Anisotropic Sampling of Planar and Two-Manifold Domains for Texture Generation and Glyph Distribution

Andrea Kratz, Daniel Baum, and Ingrid Hotz

Abstract—We present a new method for the generation of anisotropic sample distributions on planar and two-manifold domains. Most previous work that is concerned with aperiodic point distributions is designed for isotropically shaped samples. Methods focusing on anisotropic sample distributions are rare, and either they are restricted to planar domains, are highly sensitive to the choice of parameters, or they are computationally expensive. In this paper, we present a time-efficient approach for the generation of anisotropic sample distributions that only depends on intuitive design parameters for planar and two-manifold domains. We employ an anisotropic triangulation that serves as basis for the creation of an initial sample distribution as well as for a gravitational-centered relaxation. Furthermore, we present an approach for interactive rendering of anisotropic Voronoi cells as base element for texture generation. It represents a novel and flexible visualization approach to depict metric tensor fields that can be derived from general tensor fields as well as scalar or vector fields.

Index Terms—Tensor visualization, texture generation, anisotropic sampling

1 Introduction

We introduce a novel technique for generating unstructured sample distributions on planar and two-manifold domains for visualizing metric tensor fields. Due to the anisotropy of metric tensors, our focus is samples that vary in size, shape, and orientation (anisotropic samples). Such sample sets improve glyph-based visualizations [1], [2], [3], [4], [5] and build a basis for texture generation [3], [5]. The use of textures for visualization provides high flexibility to encode features and, therefore, has a great potential for the visualization of tensor fields.

Our main contribution is the design of an algorithm for the generation of anisotropic sample distributions that works both on planar and two-manifold domains. The extension of the planar case to two-manifold domains implicates additional challenges. To achieve interactive results, a compromise between an approach with a solid theoretical basis and a more practical solution is needed. The design of our solution is guided by the following requirements: 1) The sample generation should be time efficient. 2) It should only depend on very few intuitive parameters. 3) The sample distribution should have a random character (no noticeable pattern) and the sample domain should be covered densely (avoid large empty areas as well as cluttered areas which would be visually distracting).

The most critical part for the extension to two-manifold domains is the computation of distances and the generalization of distance measures. Our technique avoids costly distance computations whenever possible. Where distance computations are needed, we use fast approximations which are sufficient for our purposes and result in sample sets that fulfill our requirement of good *visual quality* (see requirement 3). Thus, our approach is not only an extension to two-manifold domains. It also significantly speeds up the generation of anisotropic sample distributions for the planar case compared to previous approaches (e.g., [5]). Due to this speedup, our approach also enables interactive slicing through three-dimensional (3D) data sets.

As second contribution, we present an interactive visualization technique that is based on the previously computed sample set. Kratz et al. [5] have recently shown that generalized anisotropic Voronoi diagrams in the planar domain can be efficiently used to represent the underlying metric. The Voronoi cells can be texturized using various types of textures to encode features of interest. Our work is motivated by Kratz [5] and represents an extension and generalization of their approach resulting in an interactive rendering method of anisotropic Voronoi cells and their texturization.

To demonstrate the potential of our approach, we present visualizations of several examples for the planar as well as the two-manifold case. In particular, we visualize the formation of endothelia cells of a blood vessel in accordance with the simulation of a blood flow in an aneurysm. Since endothelia cells naturally have shapes that are similar to anisotropic Voronoi cells, this is an application that directly benefits from our approach.

2 Related Work

The generation of sample distributions with specific properties is a central research topic in computer graphics with applications in rendering, visualization, and geometry

• *The authors are with Zuse Institute Berlin, Takustr. 7, Berlin 14195, Germany. E-mail: {kratz, baum, hotz}@zib.de.*

Manuscript received 16 Aug. 2012; revised 8 Jan. 2013; accepted 30 Apr. 2013; published online 16 May 2013.
Recommended for acceptance by P. Cignoni.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2012-08-0163.
Digital Object Identifier no. 10.1109/TVCG.2013.83.

processing. A huge amount of related work exists, most of which focuses on very specific requirements and applications. Since an exhaustive review of related methods from sampling theory and geometry processing is out of scope of this paper, we refer the reader to the following survey articles. For a deeper insight into remeshing approaches, we refer to the article by Alliez et al. [6]. And for an overview about the creation of aperiodic isotropic sample distributions, we refer to the articles by Lagae et al. [7], [8], [9]. In the following, we focus on work that is related to the approach presented in this paper.

Many previous work in generating sample distributions focuses on the generation of Poisson-disk distributions [8]. Distributions with this property cover the sample domain densely while maintaining a minimum separation given by a specified radius, the Poisson-disk radius. Common basic algorithms to create such distributions are dart throwing [10] and Lloyd relaxation [11]. These algorithms are expensive and have a high memory requirement. Thus, several enhanced methods for generating Poisson-disk distributions have been presented [8]. Among these, tile-based methods [7], [12] have gained special interest. Tile-based methods generate a small set of tiles *once*, where each tile has the Poisson-disk characteristics. Thus, tile-based methods have a low memory requirement and are suited for real-time applications. Since these methods are difficult to generalize to the anisotropic case, where the samples vary in size, shape, and orientation, work on tile-based anisotropic samplings is rare [13].

The objective of mesh generation most often is to find a mesh that is as coarse as possible and as fine as needed to represent all properties of the shape of the underlying object, or to compute simulations or interpolations on it. In this context, anisotropic meshes are beneficial for simulating functions with a strong directional character. The strength of such meshes is that the shape of the mesh elements can be controlled by an underlying anisotropy function (metric tensor field). Various approaches to tackle this problem have been presented [14], [15], [16]. In [15], for example, anisotropy is incorporated in the common *circumcircle test* to compute anisotropic Delaunay meshes. Other solutions build on the construction of an anisotropic Voronoi diagram, for example, [17], [18]. In [17], [19], the triangle mesh is then defined as the dual of the Voronoi diagram. These approaches are mathematically sound but computationally expensive and hard to generalize to the two-manifold case.

In contrast to mesh generation, remeshing algorithms have the goal to improve a given mesh with respect to specific criteria. Most advanced remeshing approaches rely on surface parameterizations. Similarly, parameterizations are used for anisotropic surface sampling [4], [13] and meshing of curved surfaces [15]. While for meshing algorithms an accurate representation of surface details is of high importance, this is not the case for our purpose, which is texturing the surface. This simplifies the sampling process in many ways. Hence, our approach does not require an expensive parameterization.

In visualization, anisotropic sample distributions have been computed to guide the positioning of glyphs [1], [2], [3], [5]. The more a sample distribution reflects the continuous behavior of an input vector- or tensor field, the more informative the final glyph-based visualization becomes. Examples for applications within the context of tensor field visualization have been presented in [20], [21]. Building on the particle-based method of [15], in [1], an approach for planar domains and volumes is presented for diffusion tensors. Here, the input diffusion tensor field is mapped to a potential energy field determining interparticle forces. For the planar case, this approach was extended in [2], which focuses on an improved initial sampling and interactivity by using an isotropic Delaunay triangulation. More exact neighbor computations are achieved with anisotropic Delaunay triangulations [5]. A limitation of these particle force-based algorithms is that they often suffer from many nonintuitive parameters.

Lloyd relaxation [11], [22] was used in [3] to generate anisotropic samplings of planar domains. As Lloyd relaxation relies on Voronoi diagrams, this approach requires the computation of an *explicit* generalized Voronoi diagram including the handling of orphans. One advantage of such a relaxation-based method over particle-based methods is that it is almost parameter-free. However, the computation of an explicit generalized Voronoi diagram on surfaces including the handling of orphans is computationally too expensive for our purposes. Building upon ideas of [3], in [4], dart throwing and relaxation are extended to anisotropic sampling. For this purpose, the euclidean distance metric is replaced by a geodesic one. However, relaxation is only applied on planar domains, while surface sampling is restricted to dart throwing in combination with surface parametrizations. The authors also present a method for the evaluation of anisotropic samples assuming warpable domains. A more general analysis method for nonuniform samplings is presented in [23]. In this paper, we avoid the need of surface parametrizations. The resulting approach is time efficient and needs only few parameters. Our approach also uses relaxation in the two-manifold domain to equalize distances.

3 Overview

The following steps summarize our method for generating anisotropic sample distributions on planar and two-manifold domains. Input is a scalar-, vector- or tensor field:

1. *Define a metric tensor field*. Since *anisotropy design* is application specific, it will be handled separately in Section 7. Until then it should suffice to assume that a suitable metric tensor field (Section 4.1) is given.
2. *Generate an initial sample distribution respecting this metric (Section 5.2)*, where the sample domain is covered densely (no holes) while the samples maintain a minimum distance (no overlaps). To generate such a distribution, we employ anisotropic triangulations (Section 4.2.1) and compute triangle fillrates to find areas that are sparsely populated. In these areas, new samples are added. For more flexible sample distributions, two additional design options are provided: the overall density can be steered by a

parameter η and a spatially varying importance function facilitates a local density adaptation.

3. *Apply a relaxation process (Section 5.3)* that equalizes triangle sizes with respect to the underlying metric tensor field. Thus, more uniform sample distances with respect to the underlying metric tensor field are achieved. The relaxation is based on the anisotropic triangulation

Finally, we present an interactive rendering technique of anisotropic Voronoi cells and their texturization (Section 6).

4 Basic Concepts

This section provides the basic concepts behind the algorithm and summarizes the relevant prerequisites.

The algorithm takes as input: 1) A geometric domain Ω that can be either a planar domain $\Omega \subset \mathbf{R}^2$ with boundary $\partial\Omega$, or a two-manifold domain $\Omega \subset \mathbf{R}^3$ with or without boundary. 2) A two- or three-dimensional (2D/3D) input field given on a uniform, triangulated or tetrahedral grid. The input field can be either scalar-, vector- or tensor-valued. 3) Optionally, a spatially varying importance function can be added to create adaptive sample distributions.

Output of the algorithm is a point distribution that fulfills the Poisson-disk characteristics (Fig. 11) with respect to an underlying metric tensor field.

4.1 Metric Tensor Fields

Regardless of the input field (scalar, vector, or tensor), our algorithm (Section 5) works on a metric tensor field, which is either a direct mapping of the input field or is derived from it. The size, shape, and orientation of the samples depend on the sample positions $\mathbf{p} \in \mathbf{R}^d$ and the local metric $\mathbf{M}(\mathbf{p}) \in \mathbf{R}^{n \times n}$ at position $\mathbf{p}$. Here, d and n are 2 in the planar case and 3 in the two-manifold case (see also Section 7). In either case, the metric tensor $\mathbf{M}$ is represented by an $n \times n$ symmetric positive-definite matrix. To reconstruct $\mathbf{M}$ at arbitrary sample positions in the planar domain, we use component-wise linear, respectively, bilinear interpolation. On surfaces, the tensors are interpolated on a per-triangle basis using barycentric coordinates.

The metric tensor describes anisotropic distances between sample positions $\mathbf{p} \in \Omega$. It can be imagined as an ellipse or ellipsoid, respectively, which is scaled according to the reciprocal eigenvalues and oriented according to the eigenvectors of $\mathbf{M}$ [3]. Consequently, ellipses and ellipsoids build our basic sample shapes.

We assume that the metric tensor field does not vary strongly within a small local environment. The idea behind this assumption is that reasonable visualization results can only be achieved if the variation of the tensor field, compared to the size of the samples, is relatively small. Only with this assumption, the samples are valid representatives for the part of the metric tensor field which they cover (see also [3], [5]).

In the following, we use the term *sample* for elliptical or ellipsoidal sample, and *metric* for a 2D or 3D metric tensor.

4.2 Delaunay Triangulations/Voronoi Diagrams

Given a planar euclidean domain Ω and a point set $P = \{\mathbf{p}_i, i = 1, \ldots, n\} \subset \Omega$, a Voronoi diagram is defined as the set of n Voronoi cells Ω_i for which holds that all points that lie within Ω_i are at least as close to $\mathbf{p}_i$ as to any other point in P.

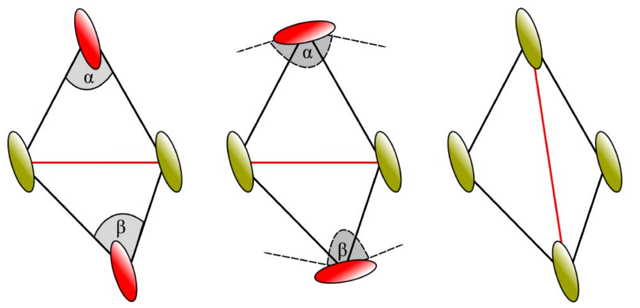

Fig. 1. Modified edge flip. Left: Configuration for which the angle condition is tested (red edge). Here, the euclidean angles are depicted. Middle: To validate the angle condition, we analyze the angles opposite to this edge with respect to the inverse metric (swapped ellipses). In this configuration, the angle condition is not fulfilled. Right: Valid configuration.

The Delaunay triangulation D of Ω w.r.t. P can be defined by a couple of equivalent properties:

- D is the dual graph of the Voronoi diagram.
- D maximizes the sum of the minimum angles.
- D guarantees that the circumcircle of each triangle does not contain any other point of P.

When generalizing these ideas to noneuclidean spaces, utilizing a spatially varying metric, the above-mentioned properties of the Delaunay triangulation are not equivalent anymore. In general, an anisotropic Delaunay triangulation is defined based on the duality property. Unfortunately, it is not guaranteed that the dual of an anisotropic Voronoi diagram actually results in a valid triangulation (e.g., [17], [19]). This is already the case for planar domains and even worse for the manifold case. To avoid this problem, several heuristics have been proposed (e.g., [14], [15], [16]). Among the algorithmic challenges are an efficient geodesic distance computation and the fact that Voronoi cells are no longer bounded by straight lines. We propose to define a triangulation that represents a meaningful neighborhood structure, independently from the Voronoi cells. In the following, we introduce our definitions to generate anisotropic triangulations and to draw anisotropic Voronoi diagrams.

4.2.1 *Anisotropic Triangulation*

To compute a Delaunay triangulation for noneuclidean metrics of parametrized surfaces, Shimada et al. [15] propose an edge-flip algorithm based on a generalized circumcircle test in parametric space. Since we deal with triangulated surfaces without given parametrization, we propose to adapt the *Delaunay angle condition* (see also [24]), also applying an edge-flip algorithm. The angle condition is easy to generalize to the anisotropic case and does not require a parametrization. Note that the edge-flip algorithm only concerns the neighborhood relation of the samples and does not change the original surface.

We check for each edge of a given triangulation if the sum of its opposite angles α and β satisfies the following condition (Fig. 1):

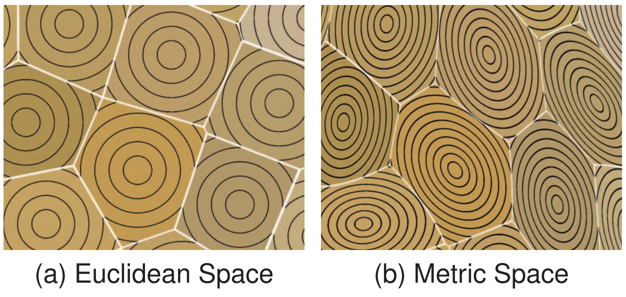

Fig. 2. Isotropic Voronoi cells in the undistorted euclidean space (a) and anisotropic Voronoi cells in the distorted metric space (b). The black lines are the isolines corresponding to the local metrics.

$$\alpha + \beta \leq \pi. \tag{1}$$

Therefore, we solely need to adapt the dot product for computing the angles α and β. For two vectors $\mathbf{v}, \mathbf{w} \in \mathbf{R}^3$ and a local metric $\mathbf{M}$, the scalar product becomes $\langle \mathbf{v}, \mathbf{w} \rangle_{\mathbf{M}} = \mathbf{v}^T \cdot \mathbf{M} \cdot \mathbf{w}$. Accordingly, the length of the vector $\mathbf{v}$ is defined as $\|\mathbf{v}\|_{\mathbf{M}} = \sqrt{\mathbf{v}^T \cdot \mathbf{M} \cdot \mathbf{v}}$. If $\mathbf{v}, \mathbf{w}$ are the two vectors that enclose the angle $\angle(\mathbf{v}, \mathbf{w})$ and share the point $\mathbf{p}$, we have

$$\angle(\mathbf{v}, \mathbf{w})_{\mathbf{M(p)}} := \arccos\left(\frac{\langle \mathbf{v}, \mathbf{w} \rangle_{\mathbf{M(p)}}}{\|\mathbf{v}\|_{\mathbf{M(p)}} \|\mathbf{w}\|_{\mathbf{M(p)}}} \right). \tag{2}$$

We evaluate the angle condition using the inverse metric (Fig. 1) and call the resulting triangulation *anisotropic triangulation*.

4.2.2 Anisotropic Voronoi Diagram

We compute anisotropic Voronoi diagrams on the basis of a well-distributed sample set (Section 5) and a simplified distance measure independently from the triangulation. The most natural generalization of the Voronoi diagram is to use a geodesic distance measure, which, however, is computationally expensive. To ensure a time-efficient solution, we apply a distance measure that has already been successfully used in previous work [3], [5], [17]. Assuming a local metric $\mathbf{M_p}$ and two points $\mathbf{p}$ and $\mathbf{q}$, the simplified distance measure is defined as

$$d_{\mathbf{M_p}}(\mathbf{p}, \mathbf{q}) = \|\mathbf{q} - \mathbf{p}\|_{\mathbf{M_p}} = \sqrt{(\mathbf{q} - \mathbf{p})^T \cdot \mathbf{M_p} \cdot (\mathbf{q} - \mathbf{p})}. \tag{3}$$

This distance measure simulates a piecewise constant metric. Thus, it fits very well to the idea of elliptic glyphs which represent the region of the tensor field that they cover and it is also generalizable to two-manifold domains. See Fig. 2 for a comparison of isotropic and anisotropic Voronoi diagrams. Further details about the computation of Voronoi diagrams are given in Section 6.1.

4.3 Projection Tensor

Some steps of our algorithm require the projection of the metric tensor $\mathbf{M}$ onto the tangent plane of a given surface (Fig. 3). The metric $\widehat{\mathbf{M}}$ projected to the surface defined by the surface normal $\mathbf{n}$ is given by

$$\widehat{\mathbf{M}} = \mathbf{P}(\mathbf{n}) \cdot \mathbf{M} \cdot \mathbf{P}^T(\mathbf{n}). \tag{4}$$

Here, $\mathbf{P}$ is the *projection tensor* given by

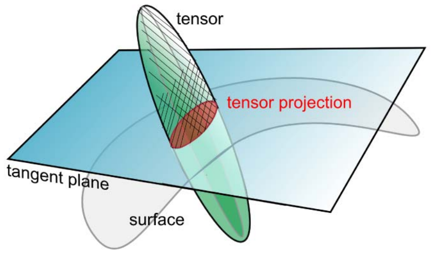

Fig. 3. Three-dimensional tensor projected onto a tangent plane.

$$\mathbf{P}(\mathbf{n}) = \mathbf{I} - (\mathbf{n} \otimes \mathbf{n}) = \begin{pmatrix} (1 - \mathbf{n}_x^2) & -\mathbf{n}_x\mathbf{n}_y & -\mathbf{n}_x\mathbf{n}_z \\ -\mathbf{n}_x\mathbf{n}_y & (1 - \mathbf{n}_y^2) & -\mathbf{n}_y\mathbf{n}_z \\ -\mathbf{n}_x\mathbf{n}_z & -\mathbf{n}_y\mathbf{n}_z & (1 - \mathbf{n}_z^2) \end{pmatrix}, \tag{5}$$

where $\mathbf{I}$ is the identity map. The projection tensor is symmetric, i.e., $\mathbf{P}^T = \mathbf{P}$. It has one eigenvector in the direction of the surface normal $\mathbf{n}$ with eigenvalue zero and two orthogonal eigenvectors, lying in the tangent plane. The eigenvectors of the projected tensor $\widehat{\mathbf{M}}$ are in general not eigenvectors of the original tensor $\mathbf{M}$.

5 Anisotropic Sample Distributions

In the following, we present our algorithm for generating anisotropic sample distributions on planar and two-manifold domains.

5.1 Anisotropic Triangulation

The core of our method builds an anisotropic triangulation as defined in Section 4.2.1.

5.1.1 Properties of the Triangulation

The triangulation, which is generated with respect to the underlying metric tensor field, has the following properties (see Fig. 4):

- Its triangles are stretched according to the metric field and the edges are oriented along the direction of the major eigenvector field [25].
- It naturally has an adaptive character, which supports our requirement of good visual quality.
- It leads to more meaningful neighbor computations in metric space [5] than isotropic triangulations do [2].

5.1.2 Algorithm

For creating the triangulation in the planar domain, we use an incremental algorithm starting with a *supertriangle* that

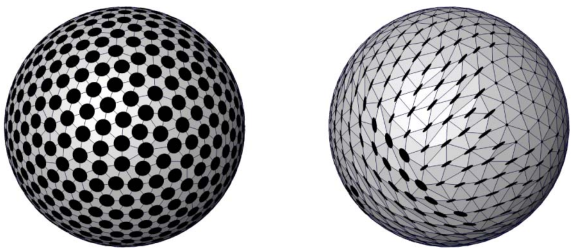

Fig. 4. Delaunay triangulation w.r.t. a uniform isotropic tensor field (left) and anisotropic triangulation w.r.t. a nonuniform anisotropic tensor field (right).

covers the whole sample domain. We successively refine the triangulation by adding new samples in sparsely populated areas (Section 5.2) thereby following the approach by Sloan [26]. Whenever a new sample is added, we check if the angle condition (1) is still fulfilled. Thus, the triangulation is built concurrently with the initial sample distribution.

For creating the triangulation in the two-manifold domain, we propose an edge-flip algorithm on the basis of the triangulated input surface. For computing edge flips in parallel, we distinguish four steps (similar to [24]):

1. Starting from the input triangulation, we check for each edge whether its adjacent triangles fulfill the angle condition (1). If it is not fulfilled, the edge needs to be flipped and, therefore, is labeled $l = 1$. Otherwise and if it is a boundary edge, it is labeled $l = 0$.
2. Next, a subset of edges with label $l = 1$ needs to be found that can be flipped in parallel. An edge can be flipped (in parallel) if the two adjacent triangles do not have another edge that is labeled.
3. Now, the actual edge flip is performed for all labeled edges in parallel.
4. Finally, the triangulation and all neighbor information are updated.

These four steps are repeated until a user-specified number of edges fulfills the angle condition (1) or a maximum number of iterations was reached. In general, this leads to an anisotropic triangulation as defined in Section 4.2.1. Depending on the underlying metric field, however, deadlocks are possible. These occur if flipping one edge results in a configuration, where another edge becomes invalid and vice versa, and it occurs if the two triangles adjacent to the current edge have another edge that needs to be flipped and this configuration cannot be eliminated over several iterations. For the further computations, however, these edges do not lead to problems, because we only need the neighbor relations to compute minimum distance and, therefore, we use second-order neighbors.

5.2 Initial Sampling

The initial sampling procedure computes a set of samples $S := \{\mathbf{p}_i \mid \mathbf{p}_i \in \Omega, i = 1, \dots, n\}$, where n is the number of samples. We require our initial sample distribution to have Poisson-disk characteristics, i.e., the samples should cover the domain densely while a minimum separation between them is maintained. The major bottleneck of methods to generate such distributions, for example, dart throwing, are costly distance computations [3], [5]. In this paper, we present a method that exploits the triangulation's properties (Section 5.1.1) to generate an initial sampling with high visual quality. Thus, we do not require distance computations to identify sample intersections. These are implicitly provided by the triangulation. The properties of the triangulation in combination with the assumption that the metric tensor field does not vary strongly within a small local environment (Section 4.1), and finally the fillrate, make it possible to avoid an explicit sample intersection test.

5.2.1 Triangle Fillrate

For identifying areas where it is beneficial to insert a new sample, we use the fillrate of a triangle as measurement

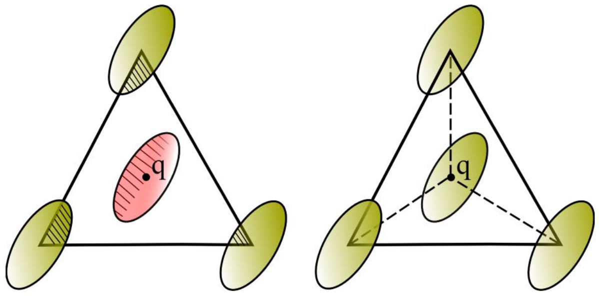

Fig. 5. To compute the fillrate of a triangle, we use the metric at the triangle's barycenter (red ellipse). If the fillrate falls below a user-defined threshold (left), a new sample is inserted at the triangle's barycenter (right) and a retriangulation (dotted lines) is initiated.

(Fig. 5). In [5], the triangle fillrate was used for population control to insert or remove samples during the refinement procedure. We use the triangle fillrate as guidance for the initial sampling procedure. To guarantee interactivity, we propose the following computations.

A triangle's fillrate is the ratio between the subarea $A_\circ$ of the triangle that is covered by the elliptical samples at its corners, and the triangle area $A_\triangle$ itself. To compute $A_\circ$, we use the determinant of the metric at the triangle's barycenter $\mathbf{q} = \frac{1}{3}(\mathbf{p}_1 + \mathbf{p}_2 + \mathbf{p}_3)$, where $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ are the triangle's vertices. As the sum of all triangle angles is always π, the area $A_\circ$ that is covered by an ellipse/ellipsoid is always half of the area of the ellipse/ellipsoid $A_\circ = 0.5 \cdot \pi \cdot \det(\mathbf{M}(\mathbf{q}))$. From $A_\circ$ and $A_\triangle$, the fillrate η is computed as

$$\eta = \frac{A_\circ}{A_\triangle}. \tag{6}$$

We can guide the distribution's density by changing the target value for η. Smaller values result in less dense initial samplings and higher values in denser samplings. To add adaptivity, an additional importance function can be used that influences the scale of the samples and, thus, also the fillrate. For example, an importance function that is guided by the magnitude of the image gradient creates initial sample distributions that are denser at the edges of the input image (see also Fig. 10).

5.2.2 Algorithm

Starting point of initial sampling is a *coarse* anisotropic triangulation of the input domain.

In the planar case, we start by randomly sampling the boundary $\partial\Omega$ of the planar sample domain and adding a few random samples within Ω. 1) For these samples, an anisotropic triangulation is computed that results in a set of triangles $T := \{t_k \mid t_k \in \Omega, k = 1, \dots, m\}$, where m is the number of triangles. 2) For each triangle t_k, its fillrate is determined. If the fillrate is below a user-defined threshold, a new sample is added at the triangle's barycenter. 3) Then, the triangulation is updated and the procedure is started again with step 1. Steps 1-3 are repeated until a desired density, specified by η, has been reached.

In the two-manifold case, we assume a given triangulated surface $\mathcal{X}$. 1) In a first step, $\mathcal{X}$ is simplified using an edge-contraction algorithm [27]. This procedure yields a coarse sample mesh $\widehat{\mathcal{X}}$ and an associated set of triangles

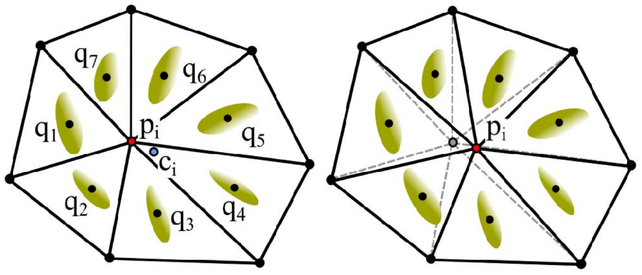

Fig. 6. For gravitational-centered relaxation, the star around each sample is considered. To compute the centroid of this star, the metric tensors (green ellipses) at the triangles' barycenters are used as weights (left). Once the star's centroid is computed, the sample position and its one-ring neighborhood are updated (right).

$T := \{t_k \mid t_k \in \Omega, k = 1, \ldots, m\}$, where m is the number of triangles. Step 2 is similar to the planar case with the difference that new sample positions need to be *projected* onto the original surface, as described in Section 5.4. 3) Next, the triangulation is updated and the procedure is started again with step 2 until a desired sample density is reached.

5.3 Gravitational-Centered Relaxation

The initial sample distribution already fulfills Poisson-disk characteristics. To equalize sample distances, we propose a gravitational-centered relaxation on the basis of the anisotropic triangulation. It equalizes triangle sizes with respect to the metric tensor field so that sample distances become more uniform (e.g., Fig. 10d). This is desirable for the rendering of generalized Voronoi diagrams (Section 6).

5.3.1 Centroid of Tensor-Weighted Star

For each sample $\mathbf{p}_i$, we consider its one-ring neighborhood $t_j \in N(\mathbf{p}_i)$ (Fig. 6). This *star* consists of n triangles that all share the sample $\mathbf{p}_i$ and have a barycenter $\mathbf{q}_j$. The centroid c_i of the star with respect to the metric is defined in analogy to the center of mass, which will be regained when using the euclidean metric:

$$c_i = \mathbf{M}_i^{-1} \cdot \sum_{j=1}^{n} A_{\triangle j} \cdot (\mathbf{M}(\mathbf{q}_j) \cdot \mathbf{p}_i), \tag{7}$$

whereby $\mathbf{M}_i = \sum_{j=1}^{n} A_{\triangle j} \cdot \mathbf{M}(\mathbf{q}_j)$.

5.3.2 Point Relocation

The point relocation comprises an update of the current sample's one-ring neighborhood, i.e., all triangles that share $\mathbf{p}_i$. For planar domains, the sample $\mathbf{p}_i$ is simply translated toward the centroid $\mathbf{c}_i$ with

$$\mathbf{p}_i \leftarrow \mathbf{p}_i + (\mathbf{c}_i - \mathbf{p}_i). \tag{8}$$

For the two-manifold case, centroids that are computed with (8) in general do not lie on $\mathcal{X}$. Therefore, the translated points need to be projected back onto $\mathcal{X}$. The point relocation for surfaces, thus, becomes

$$\begin{aligned} \mathbf{p}_i &\leftarrow P(\mathbf{p}_i + (\mathbf{c}_i - \mathbf{p}_i), \mathbf{n}_i) \\ \mathbf{p}_i &\leftarrow P(\mathbf{p}_i + (\mathbf{c}_i - \mathbf{p}_i), -\mathbf{n}_i), \end{aligned} \tag{9}$$

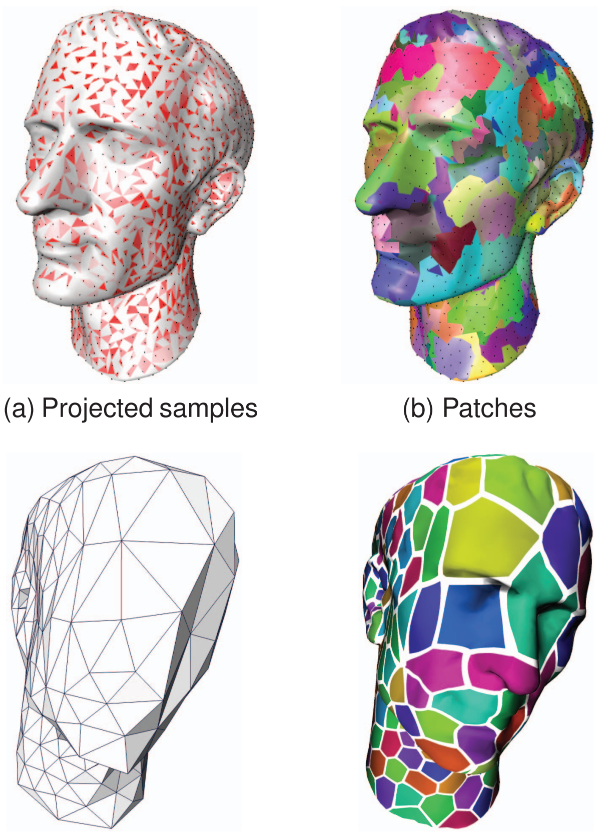

Fig. 7. Patches (b) are generated to keep the correspondence between the samples (black dots in (a)) and the original mesh (a, b, d). Thus, high-quality Voronoi cell rendering is possible (d) even if the samples are distributed sparsely (c).

where P is the projection of the translated point onto $\mathcal{X}$ (Section 5.4), either in the positive normal direction $\mathbf{n}_i$ or in the negative normal direction $-\mathbf{n}_i$.

5.3.3 Algorithm

Gravitational-centered relaxation is based on the initial sample set $S := \{\mathbf{p}_i \mid \mathbf{p}_i \in \Omega, i = 1, \ldots, n\}$, where n is the number of samples:

1. For each sample $\mathbf{p}_i$, its one-ring neighborhood is found.
2. Then, for each one-ring neighborhood, its centroid is computed.
3. Next, the sample $\mathbf{p}_i$ is translated toward this centroid $\mathbf{c}_i$.
4. Finally, in the two-manifold domain, the translated point is further projected back onto $\mathcal{X}$.

The procedure starts again with step 1 until a stable configuration has been found, i.e., when all samples lie in the centroid of their surrounding star, or if a desired number of iterations has been reached.

5.4 Back-Projection

For initial sampling and relaxation in the two-manifold domain, we need to maintain the link between the original mesh and the sample mesh. That is, for each sample, we need to know which triangle of $\mathcal{X}$ it corresponds to (Fig. 7a).

To do this efficiently, we use a *triangle octree* (see, e.g., [28]) to store all triangles of the input mesh $\mathcal{X}$. As projection of a point $\mathbf{p}_i$, we take the closest intersection point of the ray that starts in $\mathbf{p}_i$ and goes into the direction of $\mathbf{n}_i$ and $-\mathbf{n}_i$, respectively, with $\mathcal{X}$. Thereby, the octree data structure helps to quickly identify those triangles that need to be checked for intersection. The intersection test is done in a hierarchical fashion starting with the root node that encloses all triangles and traversing the octree until a node is found that is not subdivided anymore. Then, its elements, i.e., its triangles, are checked for intersection with the ray.

5.5 Volume Slicing

Slicing enables the inspection of 3D input data. These can be volume data but also 2D animated scenes or time-dependent data. To provide a *smooth* transition between the visualizations (Voronoi diagrams or glyphs) of single slices, we adapt our anisotropic sampling for planar domains in the following way: Initial sampling as described in Section 5.2 is done once for the first slice. For this slice, a gravitational-centered relaxation (Section 5.3) is computed until a stable configuration is achieved. For subsequent slices, we use the result of the previous slice as initial sample distribution. Before relaxation, we compute the fillrate for each triangle of the previous result and insert or remove samples if needed. Due to spatial coherence between the slices, these operations are rarely required. Finally, very few relaxation steps (in our examples a maximum of five steps were sufficient) are needed until a stable sample configuration is achieved, which guarantees interactivity while inspecting the 3D volume or depicting two-dimensional animated scenes.

6 Voronoi Cell Rendering

Several visualization techniques benefit from a sample distribution generated with the method presented in Section 5. In visualization, the most common application is to position glyphs according to the sample distribution [1], [2], [3], [5]. Recently, it was shown [5] that anisotropic Voronoi diagrams can be efficiently computed for visualization purposes on the basis of such sample distributions. Through the definition of Voronoi cells (Section 6.1), textures provide many visualization options to encode scalar- and vector-valued features that are contained in tensors.

To guarantee interactivity, which is one of our most important requirements, we present a GPU implementation that computes the Voronoi cells in the fragment shader. In contrast to previous GPU implementations of generalized Voronoi diagrams in the planar domain [29] or centroidal Voronoi diagrams in the two-manifold domain [30], we do not need an explicit representation of the diagram, because we use the diagram for visualization purposes only. The implementation that we present in this paper does not require a surface parameterization.

All visualization algorithms were implemented using the Open Graphics Library (OpenGL) and shader programs of the OpenGL Shading Language (GLSL).

6.1 Generalized Voronoi Cells

Let S be a set of well-defined sample positions. In our case, a generalized Voronoi diagram partitions the domain Ω into n Voronoi regions Ω_i, where each region corresponds to a *Voronoi site* that is centered at $\mathbf{p}_i$. In this work, a site has elliptical or ellipsoidal shape and is described by a metric tensor $\mathbf{M}$ (Section 4.1). A Voronoi region Ω_i of a site centered in $\mathbf{p}_i$ then is defined as the set of all points $P \subset \Omega$ that are at least as close to the site in $\mathbf{p}_i$ than to any other site in $\mathbf{p}_j \in \Omega$ with $j = 1, \ldots, n$ and $i \neq j$:

$$\Omega_i = \{P \in \Omega \mid d_{\mathbf{M}(\mathbf{p}_i)}(\mathbf{p}_i, P) \leq d_{\mathbf{M}(\mathbf{p}_j)}(\mathbf{p}_j, P)\}, \tag{10}$$

with $d_{\mathbf{M}(\mathbf{p})}$ (3) using the 3D coordinates of $\mathbf{p}_i$.

In metric space, Voronoi regions are not bounded by straight lines but by curves. Furthermore, they might be neither convex nor connected. In this case, *orphans* can appear, which are part of a Voronoi region that do not necessarily contain the region's barycenter. For computations that are based on the Voronoi diagram, such orphans have to be avoided. We present a computation based on samples that are equally distributed with respect to an underlying metric tensor field (Section 5). For such well-defined sample distributions, orphans generally do not appear and the Voronoi diagram can be computed via a simplified distance measure. Our method does not guarantee the absence of orphans in areas where the anisotropy of the metric tensor field varies strongly. However, as we use the Voronoi diagram for visualization purposes only, the rare appearance of orphans is negligible. Visual artifacts only occur when the borders of the Voronoi cells are displayed. They are not visible if the cells are texturized. Furthermore, we require that the input tensor fields locally do not have large differences in anisotropy (Section 4.1). For tensor fields that have a high *local* variation of anisotropy, the appearance of orphans would be more likely.

6.2 Preprocessing and Data Structures

This section introduces the data structures and the preprocessing necessary to compute Voronoi cells in the fragment shader.

6.2.1 Enriched Original Surface

To compute Voronoi cells, we require a data structure that provides information about the sample's local neighborhood. Therefore, input of the rendering step is the original mesh $\mathcal{X}$ *enriched* with the information about the 3D sample positions and their corresponding site ids. Since this *meta data* is encoded in the sample mesh $\widehat{\mathcal{X}}$, we *splat* the site ids of $\widehat{\mathcal{X}}$ onto the triangles of $\mathcal{X}$. That is, the id of a sample is projected onto $\mathcal{X}$ and the information is spread into its neighborhood (Fig. 7). To achieve this, we use the following approach.

Initially, all triangles of $\mathcal{X}$ are labeled with $i = -1$. Then, all vertices of $\widehat{\mathcal{X}}$ are inspected and their ids are splatted onto $\mathcal{X}$ via a breadth-first search. During point relocation in the relaxation procedure (Section 5.3), we have already computed the projection of the current site with a triangle of $\mathcal{X}$. Starting from this intersected triangle, we assign each neighboring triangle to the current site if the triangle's id is $i < 0$, i.e., no id was assigned to this triangle so far. If $i > 0$, we compute the euclidean distance in 3D space to the current site and to the previously assigned site. If the distance to the current site is smaller, the triangle's site id is

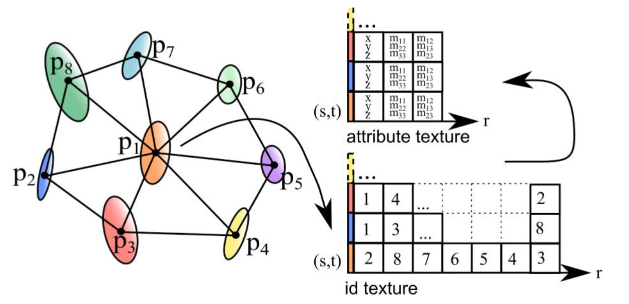

Fig. 8. GPU data structures for the computation of Voronoi cells. The id texture stores the ids of the current sample's one-ring neighborhood. The attribute texture stores the corresponding 3D coordinates and tensors.

updated. The size of the neighborhood that needs to be considered depends on the relation between the number of triangles of the original surface and the number of samples. Each sample of $\mathcal{X}$ needs to know which triangles of $\mathcal{X}$ are in its vicinity (Fig. 7b). The correspondence between $\widehat{\mathcal{X}}$ and $\mathcal{X}$ does not need to be precise, because we use second-order neighbors for the computation of the Voronoi cells. We only need one triangle that is in the vicinity of the current sample. Then, we can determine the correct neighbors using this sample as starting point.

6.2.2 GPU Data Structures

The information we need to compute Voronoi regions in the fragment shader and to evaluate the distance function given in (3) are the metric tensors, the ids and the coordinates of (at least) the current sample's one-ring neighborhood.

The basic idea is to consider the samples $\mathbf{p}_i$ computed with our sampling approach (Section 5) as Voronoi sites around which the regions Ω_i are generated. Each site is described by a *unique id* $i = 1, \ldots, n$, and by its *shape* characterized by its coordinates $\mathbf{p}_i \in \mathbf{R}^3$ and the metric $\mathbf{M}$ in $\mathbf{p}_i$. To upload this information to the GPU, where the rendering is performed, we encode this information into two textures.

Id Texture. The id texture stores the information about the local one-ring neighborhood of a site in the luminance channel of a 3D floating point texture (Fig. 8). To build this texture, we traverse all sites of the sample mesh. The site's id determines the position i, j, where its local neighborhood is stored in z-direction. If n is the number of samples or Voronoi sites, we set the texture size to $\lceil \sqrt{n}\, \rceil \times \lceil \sqrt{n}\, \rceil \times 16$. The depth of 16 is a fixed value and ensures enough memory for the whole one-ring neighborhood. Actually, the texture's depth depends on the degree (or valence) of the vertex v that represents the current Voronoi site. Consequently, a few texels will not contain any information and, thus, are superfluous. They are labeled as invalid, i.e., $i = -1$. When traversing the neighbors of a site, thus, an id of $i = -1$ indicates that all neighbors of the current site have been considered.

Attribute Texture. The attribute texture stores the coordinates and the metric tensor of a site in the *RGB*-channels of a 3D floating point texture (Fig. 8). The first texture layer stores the coordinates, where $R_1 = \mathbf{x}$, $G_1 = \mathbf{y}, B_1 = \mathbf{z}$. The second and third texture layers store the tensor components, where $R_2 = m_{11}, G_2 = m_{22}, B_2 = m_{33}$, and $R_3 = m_{12}, G_3 = m_{13}, B_3 = m_{23}$. We set the size of the metric texture to $\lceil \sqrt{n}\, \rceil \times \lceil \sqrt{n}\, \rceil \times 3$ for 3D metric tensors, and to $\lceil \sqrt{n}\, \rceil \times \lceil \sqrt{n}\, \rceil \times 2$ for 2D metric tensors.

6.3 Voronoi Cell Computation

Given all the data needed (Section 6.2) to draw the Voronoi cells on the original surface, the cells are computed in the vertex and fragment stages. All geometry information (vertex positions, normals, site ids) of the enriched original mesh is uploaded in a vertex buffer object. Surface rendering then is initiated by rendering $\mathcal{X}$.

The vertex stage mainly processes the information of the enriched original mesh. Besides the regular operations that are performed in the vertex stage, the interpolated vertex coordinates and the *flat* site ids are passed to the fragment shader.

The fragment stage then computes the Voronoi regions on a per-fragment basis. In addition to the interpolated vertex coordinates and the site id of the current vertex, the id texture and the attribute texture are provided as input. With this information, we can determine—for the current fragment—which site the interpolated vertex position is nearest to using the distance measure given in (3). Depending on this assignment, the fragment is colored or texturized.

6.4 Texturing of Voronoi Cells

To compute the texture coordinates of the current Voronoi cell, we need the interpolated vertex coordinate $\widehat{\mathbf{p}}_i$, the coordinates of the current site $\mathbf{p}_i$, and the metric tensor at the current site $\mathbf{M}(\mathbf{p}_i)$. The computation of the texture coordinates, thus, is done in the fragment shader.

The coordinates of the current vertex position are transformed into the local coordinate system of the current Voronoi cell Ω_i, i.e., the coordinate system that is distorted by the local metric and with $\mathbf{p}_i$ lying in the center:

$$\mathbf{p}_{\Omega_i} = (\mathbf{M}(\mathbf{p}_i)^{-1} \cdot (\widehat{\mathbf{p}}_i - \mathbf{p}_i)) + \frac{w}{2}. \qquad (11)$$

Here, w is the width of the input texture and $w = h$. Up to this stage, all computations were performed in $\mathbf{R}^3$. To get the 2D texture coordinates $\mathbf{p}_{uv} \in [0, 1]$, $\mathbf{p}_{\Omega_i}$ is projected into the 2D parameter space of the cell and normalized to the $[0, 1]$-range:

$$\mathbf{p}_u = \frac{\langle \mathbf{p}_{\Omega_i}, \mathbf{e}_1 \rangle}{w}; \quad \mathbf{p}_v = \frac{\langle \mathbf{p}_{\Omega_i}, \mathbf{e}_2 \rangle}{w}. \qquad (12)$$

Here, the projection vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ are the major and medium eigenvectors of the projected metric tensor $\widehat{\mathbf{M}}$ (5).

7 Anisotropy Design

Until now, we have assumed that a suitable metric tensor field (Section 4.1) is given. In this section, we explain how metric tensor fields are generated in this work. A tensor

$$\mathbf{M} = \mathbf{U} \cdot \mathbf{V} \cdot \mathbf{U}^T \qquad (13)$$

can be described by its eigenvalues and eigenvectors, where $\mathbf{U}$ is a rotational matrix whose columns are the eigenvectors of $\mathbf{M}$, and $\mathbf{V}$ is a diagonal matrix whose diagonal elements are the eigenvalues $\lambda_1, \lambda_2, \lambda_3$ of $\mathbf{M}$.

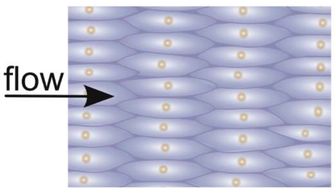

Fig. 9. Schematic depiction of endothelia cells motivated by a depiction in [31].

If the input data are scalar or vector fields, we first create the *parts*, i.e., $\mathbf{U}$ and $\mathbf{V}$, and then compose them again to get $\mathbf{M}$ (13). In more detail, we derive the information about the three eigenvectors and their scaling (represented by the eigenvalues) from the input data. If tensor fields are given as input, we first decompose the tensor into $\mathbf{U}$ and $\mathbf{V}$. Then, we have the possibility to manipulate (map, scale) the eigenvalues to finally compose the parts again to get $\mathbf{M}$ (13). In the following, we give specific examples for the data sets that are presented in Section 8.

Scalar fields. To derive a metric tensor field from an input color image (Figs. 10 and 13), we compute the image's gradient as minor eigenvector. The vector field orthogonal to the gradient field then represents the major eigenvector field aligning with the edges of the input image. Further anisotropy design then mainly subsumes the scaling and/or mapping of the eigenvalues.

Vector fields. An application that directly benefits from our approach is the visualization of endothelia cells of a blood vessel (Figs. 14 and 15). See Fig. 9 for a schematic depiction. Input data are a vector field that represents the *wall shear stress*. In this case, the vector field itself becomes the major eigenvector field and the vector field orthogonal to the input field serves as medium eigenvector field. As this data set is an example for a surface tensor field, the minor eigenvector field is represented by the surface normal, where the corresponding eigenvalue is set to $\lambda_3 = \epsilon > 0$.

Tensor fields. Finally, arbitrary tensor fields are possible input data for our algorithm. In that case, we first decompose the input tensor $\mathbf{T}$ into its eigenvalues and eigenvectors $\mathbf{T} = \mathbf{U} \cdot \mathbf{V} \cdot \mathbf{U}^T$. Then, the eigenvalues can be scaled through eigenvalue mapping [32].

8 Results and Applications

The evaluation of our algorithm is guided by the requirements that we have named in the introduction:

- Time-efficient sample generation.
- Few intuitive parameters.
- Interactive rendering.
- Visualization results of high visual quality.

To demonstrate that the presented algorithms meet these requirements, we have applied the proposed methods to various data sets from different applications. For the planar case, we have used color images as input. For the two-manifold case, a surface vector field was used as input.

In the following, we evaluate the different stages of our algorithm separately. All results presented in this paper were obtained on an Intel Xeon X5550 2.67-GHz system with eight cores and an NVIDIA GeForce GTX680 graphics card.

8.1 Initial Sampling

Planar domain. Fig. 10 shows results of initial sampling in the planar domain. The images vary by the three design parameters for the sampling: global density steered by the fillrate η, an additional spatially varying importance function (Section 5.2) that is guided by the magnitude of the image gradient, and the metric tensor field that was derived from the input image. Depending on the input data and these parameters, our initial sampling strategy generates approximately 10,000 samples per second. On average, the generation of the images depicted in Fig. 10 took about 0.5 seconds. As comparison, anisotropic dart throwing as presented in [4] generates about 200 samples per second and anisotropic dart throwing as presented in [5] generates about 100 samples per second. The images show that the initial sampling strategy that was presented in this paper, efficiently generates sample distributions that avoid holes and clutter. It reaches already a high quality even without the relaxation process and may be sufficient for many applications. The uniformity of the sample distribution depends on the underlying metric tensor field. See also Fig. 11 for an analysis of the quality of the sample distribution.

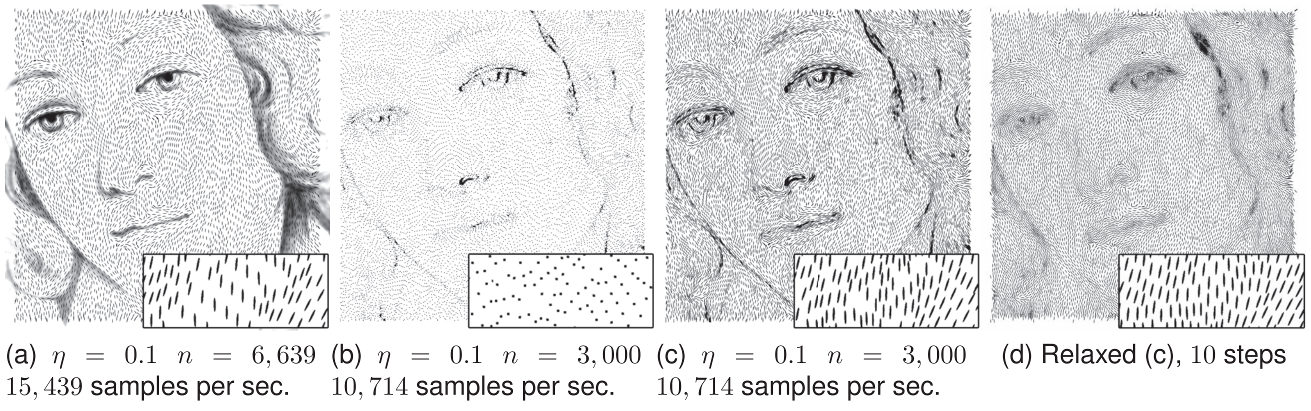
(a) $\eta = 0.1$ $n = 6,639$ $15,439$ samples per sec.
(b) $\eta = 0.1$ $n = 3,000$ $10,714$ samples per sec.
(c) $\eta = 0.1$ $n = 3,000$ $10,714$ samples per sec.
(d) Relaxed (c), 10 steps

Fig. 10. Results for initial sampling (a-c) and relaxation (d). The global density is controlled via the fillrate η. Local density is controlled via an additional importance function based on the image's gradient (c). Anisotropy is steered by the choice of metric: In (b), an isotropic metric was used; the other examples (a, c, d) show an anisotropic metric.

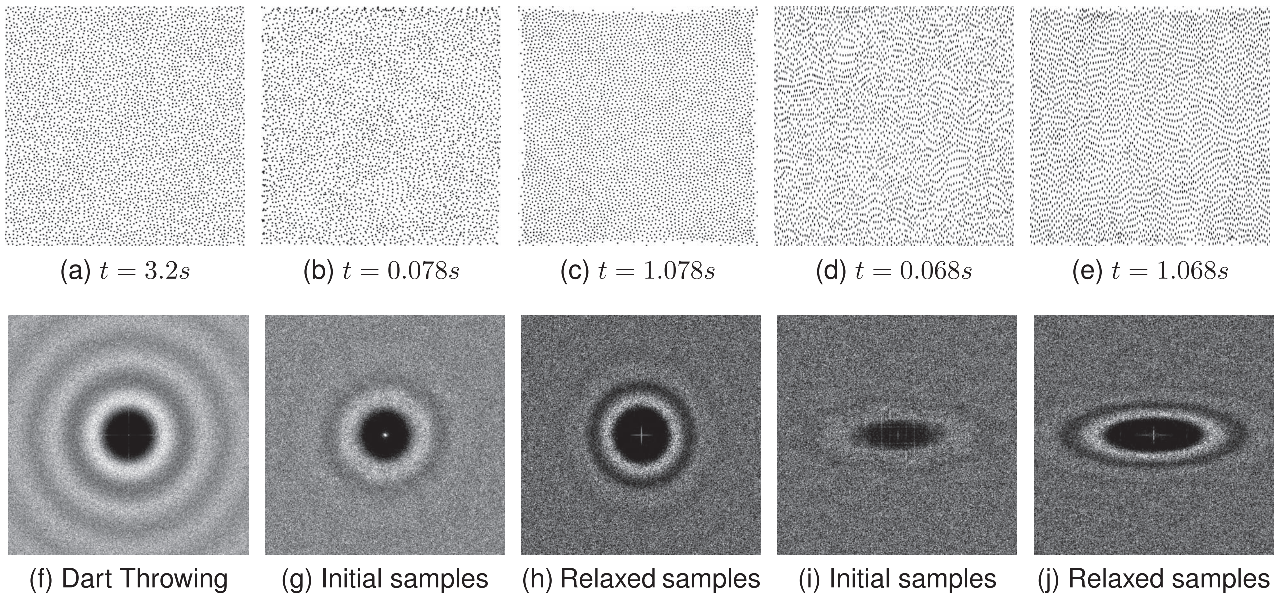

Fig. 11. Isotropic point sets (a-c), anisotropic point sets (d,e) and their corresponding power spectra (f-j). The number of samples in all examples was $n = 4{,}000$. The power spectra all have the characteristics of a blue noise spectrum. That is, a zero region for low frequencies and a relatively constant high-frequency region. Also almost no repetition artifacts and no grid artifacts are visible. Overlaps at the borders arise because of the random initialization and because our method does not compute any sample intersections. For the analysis, we have used the point set analysis tool presented in [33].

Metric	Original Mesh (#vertices)	Initial Mesh (#vertices)	Sample Mesh (#vertices)	Initial (ms)	Relaxation (ms)
Uniform isotropic (a)	6077 (Sphere)	217	430 + 217	80	29
Uniform anisotropic (b)	6077 (Sphere)	217	413 + 217	75	26
Non-uniform anisotropic (c)	6077 (Sphere)	217	11587 + 217	716	38
Uniform isotropic (d)	1422 (Calypso)	400	2744 + 400	70	71
Uniform anisotropic (e)	1422 (Calypso)	400	1591 + 400	51	5
Non-uniform anisotropic (f)	1422 (Calypso)	400	236 + 400	24	13
Non-uniform anisotropic (Figs. 13, 15)	112088 (Aneurysm)	1127	10535 + 1127	1600	900

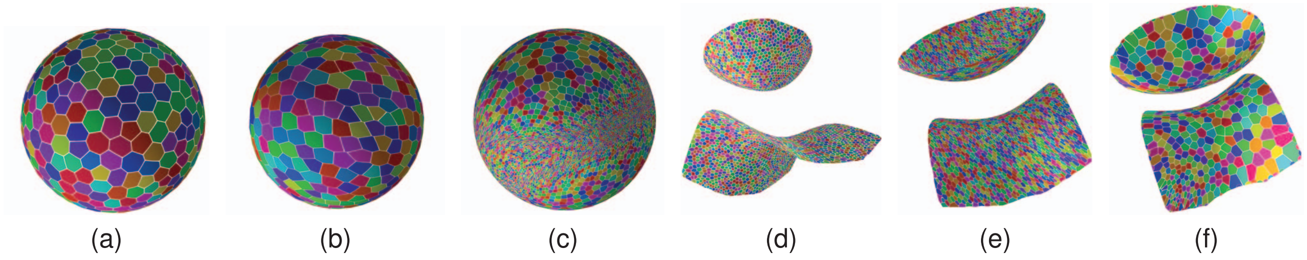

Fig. 12. Timings for the sample generation in the two-manifold domain including the generation of anisotropic triangulations. Once the sample set has been generated, rendering performs at interactive rates (≈ 100 fps). Timings for the relaxation are given for a single iteration. For the examples that are given in this paper, we have computed 100 relaxation steps. A stable sample configuration, however, is generally achieved after 10 to 20 steps.

Two-manifold domain. Fig. 14 depicts results for the sample distribution and the corresponding anisotropic meshes in the two-manifold domain. The metric tensor field is designed on the basis of the wall shear stress of a blood-flow simulation. In this example, the fillrate was set to $\eta = 1.0$. That is, as many samples as possible were distributed on the surface. The resulting anisotropic triangulation represents the anisotropy of the underlying metric. Depending on the complexity of the original surface, our initial sampling strategy generates around 6,000 samples per second in the two-manifold domain. The most time-consuming steps are the projection of new samples onto the original surfaces and the retriangulation of the sample mesh. The table in Fig. 12 provides an overview of times that were needed to generate an initial sample distribution for two analytic examples (sphere and calypso) and the aneurysm data set. It can be observed that the time needed for initial sampling also depends on the size of the original mesh. That is, adding a single sample took about ≈ 0.06 ms for the sphere data set, ≈ 0.03 ms for the calypso mesh and ≈ 0.08 ms for the aneurysm data set. Overall, the time needed for initial sampling depends on many aspects: the coarseness of the initial mesh, the variance of the metric field, the size of the metric tensors, and how many samples can be added in a single step. In this sense, the timings given in Fig. 12 should be treated only as reference.

8.2 Relaxation

Planar domain. In Fig. 10d, a relaxation result after 10 iterations is shown. The relaxed sample distribution is much more uniform. Focusing on the sample images, the initial sampling can be more pleasing to the eye due to the formation of regular patterns during the relaxation process.

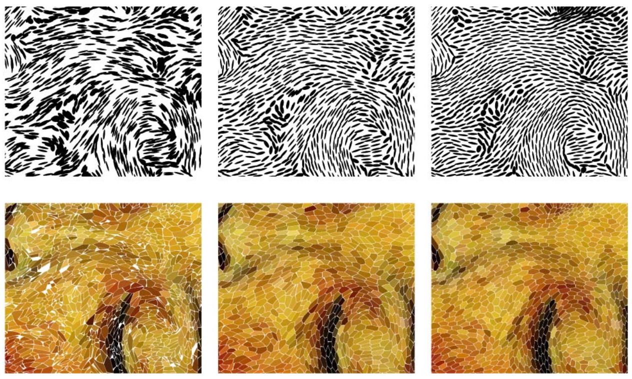
Fig. 13. Top: Sample distributions (random, initial, relaxed). Bottom: Corresponding Voronoi cell rendering. The zooms show that holes and overlaps in the sample distribution result in artifacts and erroneous cells in the Voronoi visualization. For well-defined sample distributions (middle, right), also the anisotropic Voronoi diagram is of high quality.

But the relaxed version results in less artifacts when rendering the Voronoi cells. The effect on the quality of the Voronoi-cell rendering is further demonstrated in Fig. 13. Using a random sample distribution as input, artifacts occur in the Voronoi-based visualization that are caused by holes and clutter in the initial sample distribution. For a good approximation of anisotropic Voronoi diagrams, a well-distributed sample set is required as input. While the initial sample set already produces good results, an even higher visual quality for the Voronoi diagram is achieved with the relaxed sample set. For the generation of Fig. 13, 100 relaxation steps were carried out in 1.5 seconds. That is, a single relaxation step in the planar domain took 15 ms.

Two-manifold domain. Fig. 14 shows an anisotropic mesh and the corresponding glyph distribution for the aneurysm data set. The two closeups show the triangulation before and after relaxation. It can be seen that the triangle sizes equalize well with respect to the underlying metric tensor field. To demonstrate that our method can deal with highly anisotropic and varying metrics, we have exaggerated the anisotropy in this example. Fig. 15 shows an alternative visualization of the aneurysm data set. Here, a texture and a metric tensor field were designed such that the final visualization resembles a schematic depiction of endothelia cells, which align with the blood flow (Fig. 9).

The last column in the table in Fig. 12 provides an overview of times that were needed for a single relaxation step for the examples presented in this paper. The times that are needed for relaxation mainly depend on the size of the sample mesh but also on the quality of the initial sampling, because it influences how many samples are moved in a single step. The major bottleneck here is the back-projection of relocated samples onto the original surface. A single relaxation step for the aneurysm data set takes about 0.9 seconds.

8.3 Anisotropic Voronoi Cell Rendering

Fig. 13 depicts an example application for the Voronoi-based rendering in the planar domain: the generation of mosaic-like images. An example from scientific visualization is the schematic depiction of endothelia cells of a blood vessel (Fig. 15). Since endothelia cells naturally have shapes that equal Voronoi cells, this is an example that directly benefits from our approach.

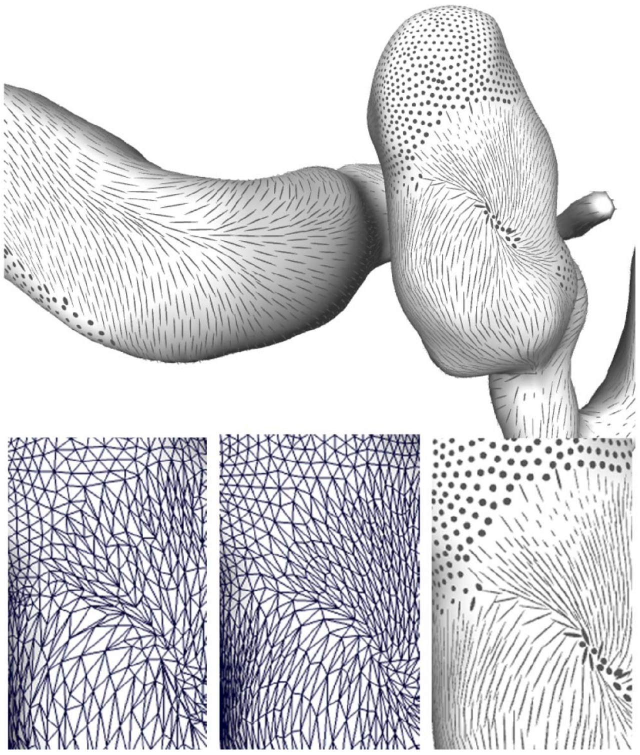
Fig. 14. Results for the generation of anisotropic sample distributions in the two-manifold domain ($\eta = 1.0, n = 10,535, t = 1.6$ s). The zooms show the anisotropic mesh before (left) and after (middle) 100 relaxation steps and the resulting glyph distributions on the basis of the relaxed sample set (right).

9 Discussion and Conclusion

In this paper, we have presented a method for the generation of anisotropic sample distributions in the planar domain, as well as in the two-manifold domain. Moreover, we have presented interactive rendering of anisotropic Voronoi cells.

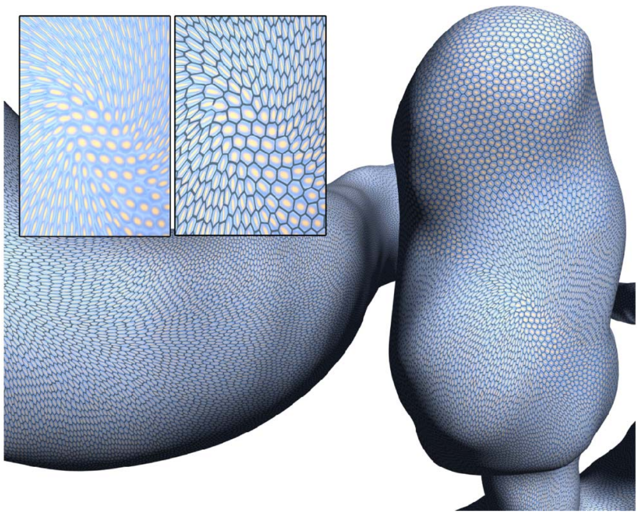
Fig. 15. Visualizations generated via anisotropic Voronoi cell rendering. Voronoi cells were generated on the basis of a surface vector field that represents the wall shear stress. The input texture was designed so that it resembles a schematic depiction of the endothelia cells of a blood vessel. The Voronoi cells align with the blood flow. The zooms show the same region without and with border.

The sampling approach consists of two main steps, the initial sampling and the subsequent relaxation. The goal of initial sampling is to generate sample distributions that cover the underlying domain densely while significant holes and cluttered areas are avoided. We have shown that this can be achieved efficiently through the use of a *density measure* that we call *triangle fillrate* in combination with anisotropic triangulations. This combination of an intuitive measure and a data structure that reflects the underlying data helps to identify areas where it is beneficial to insert new samples. The resulting method can be considered as *guided dart throwing*, where costly intersection as well as conflict checks are not needed. Furthermore, in contrast to previous approaches (e.g., [5]), the only parameter that needs to be specified by a user is the fillrate to control the distribution's density. The fillrate automatically controls the number of samples. If relaxation is required, the user also needs to specify a maximum number of relaxation steps. However, a fixed value of 100 works for most examples and relaxation in general stops earlier. As an additional option, an importance function can be used to generate adaptive sample distributions. Motivated by Lloyd relaxation, which is commonly used in remeshing and sampling approaches, we propose a gravitational-centered relaxation to equalize triangle sizes with respect to the metric tensor field. Gravitational-centered relaxation has the advantage that it is solely triangle-based and does not require an explicit representation of an anisotropic Voronoi diagram. This makes it a stable and time-efficient method to generate more uniform sample distributions as they are needed for the visualization of the anisotropic Voronoi diagram. For applications such as stippling, however, the relaxed sample set might be too uniform. Here, the initial sampling result might be preferable over the relaxed version.

The most time-consuming step during initial sampling and relaxation in the two-manifold domain is the back-projection. For the timings in this paper, the back-projection was done every time a new sample was inserted or every time a sample was moved. Here, a speedup can be achieved if the back-projection is only done once after adding all samples. Then, a breadth-first search starting from the last intersected triangle might be sufficient.

Once the sample set is computed, anisotropic Voronoi-cell rendering is achieved at interactive frame rates. In the current implementation, we have used quadratic textures as GPU data structures, which results in some redundant storage and, thus, a higher memory consumption than actually needed. We plan to improve this through the use of independent texture fetches and index buffer objects. However, for the examples presented in this paper, memory consumption was not an issue. For good visualization results, a lower number of samples are preferable over a higher number, because many samples result in many small Voronoi regions that are difficult to be perceived by a human observer.

The presented approach is very flexible and we believe that it is applicable to many more scenarios than those that were presented in this work. We believe that the texturization of anisotropic Voronoi cells builds the foundation for many new possibilities to visualize tensor fields ranging from the design of suitable textures through texture synthesis. Considering the visualization of endothelia cells, we have focused on their schematic depiction in this paper. For the future, we plan more realistic visualizations which could be achieved, for example, by adding noise to the cell boundaries.

Acknowledgments

The authors thank Leonid Goubergrits and Jens Schaller for helpful discussions and for providing the aneurysm data set.

References

[1] G. Kindlmann and C.-F. Westin, "Diffusion Tensor Visualization with Glyph Packing," *IEEE Trans. Visualization and Computer Graphics,* vol. 12, no. 5, pp. 1329-1336, Sept./Oct. 2006.

[2] M. Hlawitschka, G. Scheuermann, and B. Hamann, "Interactive Glyph Placement for Tensor Fields," *Proc. Third Int'l Conf. Advances Visual Computing (ISVC),* vol. 1, pp. 331-340, 2007.

[3] L. Feng, I. Hotz, B. Hamann, and K. Joy, "Anisotropic Noise Samples," *IEEE Trans. Visualization and Computer Graphics,* vol. 14, no. 2, pp. 342-354, Mar./Apr. 2008.

[4] H. Li, L.-Y. Wei, P.V. Sander, and C.-W. Fu, "Anisotropic Blue Noise Sampling," *ACM Trans. Graphics,* vol. 29, no. 6, pp. 1-12, 2010.

[5] A. Kratz, N. Kettlitz, and I. Hotz, "Particle-Based Anisotropic Sampling for Two-Dimensional Tensor Field Visualization," *Proc. Vision, Modeling, and Visualization (VMV '11),* pp. 145-152, 2011.

[6] P. Alliez, G. Ucelli, C. Gotsman, and M. Attene, "Recent Advances in Remeshing of Surfaces," AIM@SHAPE EU Network, State-of-the-Art Report, 2005.

[7] A. Lagae and P. Dutr, "A Comparison of Methods for Generating Poisson Disk Distributions," *Computer Graphics Forum,* vol. 27, no. 1, pp. 114-129, 2008.

[8] A. Lagae, C.S. Kaplan, C.-W. Fu, V. Ostromoukhov, J. Kopf, and O. Deussen, "Tile-Based Methods for Interactive Applications," *Proc. ACM SIGGRAPH,* Aug. 2008.

[9] A. Lagae, S. Lefebvre, R. Cook, T. DeRose, G. Drettakis, D.S. Ebert, J.P. Lewis, K. Perlin, and M. Zwicker, "State of the Art in Procedural Noise Functions," *Proc. State of the Art Reports (EG '10),* 2010.

[10] R.L. Cook, "Stochastic Sampling in Computer Graphics," *ACM Trans. Graphics,* vol. 5, no. 1, pp. 51-72, 1986.

[11] S.P. Lloyd, "Least Squares Quantization in PCM," *IEEE Trans. Information Theory,* vol. IT-28, no. 2, pp. 129-137, Mar. 1982.

[12] J. Kopf, D. Cohen-Or, O. Deussen, and D. Lischinski, "Recursive Wang Tiles for Real-Time Blue Noise," *ACM Trans. Graphics,* vol. 25, no. 3, pp. 509-518, 2006.

[13] J.A. Quinn, F. Langbein, Y.-K. Lai, and R.R. Martin, "Generalized Anisotropic Stratified Surface Sampling," *IEEE Trans. Visualization and Computer Graphics,* to be Published, 2012.

[14] F.J. Bossen and P.S. Heckbert, "A Pliant Method for Anisotropic Mesh Generation," *Proc. Fifth Int'l Meshing Roundtable,* pp. 63-74, 1996.

[15] K. Shimada, A. Yamada, and T. Itoh, "Anisotropic Triangular Meshing of Parametric Surfaces via Close Packing of Ellipsoidal Bubbles," *Proc. Sixth Int'l Meshing Roundtable,* pp. 375-390. 1996.

[16] X.-Y. Li, S.-H. Teng, and A. ngr, "Biting Ellipses to Generate Anisotropic Mesh," *Proc. Eighth Int'l Meshing Roundtable (IMR '99),* pp. 97-108, 1999.

[17] F. Labelle and J.R. Shewchuk, "Anisotropic Voronoi Diagrams and Guaranteed-Quality Anisotropic Mesh Generation," *Proc. 19th Ann. Symp. Computational Geometry,* pp. 191-200, 2003.

[18] J.-D. Boissonnat, C. Wormser, and M. Yvinec, "Locally Uniform Anisotropic Meshing," *Proc. Ann. Symp. Computational Geometry,* pp. 270-277, 2008.

[19] Q. Du and D. Wang, "Anisotropic Centroidal Voronoi Tessellations and Their Applications," *SIAM J. Science Computing,* vol. 26, no. 3, pp. 737-761, Mar. 2005.

[20] G. Chen, D. Palke, Z. Lin, H. Yeh, P. Vincent, R. Laramee, and E. Zhang, "Asymmetric Tensor Field Visualization for Surfaces," *IEEE Trans. Visualization and Computer Graphics,* vol. 17, no. 12, pp. 1979-1988, Dec. 2011.

[21] M. Goldau, A. Wiebel, N.S. Gorbach, C. Melzer, M. Hlawitschka, G. Scheuermann, and M. Tittgemeyer, "Fiber Stippling: An Illustrative Rendering for Probabilistic Diffusion Tractography," *Proc. IEEE Symp. Biological Data Visualization (BioVis)*, pp. 23-30, 2011.
[22] Q. Du, V. Faber, and M. Gunzburger, "Centroidal Voronoi Tessellations: Applications and Algorithms," *SIAM Rev.*, vol. 41, pp. 637-676, 1999.
[23] L.-Y. Wei and R. Wang, "Differential Domain Analysis for Non-Uniform Sampling," *ACM Trans. Graphics*, vol. 30, no. 4, pp. 1-10, July 2011.
[24] C. Navarro, N. Hitschfeld-Kahler, and E. Scheihing, "A Parallel GPU-Based Algorithm for Delaunay Edge-Flips," *Proc. 27th European Workshop Computational Geometry*, M. Hoffmann, ed., pp. 75-78, 2011.
[25] K. Shimada, A. Yamada, and T. Itoh, "Anisotropic Triangulation of Parametric Surfaces via Close Packing of Ellipsoids," *Int'l J. Computational Geometry Applications*, vol. 10, pp. 417-440, 2000.
[26] S.W. Sloan, "A Fast Algorithm for Constructing Delaunay Triangulations in the Plane," *Advances in Eng. Software*, vol. 9, pp. 35-55, 1987.
[27] M. Garland and P.S. Heckbert, "Surface Simplification Using Quadric Error Metrics," *Proc. ACM Conf. Computer Graphics and Interactive Techniques (SIGGRAPH '97)*, pp. 209-216, 1997.
[28] H. Samet, "An Overview of Quadtrees, Octrees, and Related Hierarchical Data Structures," *Theoretical Foundations of Computer Graphics and CAD*, R.A. Earnshaw, ed., pp. 51-68, Springer, 1988.
[29] Z. Yuan, G. Rong, X. Guo, and W. Wang, "Generalized Voronoi Diagram Computation on GPU," *Proc. Eighth Int'l Symp. Voronoi Diagrams in Science*, pp. 75-82, 2011.
[30] G. Rong, Y. Liu, W. Wang, X. Yin, D. Gu, and X. Guo, "GPU-Assisted Computation of Centroidal Voronoi Tessellation," *IEEE Trans. Visualization and Computer Graphics*, vol. 17, no. 3, pp. 345-356, May 2011.
[31] A.M. Malek, S.L. Alper, and S. Izumo, "Hemodynamic Shear Stress and Its Role in Atherosclerosis," *J. Am. Medical Assoc.*, vol. 282, no. 21, pp. 2035-2042, 1999.
[32] I. Hotz, L. Feng, H. Hagen, B. Hamann, K. Joy, and B. Jeremic, "Physically-Based Methods for Tensor Field Visualization," *Proc. IEEE Conf. Visualization (Vis '04)*, pp. 123-130, 2004.
[33] T. Schlömer and O. Deussen, "Accurate Spectral Analysis of Two-Dimensional Point Sets," *J. Graphics, GPU, and Game Tools*, vol. 15, no. 3, pp. 152-160, 2011.

Andrea Kratz received the Diploma in computer science from the University of Koblenz-Landau. She is working toward the PhD degree in the visualization group of the Zuse Institute Berlin. For her diploma thesis, she was a student assistant at the VRVis research center in Vienna, Austria. Her research interests include tensor visualization, GPU programming, and sampling methods.

Daniel Baum studied computer science from Humboldt University Berlin, Germany, and the University of Edinburgh, Scotland, and received the MS degree from Humboldt University and shortly after joined the Zuse Institute Berlin. During the PhD degree, which he received from Freie Universität Berlin, Germany, he worked in the field of molecular visualization and similarity. His current interests include the analysis of image as well as molecular data.

Ingrid Hotz received the MS degree in theoretical physics from the Ludwig Maximilian University in Munich Germany and the PhD degree from the Computer Science Department, University of Kaiserslautern, Germany. During 2003 and 2006, she was a postdoctoral researcher at the Institute for Data Analysis and Visualization (IDAV) at the University of California. Currently, she is a leader of a junior research group at the Zuse Institute in Berlin Germany. Her research interests are in the area of data analysis and scientific visualization with focus on tensor and vector fields.