Table 1 gives timings for a Intel Core i7 920 CPU with an NVIDIA GTX580 GPU for the four meshes discussed, broken into segmentation and parameterization, curve generation, and sampling. Most

TABLE 2: *The discrepancy gradient for increasing sample set sizes for Poisson disk sampling and our method.*

Method	**Rectangle**	**Circle**	**Triangle**
Poisson Disk	-0.56	-0.56	-0.54
Our method	-0.73	-0.72	-0.70

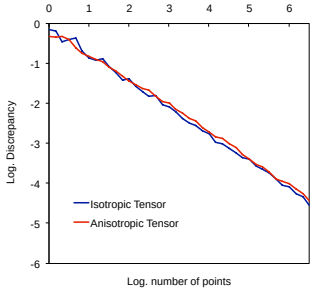

Fig. 15: *Discrepancy of sample distributions generated with respect to an isotropic and anisotropic uniform tensor field.*

of the time in curve generation is taken in computing dilated raster images of the parameterized mesh segments. Curve generation and sampling stages are parallelized, and thus the longest thread execution time is reported. Accurate timing of the sampling phase is difficult due to the very short execution time, but after prior steps have been performed, is done in real-time.

7.2 Discrepancy

We next evaluate the sampling quality of our method, first measuring the *star discrepancy* [2]. We limit this to planar cases due to the difficulty of computing meaningful quality measures on general surfaces with arbitrary tensor fields. Fig. 15 shows the discrepancy for both a constant isotropic and constant anisotropic point sampling with a ratio of 4:1 in $[0,1]^2$ using our sampling approach. Tests were performed for sample sets of size $N = 2^l$ and $N = 2^l + 2^{l-1}$ for $l = 1, \ldots, 20$, using rectangular sampling shapes (as in [13]), and averaged over three runs. A graph was plotted of log of the discrepancy against $\log(N)$. The area of the sample rectangles was computed with respect to the metric defined by the constant tensor in $[0,1]^2$ to account for the desired anisotropy. Gradients of least squares lines fitted to the isotropic and anisotropic cases were 0.74 and 0.72, respectively, agreeing with previous results for stratified sampling in the literature, and consistent with the conclusion that our anisotropic sampling method produces low-discrepancy distributions.

We also compare the discrepancy of uniform Poisson disk samples to our method. Tests were executed as above, except that sample shapes were varied—in addition to standard axis-aligned rectangles, triangles with one point fixed at the origin and quarter-circles with the center at the origin were used. Table 2 shows the gradients for a least squares line for both distributions, for each sample shape. Our method, producing stratified samples, shows a gradient considerably steeper than that for Poisson disk sampling. Stratified sampling has previously been shown to have a lower discrepancy than Poisson disk sampling [9].

7.3 Algorithm Verification

This section demonstrates the robustness of our approach with respect to (i) angular error between the Hilbert curve and the tensor field, (ii) errors that may occur at boundaries between segments, and (iii) quantization errors that may occur as the number of segments increases while keeping sampling rate constant.

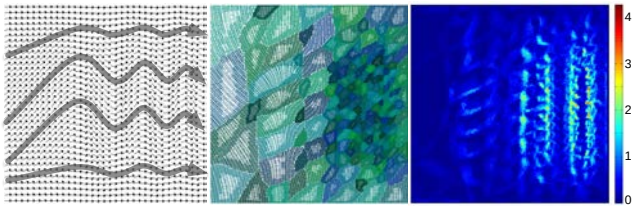

Fig. 16: *Left: tensor field (largest eigenvector shown) in a planar region. Middle: Hilbert curve segmentation according to tensor field. Angular variance threshold $\gamma = 5°$. Right: angular error between input field and Hilbert curve.*

7.3.1 Angular Error

Our sampling method relies on the local alignment of the Hilbert curve with the tensor field (see Fig. 5). After a mesh segment has been parameterized such that the directional variation of the field within that segment is kept small (see Section 5), it is checked for *validity*. This requires that no angle between the projection of the largest eigenvector $\mathbf{m_1}$ and the basis vector of the parameter domain should be greater than a parameter γ. In order to experimentally investigate this, we define a tensor field on a planar mesh that increases in speed of variation from left to right. The mesh is segmented automatically according to the field and parameterized with $\gamma = 5°$. The angular error between the Hilbert curve and the tensor field is then measured and plotted as an image (see Fig. 16). The maximum error in this example is less than $4.5°$, with the higher errors occurring in the more complex region of the field. If γ were reduced, segmentation would increase to allow for this required increase in accuracy.

7.3.2 Boundary Error

To investigate the presence of sampling discontinuities along the segment boundaries, we generated a single complex boundary on a planar mesh, defining two distinct segments. For this experiment, a constant, isotropic tensor field was used. We computed the discrepancy, as defined in Section 7.2, for the mesh segmented in this way, and for the same mesh and field with no boundary (i.e. as a single segment). Fig. 17 shows the segmentation and discrepancy results. The graph indicates that the discrepancy of the sample distribution is not made worse by the introduction of the boundary (also see further discrepancy testing below).

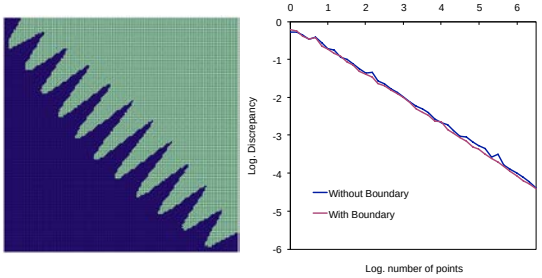

Fig. 17: *Left: complex boundary defining two segments. Right: discrepancy of sample distributions generated according to boundary, and generated without boundary.*

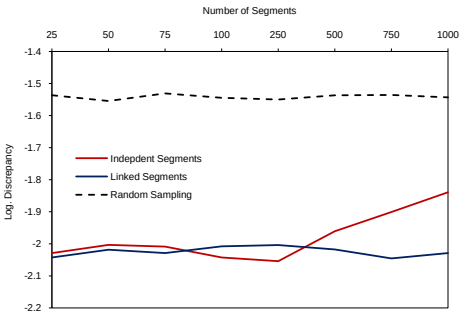

Fig. 18: *Discrepancy of* $1,000$ *sample points with an increasing degree of segmentation. Segments were left independent, or linked into several pairwise-adjacent sequences. Random sampling is shown for comparison.*

7.3.3 Quantization Error

To investigate how the number of segments for a mesh affects the quality of the sampling, we increased the number of segments while the number of point samples was held fixed. We utilized the same planar mesh and tensor field as in Fig. 16. As a uniform segmentation, or stratification, may positively influence the sampling, potentially masking an error, a relatively complex field was used in order to generate a more complex segmentation. By varying the parameter γ, the mesh was automatically segmented into between about 25 and $1,000$ segments, and sampled with $1,000$ points. The discrepancy was then computed as defined in Section 7.2 (see Fig. 18). Without our post-processing step that links segments together (see Section 6.2), as the number of *independent* segments grows over about 300, or one third of the number of points, the discrepancy can be seen to increase. When compared to a random sampling of the domain, this is a significant change. Such a scenario, where such a large number of segments is sampled so sparsely is unlikely to occur unless the mesh density is insufficient to represent a complex tensor field. However, if segments are linked using our post-processing method, the resulting regions are far larger, essentially removing the error (see Fig. 18, *linked* segments).

7.4 Spectral Analysis

We now investigate further properties of the sampling using spectral analysis methods [45]. Fig. 19 shows a spectral analysis of uniform isotropic and anisotropic tensor cases in the plane. In the anisotropic case, a constant anisotropy of 4:1 is used. For both cases, we show the single sample distribution (left), the mean periodogram (middle), radially averaged power spectrum density (RAPSD) (top right), and anisotropy (bottom right) of the point sampling. The anisotropy measures the radial symmetry. All results are averaged over 8 runs. Due to the random element of our sampling, there are no sharp concentric rings in the mean periodogram, unlike for Poisson disk sampling (see [6]), and the RAPSD also does not peak multiple times in the mid frequencies. The distribution also has an absence of low-frequency components and very consistent mid and high frequencies, demonstrating the good blue noise characteristics of our sample distributions.

7.5 Differential Domain Analysis

We apply recent work on *differential domain analysis* [8], building upon [10], [18], to allow detailed analysis of samplings with

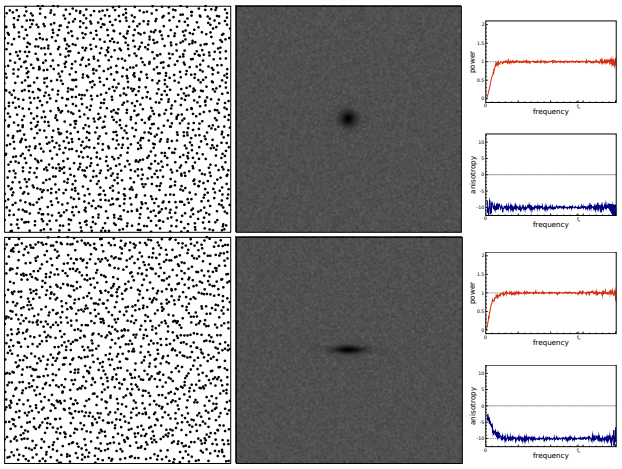

Fig. 19: *Spectral analysis,* $1,500$ *points. Samples, mean periodogram, radially averaged power spectrum density and anisotropy. Isotropic (top), anisotropic (bottom), planar.*

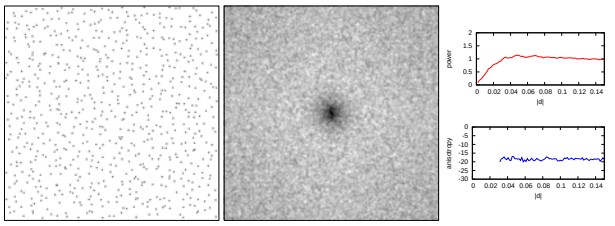

Fig. 20: *Differential domain analysis of our sampling.* 720 *samples; uniform, isotropic sampling. Sampling, spectrum image, radial mean and anisotropy, averaged over* 10 *runs.*

varying density and anisotropy, including on mesh surfaces. All tests were run with identical parameters to those in [8], making the results directly comparable. Tests on Poisson disk sampling and certain other methods, as well as a thorough description of these parameters and the density functions used, are given there. Results should appear consistent with the isotropic case, as the sampling is analyzed in the differential domain. Fig. 20 shows 720 samples uniformly generated using our sampling method, and the resulting spectrum image, radial mean and anisotropy averaged over 10 runs. We set $\epsilon = 10$ and used a Gaussian kernel. Fig. 21 shows density-controlled samples generated using our sampling method and a 2D Gaussian function ($2,500$ samples in $[0,2]^2$), and the Balzer function [17] ($10,000$ samples in $[0,4]^2$), and the resulting spectrum, radial mean and anisotropy averaged over 10 and 4 runs respectively with $\epsilon = 12$ and a Gaussian kernel. Fig. 22 shows anisotropic samples generated using our sampling method using a shear function ($2,500$ samples in $[0,2]^2$) and a perspective function [10] ($10,000$ samples in $[0,4]^2$), and the resulting spectrum, radial mean and anisotropy averaged over 10 and 4 runs respectively with $\epsilon = 12$ and a Gaussian kernel. Fig. 23 shows the Bunny and 2-torus meshes with uniform and density-controlled samples (according to curvature) generated using our method, and the resulting spectrum, radial mean and anisotropy. Both meshes were normalized to have a surface area of 1, and for each example $3,000$ samples were generated, with the results averaged over 8 runs. We set $\epsilon = 10$ and used a Gaussian kernel.

The results of the differential domain analysis show that our method produces consistent samplings, for uniform, density-controlled and anisotropic functions. Anaylsis results are compa-

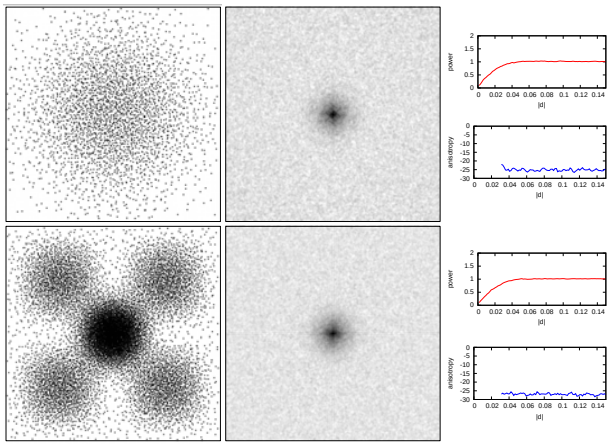

Fig. 21: *Differential domain analysis of our sampling method, isotropic, density-controlled sampling. 2D Gaussian blob (*$2,500$ *samples in* $[0,2]^2$*) and Balzer function (*$10,000$ *samples in* $[0,4]^2$*). Sampling, spectrum image, radial mean and anisotropy averaged over* 10 *and* 4 *runs respectively.*

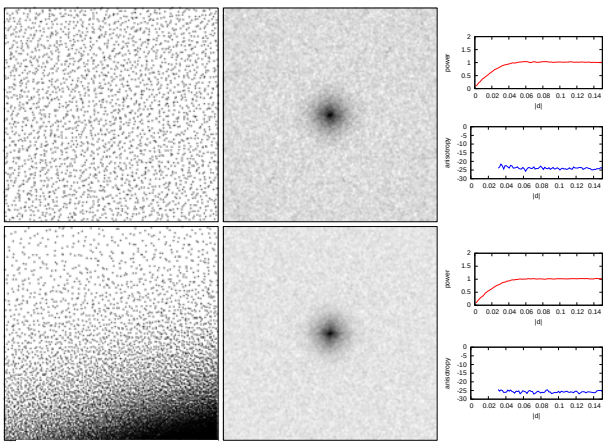

Fig. 22: *Differential domain analysis of our sampling method, anisotropic sampling. Shear (*$2,500$ *samples in* $[0,2]^2$*) and perspective functions (*$10,000$ *samples in* $[0,4]^2$*). Sampling, spectrum image, radial mean and anisotropy averaged over* 10 *and* 4 *runs respectively.*

rable to those for uniform jittered/stratified sampling [8]. Both the mean and anisotropy plots are very stable throughout, indicating that our sampling method can accurately represent the functions.

Consistently with stratified sampling, our method introduces a limited degree of randomness in each sample, avoiding regularities in the point distribution, and the very sharp local neighborhood spacing prevalent in Poisson disk methods. This randomness is limited to the local stratum of the sample, which is mapped from the parameter domain onto the surface in a manner aligned with the tensor field and with spacing stretched according to the local anisotropy. This ensures that the randomness does not destroy the anisotropy. However, even in a case of global randomness, if we were, for example, to produce a random, isotropic, sampling in the plane, and stretch it along one axis, we would introduce an anisotropic spacing. The anisotropy of the sample distributions produced is, however, less visually clear than in a regularly sampled domain, as the distance between samples is not constant and the samples are not arranged on a grid (see Fig. 8). Results in this section demonstrate that whilst the anisotropy in our samplings may not be visually obvious, it is consistently present when a formal analysis is performed.

8 Limitations and Future Work

We have presented an algorithm for generating anisotropic, low-discrepancy, stratified point samples on triangle meshes. Our method allows the user to control the sampling density and anisotropy via a tensor field. After preprocessing, it can sample and re-sample meshes in real-time.

Experiments designed to validate our algorithm show that despite potential for problems due to (i) boundaries between segments on the mesh, (ii) quantization errors when assigning samples to segments, and (iii) angular errors in correctly following the prescribed tensor field, our method performs as intended, and these potential issues have no negative effects on the sampling output. Anisotropic discrepancy measures show that our sampling is consistent with results for stratified sampling on the plane and on meshes. Frequency and differential domain analysis confirm that samples in both the plane and on surface meshes correctly represent the input tensor. Spectral analysis shows that the local neighbourhood of Poisson disk samples is more regular than that of samples produced by our approach. Discrepancy measures show that the global coverage of the meshes using our method is better.

The lack of local regularity in our anisotropic stratified sampling appears highly usefull for artistic rendering. However, whilst strucutred Poisson disk methods and quasi-random sequences such as [26] are less suitable for artistic approaches, more recent Poisson disk methods introducing less regularity may be interesting to consider. Our method seems less generally applicable to remeshing, as it would be much harder to guarantee triangle equilaterality. However, a less regular tessellation may result in fewer illumination and shadowing artifacts during rendering.

A requirement for methods like ours is that of a *moderate* quality tensor field as an input. If the field is poorly defined, e.g., in isotropic areas, our approach may result in a considerable over-segmentation. If the field is user-defined, this is generally avoidable, but may be more problematic if the field is derived from intrinsic data such as principal curvatures; the field may even inherit properties from the field generation algorithm. Gaussian smoothing lessens the problem, but is unlikely to eliminate it. A possible solution may be to merge these isotropic segments.

Many field singularities may result in a complex segmentation, and thus considerably increased computation. Whilst we have no specific requirements on the regularity of the triangulation, or the resolution or topological complexity of the input mesh, increased topological complexity will rapidly increase the number of segments that are generated, and thus the computational requirements. Their effects on the tractability of the problem should be noted. One possible approach to reduce the computational cost of handling singularities would be to locate the presence of a singularity in a triangle by computing its winding number [29].

An obvious limitation of our method is its reliance on mesh parameterization. The optimization process aligns segments of the mesh as closely to the field as possible. The validity of this alignment is governed by a parameter limiting the maximum allowed error. However, whilst this is both controllable and

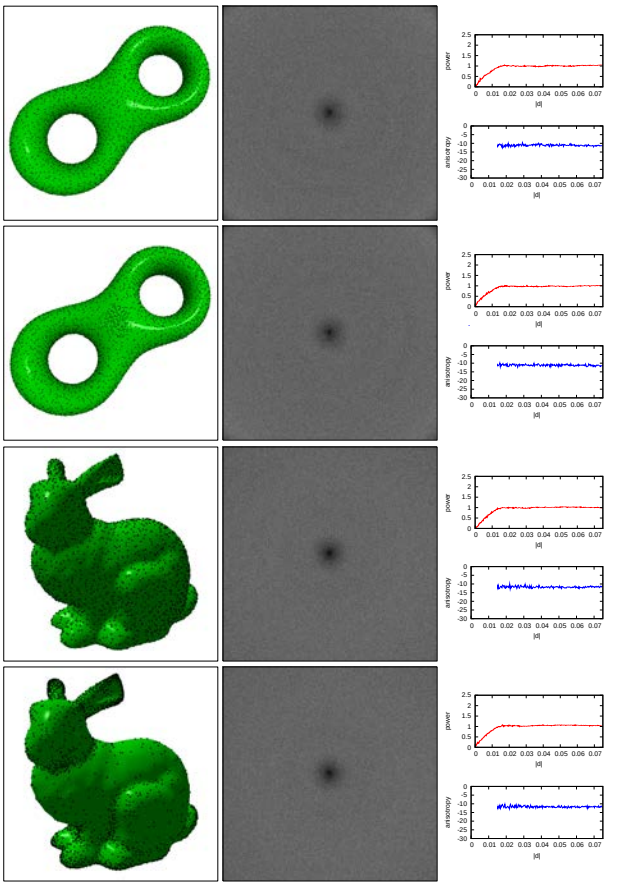

Fig. 23: *Differential domain analysis of our sampling method with uniform and density-controlled surface sampling for the Stanford Bunny and 2-torus model (*$3,000$ *samples). Point sampling, spectrum image, radial mean and anisotropy averaged over* 8 *runs.*

demonstrated to have little impact, an error will exist, especially near field singularities. The presence of this error is inherently linked to the discretization of the tensor field and the mesh, but a method that avoids the parameterization may provide a solution which is closer to the ideal result.

With the control the user has over the sampling, our method has many potential applications in such areas as interactive artistic rendering, fur and hair rendering, point-based rendering, surface measurement, and global illumination. Here we have shown splat-based rendering to visualize the sample distributions, and stroke-based rendering to illustrate the visual quality of our sampling method in non-photorealistic applications. Many other areas with application-dependent quality measures remain to be investigated.

Our method's real-time re-sampling capability is well-suited to fine-grain view-dependent and level of detail rendering. A more difficult problem to is a lack of temporal coherence between successive samplings, leading to undesirable rendering artifacts. A solution may be to locally add samples instead of resampling the entire curve, in such a way that spatial properties of the existing sampling are not destroyed. In addition, changes to the field or mesh will necessitate further preprocessing due to local re-adaptiation of the Hilbert curve. Model deformations may be handled by segment subdivision, and local re-computation of surface properties, but a global change in the field may require complete re-segmentation and parameterization. Future work will look at solving these problems, and the extension of [36] and [30] for generalized tensor field segmentation on triangle meshes.

Acknowledgments

We thank L.-Y. Wei and R. Wang for help with their differential domain analysis software, A. Agraz for help with the stroke-based rendering pipeline, S. Shellard for help creating stylized brush strokes, and the three artists for their stippling examples. The authors acknowledge funding from the Welsh Government via the One Wales Research Institute for Visual Computing (RIVIC).

References

[1] R. A. Ulichney, "Dithering with blue noise," *Proc. of IEEE*, vol. 76, no. 1, pp. 56–79, 1988.

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

[3] T. Isenberg, P. Neumann, S. Carpendale, M. C. Sousa, and J. A. Jorge, "Non-photorealistic rendering in context: an observational study," in *Proc. Int. Symp on Non-photorealistic Animation and Rendering*, 2006, pp. 115–126.

[4] R. Maciejewski, T. Isenberg, W. Andrews, D. Ebert, M. Sousa, and W. Chen, "Measuring stipple aesthetics in hand-drawn and computer-generated images," *Comp. Graph. and App., IEEE*, vol. 28, no. 2, pp. 62–74, 2008.

[5] A. Hertzmann, "A survey of stroke-based rendering," *IEEE Comp. Graph. and App.*, vol. 23, no. 4, pp. 70–81, 2003.

[6] A. Lagae and P. Dutré, "A comparison of methods for generating poisson disk distributions," *Comp. Graph. Forum*, vol. 27, no. 1, pp. 114–129, 2008.

[7] Q. Du, V. Faber, and M. Gunzburger, "Centroidal Voronoi tessellations: applications and algorithms," *SIAM Review*, vol. 41, no. 4, pp. 637–676, 1999.

[8] L.-Y. Wei and R. Wang, "Differential domain analysis for non-uniform sampling," in *Proc. ACM SIGGRAPH*, 2011, pp. 50:1–50:10.

[9] D. P. Dobkin, D. Eppstein, and D. P. Mitchell, "Computing the discrepancy with applications to supersampling patterns," *ACM Trans. Graph.*, vol. 15, no. 4, pp. 354–376, 1996.

[10] H. Li, L.-Y. Wei, P. V. Sander, and C.-W. Fu, "Anisotropic blue noise sampling," *ACM Trans. Graph.*, vol. 29, pp. 167:1–167:12, 2010.

[11] J. A. Quinn, F. C. Langbein, and R. R. Martin, "Low-discrepancy point sampling of meshes for rendering," in *Symp. on Point Based Graph.*, 2007, pp. 19–28.

[12] M. Steigleder and M. McCool, "Generalized stratified sampling using the Hilbert curve," *Journal Graph. Tools*, vol. 8, no. 3, pp. 41–47, 2003.

[13] J. A. Quinn, F. C. Langbein, R. R. Martin, and G. Elber, "Density-controlled sampling of parametric surfaces using adaptive space-filling curves," in *Proc. Geometric Modeling and Processing*, 2006, pp. 658–678.

[14] B. Lévy and Y. Liu, "Lp centroidal voronoi tessellation and its applications," *ACM Trans. Graph.*, vol. 29, pp. 119:1–119:11, 2010.

[15] R. Fattal, "Blue-noise point sampling using kernel density model," *ACM Trans. Graph.*, vol. 30, no. 4, pp. 48:1–48:12, 2011.

[16] D. Cline, S. Jeschke, K. White, A. Razdan, and P. Wonka, "Dart throwing on surfaces," *Comp. Graph. Forum*, vol. 28, no. 4, pp. 1217–1226, 2009.

[17] M. Balzer, T. Schlömer, and O. Deussen, "Capacity-constrained point distributions: a variant of lloyd's method," *ACM Trans. Graph.*, vol. 28, pp. 86:1–86:8, 2009.

[18] J. Bowers, R. Wang, L.-Y. Wei, and D. Maletz, "Parallel poisson disk sampling with spectrum analysis on surfaces," *ACM Trans. Graph.*, vol. 29, pp. 166:1–166:10, 2010.

[19] Y. Miao, R. Pajarola, and J. Feng, "Curvature-aware adaptive re-sampling for point-sampled geometry," *Comp. Aided Design*, vol. 41, no. 6, pp. 395–403, 2009.

[20] D. Nehab and P. Shilane, "Stratified point sampling of 3D models," in *Symp. on Point-Based Graph.*, 2004, pp. 49–56.

[21] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein, "Real-time hatching," in *Proc. ACM SIGGRAPH*, 2001, pp. 579–584.

[22] S. Y. Kim, R. Maciejewski, T. Isenberg, W. M. Andrews, W. Chen, M. C. Sousa, and D. S. Ebert, "Stippling by example," in *Proc. Int. Symp. on Non-Photorealistic Animation and Rendering*, 2009, pp. 41–50.

[23] D. Martín, G. Arroyo, M. V. Luzón, and T. Isenberg, "Example-based stippling using a scale-dependent grayscale process," in *Proc. Int. Symp. on Non-Photorealistic Animation and Rendering*, 2010, pp. 51–61.

[24] L.-Y. Wei, "Multi-class blue noise sampling," *ACM Trans. Graph.*, vol. 29, pp. 79:1–79:8, 2010.

[25] C.-H. Huang, K.-C. Chang, and C. Wen, "Non-iterative stippling of greyscale three-dimensional polygon meshed models," *Comp. Vision, IET*, vol. 4, no. 2, pp. 138–148, 2010.

[26] T. Umenhoffer, L. Szécsi, and L. Szirmay-Kalos, "Hatching for motion picture production," *Comp. Graph. Forum*, vol. 30, no. 2, pp. 533–542, 2011.

[27] J. Quinn, "Low-discrepancy point sampling of 2D manifolds for visual computing," Ph.D. dissertation, Cardiff University, 2009.

[28] L. Feng, I. Hotz, B. Hamann, and K. Joy, "Anisotropic noise samples," *IEEE Trans. Vis. and Comp. Graph.*, vol. 14, no. 2, pp. 342–354, 2008.

[29] E. Zhang, K. Mischaikow, and G. Turk, "Vector field design on surfaces," *ACM Trans. Graph.*, vol. 25, no. 4, pp. 1294–1326, 2006.

[30] E. Zhang, J. Hays, and G. Turk, "Interactive tensor field design and visualization on surfaces," *IEEE Trans. Vis. and Comp. Graph.*, vol. 13, no. 1, pp. 94–107, 2007.

[31] J. Palacios and E. Zhang, "Rotational symmetry field design on surfaces," *ACM Trans. Graph.*, vol. 26, no. 3, 2007.

[32] Y.-K. Lai, M. Jin, X. Xie, Y. He, J. Palacios, E. Zhang, S.-M. Hu, and X. Gu, "Metric-driven rosy field design and remeshing," *IEEE Trans. Vis. and Comp. Graph.*, vol. 16, no. 1, pp. 95–108, 2010.

[33] C.-Y. Yao, M.-T. Chi, T.-Y. Lee, and T. Ju, "Region-based line field design using harmonic functions," *IEEE Trans. on Vis. and Comp. Graph.*, vol. 18, pp. 902–913, 2012.

[34] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr, "Discrete differential geometry operators for triangulated 2-manifolds," in *VisMath III*, 2002, pp. 35–57.

[35] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun, "Anisotropic polygonal remeshing," in *Proc. ACM SIGGRAPH*, 2003, pp. 485–493.

[36] T. Delmarcelle and L. Hesselink, "The topology of symmetric, second-order tensor fields," in *Proc. of the Conf. on Vis.*, 1994, pp. 140–147.

[37] N. Ray, W. C. Li, B. Lévy, A. Sheffer, and P. Alliez, "Periodic global parameterization," *ACM Trans. Graph.*, vol. 25, no. 4, pp. 1460–1485, 2006.

[38] Y.-K. Lai, S.-M. Hu, R. R. Martin, and P. L. Rosin, "Rapid and effective segmentation of 3D models using random walks," *Comp. Aided Geom. Design*, vol. 26, no. 6, pp. 665–679, 2009.

[39] X. Tricoche, "Vector and tensor topology simplification, tracking, and visualization," Ph.D. dissertation, Schriftenreihe / Fachbereich Informatik, Universität Kaiserslautern, 2002.

[40] C. Auer and I. Hotz, "Complete tensor field topology on 2d triangulated manifolds embedded in 3d," *Comp. Graph. Forum*, vol. 30, no. 3, pp. 831–840, 2011.

[41] C. Auer, C. Stripf, A. Kratz, and I. Hotz, "Glyph- and texture-based visualization of segmented tensor fields," in *Int. Conf. on Inf. Vis. Theory and App.*, 2012, pp. 670–677.

[42] Y.-K. Lai, S.-M. Hu, and R. R. Martin, "Surface mosaics," *The Visual Comp.*, vol. 22, no. 9-11, pp. 604–611, 2006.

[43] P. Guigue and O. Devillers, "Fast and robust triangle-triangle overlap test using orientation predicates," *Journal of Graph., GPU, and Game Tools*, vol. 8, no. 1, pp. 25–42, 2003.

[44] Y.-K. Lai, S.-M. Hu, and T. Fang, "Robust principal curvatures using feature adapted integral invariants," in *SIAM/ACM Joint Conference on Geometric and Physical Modeling*, 2009, pp. 325–330.

[45] T. Schlömer and O. Deussen, "Toward a standardized spectral analysis of point sets with applications in graphics," University of Konstanz, Germany, Tech. Rep., 2010.

Jonathan Quinn received his PhD from Cardiff University in 2010. He is currently a Research Officer for the One Wales Research Institute of Visual Computing at the School of Computer Science & Informatics, Cardiff University. His research interests include computer graphics, geometry processing and sampling, and image-based modelling.

Frank Langbein received a PhD in 2003 from Cardiff University on "Beautification of Reverse Engineered Geometric Models" and a Diploma in Mathematics from Stuttgart University in 1998. He is currently a lecturer at the School of Computer Science & Informatics, Cardiff University, working on modelling, simulation, control and machine learning applied to geometry, quantum technology, chemical synthesis and perception. He is a member of the AMS and the IEEE.

Yu-Kun Lai received his bachelors degree and PhD degree in computer science from Tsinghua University in 2003 and 2008, respectively. He is currently a lecturer of visual computing in the School of Computer Science & Informatics, Cardiff University. His research interests include computer graphics, geometry processing, image processing and computer vision.

Ralph Martin leads the Visual Computing research group at Cardiff University, and is Director of Scientific Programmes of the One Wales Research Institute of Visual Computing. His output includes over 200 papers and 12 books covering such topics as solid modelling, surface modelling, reverse engineering, intelligent sketch input, mesh processing, video processing, computer graphics, vision based geometric inspection, and geometric reasoning. He is on the editorial boards of "Computer Aided Design", "Computer Aided Geometric Design", "Geometric Models", and several other journals.