Stripe Parameterization of Tubular Surfaces

Felix Kälberer, Matthias Nieser, and Konrad Polthier

Freie Universität Berlin

Abstract. We present a novel algorithm for automatic parameterization of tube-like surfaces of arbitrary genus such as the surfaces of knots, trees, blood vessels, neurons, or any tubular graph with a globally consistent stripe texture. Mathematically these surfaces can be described as thickened graphs, and the calculated parameterization stripe will follow either around the tube, along the underlying graph, a spiraling combination of both, or obey an arbitrary texture map whose charts have a 180 degree symmetry.

We use the principal curvature frame field of the underlying tube-like surface to guide the creation of a global, topologically consistent stripe parameterization of the surface. Our algorithm extends the QuadCover algorithm and is based, first, on the use of so-called projective vector fields instead of frame fields, and second, on different types of branch points. That does not only simplify the mathematical theory, but also reduces computation time by the decomposition of the underlying stiffness matrices.

1 Introduction

Tubular surfaces appear in many application areas such as networks of blood vessels and neurons in medicine, or tube and hose systems in industrial environments. Often a tubular structure must be recovered and segmented from noisy scan data. Our *stripe parameterizer* is an efficient and automatic method for the parameterization and remeshing of free-form tubular surfaces given as triangle meshes. Our special focus lies on free-form surfaces which are not made out of regular, cylindrical tube pieces - those can be handled better by other algorithms from CAD. An additional benefit of the stripe parameterization is the enhanced visualization of the underlying geometric structure.

1.1 Previous work

Surface parameterization is an active research area. We will not attempt a complete review of the literature but instead refer the reader to the surveys by Floater and Hormann [1] and [2].

A very early surface parameterization method is the Tutte's [3] barycentric graph embedding. Tutte embeddings are of combinatorial structure and do not capture the geometry of the surface. Early global parameterization methods were introduced by Haker, Gu and Yau [4–6] who studied conformal parameterizations. Conformal maps are angle preserving at the cost of possibly large length distortions, as angles and lengths can not be preserved at the same time.

To reduce length distortion, Kharevych et al. [7] used cone singularities for conformal parameterization, which relax the constraint of a flat parameter domain at few isolated points. Such singularities have proven to be essential for high quality parameterizations and have been used in other parameterization schemes as well.

Tong et al. [8] use singularities at the vertices of a hand-picked quadrilateral meta layout on the surface. The patches the meta layout are then parameterized by solving for a global harmonic one form. Dong et al. [9] use a similar idea for parmeterization but create the quadrilateral meta layout automatically from the Morse-Smale complex of eigenfunctions of the mesh Laplacian.

Ray et al. [10] parameterize surfaces of arbitrary genus with periodic potential functions guided by two orthogonal input vector fields. This leads to a continuous parameterization except in the vicinity of singular points on the surface. These singular regions are detected and reparameterized afterwards. With the QuadCover algorithm [11] we built upon their idea to use an input field as guiding directions for parameter lines. Input fields can be principal curvature directions, for example, or user-designed fields using one of the recent tools for the design of rotational symmetry fields like [12], [13], [14], or [15]. The idea of QuadCover is to find a parameterization whose gradient matches the input directions as well as possible.

The literature on parameterization of tubular objects is by far not as extensive as for general surfaces. Huysmans et al. [16] construct a progressive mesh which they map to an open cylinder. A subsequent iterative scheme optimizes the vertex positions in the cylindrical domain. Unfortunately, that method can not handle bifurcations. Antiga and Steinman [17] handle blood vessels with bifurcations by splitting the vessel tubes at their branches, and parameterize each segment separately which leaves visible artifacts at the joints of the segments. Zhu et al. [18] use conformal parameterizations on tubular objects. Since conformal maps do not allow precise control over the direction of parameter lines, they cannot be aligned with the tube axis.

1.2 Contributions

We introduce the stripe parameterizer, an algorithm for the generation of globally consistent stripe parameterizations, see Fig. 1 and 9. Each parameterization is a collection of texture maps which may also be used to remesh and segment a surface. The stripe parameterizer is a generalization of QuadCover, which parameterizes general surface meshes. The stripe parameterizer allows to map stripe patterns onto a surface, *i. e.*, texture maps whose individual charts are symmetric with respect to rotations of 180 degrees. In contrast to QuadCover where all texture image charts have to be symmetric with respect to 90 degree rotations, the stripe parameterizer allows a more general set of texture images with only 180 degree symmetry.

We develop the mathematical theory for stripe parameterizations and discuss the differences to grid parameterization techniques including those in Quad-Cover. Stripe parameterizations allows only a subset of the branch point types

of QuadCover. For example, cone points of index 1/4 at the corners of a cube can not be used in stripe parameterizations since 90 degree rotational symmetric textures charts would be required, see the cube in Fig. 2.

Only one type of branch points can occur on a 2-sheeted covering, so there is no need to handle different branch types. The 4-sheeted branched covering surface from QuadCover projects onto a unique 2-sheeted branched covering surface for the stripe parameterizer. Furthermore, the stiffness matrix from QuadCover decomposes into two matrices of quarter size. Thus, the numerical effort of computing a stripe parameterization is seriously reduced.

We tested the stripe parameterizer on several test models and real world examples including clinical data and various tree-like surfaces.
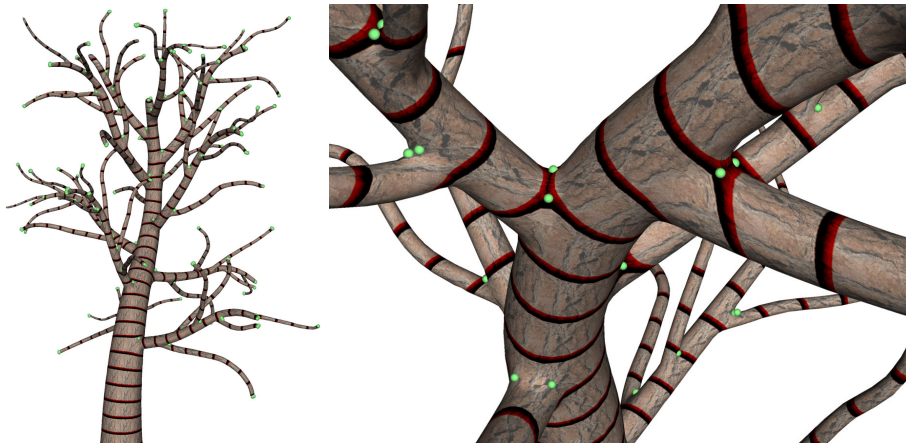

Fig. 1. Tree with stripe parameterization. Singularities are marked in green. The texture image consists of a vertical stripe visualizing the u-isolines of the parameterization.

2 Overview

A stripe parameterization is a special case of a (u, v)-parameterization, where the parameter lines can be globally separated into u-lines and v-lines, as in Fig. 1. This separation property is not present on general surfaces, if singularities of quarter index are present.

Stripe parameterizations can be used for mapping texture images which are symmetric by rotations of 180 but not necessarily 90 degrees, such as stripe textures. An example of a parameterization which is not suitable for mapping stripe textures is shown in Fig. 2.

Projective fields. The parameterization is guided by a so-called *projective field*, which is a vector field on M, where the vectors v and $-v$ are identified for all $v \in T_p M$, $p \in M$. Thus, the vectors may change their sign without producing
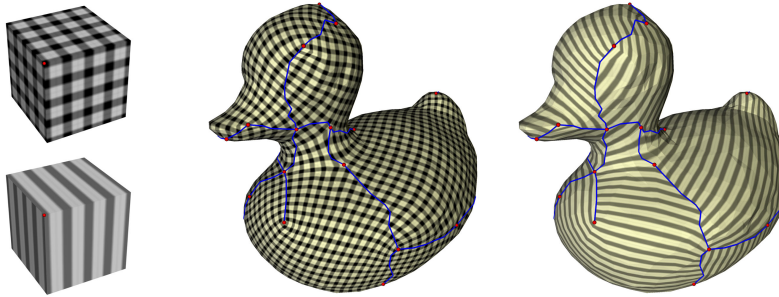
Fig. 2. QuadCover parameterization with quad texture and stripe texture. Stripe textures can not simply be used in parameterizations from QuadCover.

a discontinuous projective field. Note, that projective fields are a special case of N-RoSy fields for $N = 2$ as introduced by Palacios and Zhang [13].

The algorithm takes two projective fields as input and generates two scalar functions (u and v), whose gradients match up with the input fields as well as possible. The coordinates u and v can be used as texture coordinates in order to map a pattern onto the surface. If you are only interested in a stripe pattern, you could use only one input field and skip the computation of v.

Construct an input field. It is up to the user to construct an input field. A canonical choice is the field of minimum curvature: In each point, the corresponding vector points in direction of the (absolute) smaller principle curvature and has unit length. Using this field (together with the 90 degrees rotated field) as the input fields yields nearly curvature line parameterizations.

The algorithm. Starting from a given projective field, the algorithm first constructs a locally integrable field. Second, the surface is cut open to a topological disk and this field is integrated yielding a parameterization. Third, the parameterization is adapted such that the grid lines are connected across the cuts. Details are given in Sect. 4.

Special issues arise when the projective field has singularities. They are resolved by using branched covering spaces. The projective field naturally simplifies to a single vector field on the covering, and then standard Hodge decomposition techniques are used to assure global integrability. Details are explained in Sect. 3.2.

3   Mathematical Setting

We use the theory of QuadCover's 4-fold symmetric fields and apply it to the projective vector field setting with 2-way symmetry properties. We introduce the notion of projective vector fields and discuss consequences for the branched 2-fold covering spaces. We will describe our concepts in the smooth cast first, followed by the discretization for triangle meshes.

3.1 Parameterizations and Matchings

Smooth case. Given a smooth 2-manifold M with charts $U_i \subset M$, $\sum_i U_i = M$. A parameterization is a collection of diffeomorphisms $f_i = (u_i, v_i)$ that map all charts into the parameter domains $f_i : U_i \to \Omega_i \subset \mathbb{R}^2$. One can now visualize the parameter lines on M as the preimage under f_i of the unit grid $\mathbb{N} \times \mathbb{R}$ (u_i lines) and $\mathbb{R} \times \mathbb{N}$ (v_i lines).

A **globally continuous stripe parameterization** consists of parameter functions f_i in the charts U_i, such that the u_i lines (resp. v_i lines) coincide in all regions where two charts U_i, U_j overlap. Thus, the parameter lines of a stripe parameterization can be globally separated into u-lines and v-lines.

Given two guiding fields on the surface, we will only focus on computing the u-component from the first input field, as the same rules apply for computing v from the second field.

The transition functions between adjacent charts of a stripe parameterization satisfy two conditions: First, the gradients of u_i and u_j have to agree up to sign, because we do not distinguish u_i lines and $-u_i$ lines on the parameterized surface. Thus, the gradients of the charts are related by

$$\nabla u_i(p) = (-1)^{r_{ij}} \nabla u_j(p), \quad p \in U_i \cap U_j \tag{1}$$

with a constant number $r_{ij} \in \{0, 1\}$ on the intersection $\Omega_i \cap \Omega_j$. We call the values r_{ij} *matchings* between charts U_i and U_j.

Second, the values of u_i and u_j may differ only by integer values, since the u lines in the unit grid are invariant under translations by integer values.

Thus, we require the values of u in overlapping charts U_i and U_j to fulfill:

$$u_j(p) = (-1)^{r_{ij}} u_i(p) + t_{ij} \;, \quad r_{ij} \in \{0, 1\}, \;\; t_{ij} \in \mathbb{N}, \;\; p \in U_i \cap U_j \;. \tag{2}$$

Discretization. Each triangle of the mesh is considered as a chart. The transition function between two adjacent triangles is fully determined by the matching and the translation vector associated to their common edge, see (2). See Sect. 4.1 for details on how we compute the matching.

3.2 Projective Fields

A parameter function u can be characterized by its gradient field. In each chart, the gradient field ∇u is a continuous vector field. At the transition between two charts U_i, U_j, the sign of the vectors may flip depending on the matching. Thus, we cannot describe the derivatives of u as a globally defined vector field, but use *projective fields* which are invariant under sign flips.

Definition 1. *Given a manifold M with charts U_i and matchings r_{ij}. A **projective field** K on M is a collection of one vector field K_i in each chart U_i, such that for all overlapping charts $U_i \cap U_j \neq \emptyset$:*

$$K_j = (-1)^{r_{ij}} K_i. \tag{3}$$

Discretization. The projective fields are piecewise constant on the triangles. Store one vector per triangle and the matching number on each edge. This fully defines a discrete projective field. An odd matching at any edge means that the vector in one adjacent triangle corresponds to the negated vector of the other triangle.

3.3 Branched Covering Spaces

We use the notion of branched covering surfaces for an equivalent description of projective fields. A projective field on the input surface can be regarded as a vector field on a covering surface. This allows us to apply standard vector field calculus to projective fields.

Coverings. First, recall some definitions on Riemann surfaces, see [19–21]. We will give an abstract definition of a covering and explain below how we actually construct one.
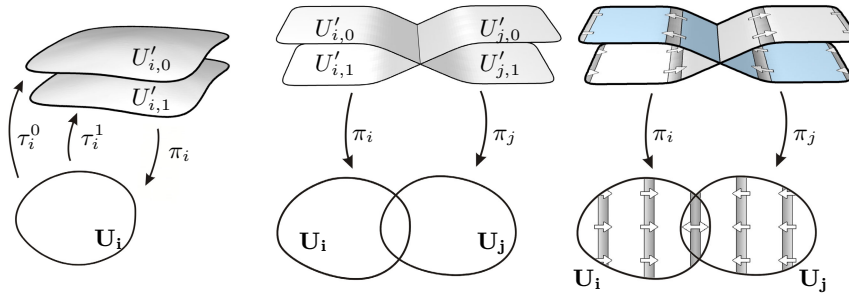

Fig. 3. From left to right: Trivial covering. / Patching two coverings together with matching $r_{ij} = 1$. / A projective field lifted to a vector field on the covering.

Definition 2. *Let M be a Riemann surface. A **2-sheeted covering** M' of M is a Riemann surface with a local homeomorphism $\pi : M' \to M$ with the property: For each point $p \in M$, there exists a neighborhood U_p whose preimage $\pi^{-1}(U_p)$ is the union of exactly two pairwise disjoint topological disks. Fig. 3, left shows a 2-sheeted covering.*

*We allow **branch points** p in our setting, where the preimage of a neighborhood of p is exactly one topological disk (instead of two), cp. Fig. 4, middle.*

Construction. We construct a covering of M as follows: From each chart U_i, make two copies (*layers*) and name them $U'_{i,0}$ and $U'_{i,1}$. Let $\pi_i : U'_{i,0} \cup U'_{i,1} \to U_i$ be the operator which projects the copies back to U_i and $\tau_{i,0} : U_i \to U'_{i,0}$, $\tau_{i,1} : U_i \to U'_{i,1}$ the inverse maps. The two layers $U'_{i,0} \cup U'_{i,1}$ together with π_i is called the *2-sheeted trivial covering* of the chart U_i (Fig. 3, left).

In the next step, we glue these layers at the overlaps of the adjacent charts together. For each pair of charts the layers can be glued in two different ways. The matchings r_{ij} define how the layers are identified.
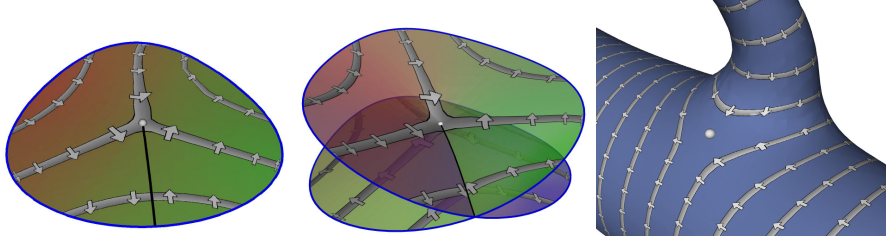
Fig. 4. Left: Stripe parameterization with branch point. The isolines of the u function and its gradient vectors are drawn. Middle: The same parameter function on the 2-sheeted covering (the covering surface is not embedded, it has self-intersections). Right: Branch point on a parameterized tube object.

Definition 3. *A **covering surface induced by matchings** r_{ij} is uniquely defined by the following construction:*

Let (U_i', π_i) be 2-sheeted trivial coverings of the charts U_i. The covering surface is given as the union of all U_i' where the following points are identified: In each two overlapping charts U_i, U_j, identify all points $\tau_{i,0}(p)$ with $\tau_{j,r_{ij}}(p)$ and $\tau_{i,1}(p)$ with $\tau_{j,1-r_{ij}}(p)$, $p \in U_i \cap U_j$ (see Fig. 3, middle).

Since the trivial coverings of charts have no branch points and the charts cover the surface, we cannot construct any branch points this way. We allow branch points by removing single points from the surface. Depending on the matchings we obtain a branch point there as shown in Fig. 3.

Discretization. In the discrete setting, branch points are located at vertices. On a 2-sheeted covering they occur when the sum of all matchings of incident edges is odd. This means starting somewhere in the neighborhood of v and walking once around the vertex ends on a different layer in the covering than the start point.

3.4 Vector Fields on Covering Spaces

Projective fields can be described as vector fields on a covering surface. This result allows us to apply the classical vector field theory to projective fields.

A projective field K on M with matchings r_{ij} canonically lifts to a vector field K' on the covering induced by r_{ij}. In each chart U_i, define the vectors on its trivial covering as follows: For all $p \in U_{i,0}'$ set $K'(p) := K_i(\pi_i(p))$ and for $p \in U_{i,1}'$ set $K(p) := -K_i(\pi_i(p))$, see Fig. 3, third image.

The result is a globally well defined vector field K' on M', since the layers of the covering are connected in the same way as the vector fields permute when another chart is chosen.

Definition 4. *Let M be a manifold with matchings r_{ij} and M' the induced covering. A projective field lifted to a vector field K on M' is called a **covering field** of M.*

4 Stripe Parameterizer Algorithm

In this section we describe the main extensions and simplifications which have been made to QuadCover to yield the stripe parameterizer. An important difference is the use of projective vector fields instead of frame fields.

Compute the potential function. Given a surface M together with a projective field K, or, equivalently, a covering surface M' with a vector field K'. The parameter function is a scalar function $u' : M' \rightarrow \mathbb{R}$. It can be projected back to a parameter function $u : M \rightarrow \mathbb{R}$ by taking the values of u' in one of the two layers (it does not matter which layer is taken, because the parameter lines in both layers will be congruent).

The parameterization algorithm is divided into two stages: Assuring local continuity and global continuity. The two stages are explained in Sect. 4.3 and 4.4. Sect. 4.1 deals on the creation of an input field. For the integration of a projective field, we need to cut the surface open at a given cut graph. Sect. 4.2 explains the construction of such a cut graph.

4.1 Generate Input Field and Matching

Curvature field. In our experiments, we used the field of minimum principle curvatures as input to the parameterizer. Discrete principal curvature directions and values can be calculated as proposed in [22] or [23]. Note that we deal with curvatures given on triangles, not on vertices.

Finally, one gets a unit vector v in each triangle pointing in along that principle curvature direction which corresponds to the (absolute) smaller curvature value. In triangle t, set $K_0(t) := v$ and $K_1(t) = -v$.

We define matchings r_{ij} between every two adjacent triangles t_i and t_j by setting $r_{ij} = 0$ if $\langle K_0(t_i), K_0(t_j) \rangle \geq 0$ and $r_{ij} = 1$ otherwise. This ensures that the field does not turn around, but proceeds as straight as possible.

Note that the position of branch points immediately follows from the choice of matchings and the matchings are determined by the input field. A branch point arises at each vertex where the sum of matching of outgoing edges is odd.

4.2 Generating Cut Paths

A cut graph is a graph G embedded in the surface, such that $M \setminus G$ is a topological disk. We use a cut graph for the integration of projective fields in Sect. 4.3, and use cut paths for the global continuity in Sect. 4.4.

Cut paths on M. Loosely speaking, cut paths are a set of paths on the surface whose union is a cut graph. On closed surfaces, generating loops of the first fundamental group are suitable cut paths. In QuadCover, we use certain generators, namely the shortest system of loops as computed in Erickson and Whittlesey [24]. A system of loops is a set of $2g$ simple loops with a common base point, whose union is a cut graph.

We can treat branch points as tiny holes, as if they were removed from the surface (see Sect. 3.3). Erickson and Whittlesey handled closed surfaces

only, but we might have a surface with boundary. Once we have more than one boundary component, each additional boundary component needs one path. Thus, in presence of $b > 1$ boundary components (or branch points) we need $2g + b - 1$ paths.

In our implementation for triangle meshes we identify all boundary vertices and branch points into one point B. On this surface (now without any boundary), we apply the method of [24] with B as the base point. When we undo the identification of boundary points, the paths which looped through B now turn into paths that connect boundary components and branch points.

Cut paths on covering. For our algorithm, we need cut paths on the covering surface M'. We get the cut paths by computing them on M and lifting them to the covering. The resulting paths cut M' into two separate simply connected pieces. Thus, one of the paths could theoretically be discarded. It does not matter for our method that the covering decomposes into two pieces. It is more important, that the cut paths are symmetric with respect to a change of layers, *i. e.*, for each path there is another path which runs in the other layer and has the same projection onto M, see Sect. 4.4.
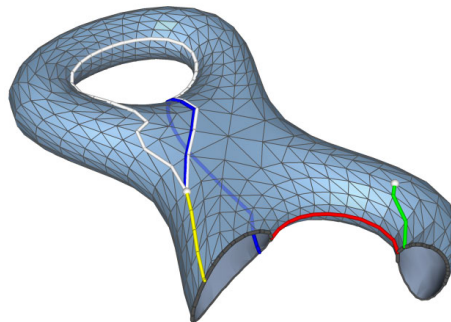
Fig. 5. Surface with boundary and two branch points. The colored lines visualize the five cut paths.

4.3 Local Continuity

The gradient of the parameterization should align with the given input field as well as possible, *i. e.*, u' should minimize the energy

$$E(u') = \int\limits_{M'} \|\nabla u' - K'\|^2 dA. \tag{4}$$

Recall, that the vectors of K' are identical up to a different sign in the two layers. Since the energy has a unique minimum and due to the symmetric shape M', the solution u' is also a map with negated function values in different layers.

The optimization problem (4) can be solved using the Hodge-Helmholtz decomposition. It states, that any vector field K' has a unique decomposition

$$K' = P + C + H \tag{5}$$

with a gradient field P, a cogradient field C and a harmonic field H. P and H are curl free (locally integrable), whereas C contains the curl part. Furthermore, the three spaces of potential fields, copotential fields and harmonic vector fields are perpendicular in L^2 norm. Thus, discarding the second term leads to a curl

free field $\tilde{X}' := P + H$ whose integral is the minimizer of Energy (4). The middle term C of the Hodge-Helmholtz decomposition is a non-conforming function and is found by solving a linear system of equations with one variable per edge. For details on the Hodge decomposition and integration of discrete vector fields see [25].

So far, the parameterization algorithm outlines as follows:

1. Perform Hodge-Helmholtz decomposition of input field K'.
2. Discard the non-integrable curl part and obtain a locally integrable field.
3. Cut the surface open to be simply connected and lift the cut graph to the covering, such that the covering is cut into two connected pieces.
4. Obtain the parameterization u' by integration: Perform a linear run over all vertices (using a growing disk which does not cross the cut graph) and compute the parameter values at each vertex such that the gradient matches up with the vector field.

4.4 Global Continuity

The parameter lines of the solution u' from the previous paragraph are not necessarily continuous everywhere. They may have a mismatch at the cut graph G, see Fig. 6.

Let γ_i be a set of cut paths. For each path γ_i and each point $p \in \gamma_i$, one can measure the **gap** d_i (discontinuous jump) as the difference of function values on the right and left side of the path.

The parameterization can now be "repaired" such that the parameter lines match up. This is exactly the case if all gaps are integer values. The repairing algorithm is based on the following observation: along each path γ_i, the gap is always a constant d_i, since the derivative of the function is locally integrable. Note, that there is an exception if two paths γ_i, γ_j merge and run on top of each other. In this case, the gap turns into $d_i + d_j$. For further details, see [11].

Fig. 6. Parameterization after the first stage. Grid lines are discontinuous across the cuts.

Thus, the grid lines are globally continuous if and only if all $d_i \in \mathbb{Z}$. In order to adapt the function to fulfill the global continuity condition, we add a scalar function ψ to u' such that $\tilde{u}' := u' + \psi$ satisfies $\tilde{d}'_j \in \mathbb{Z}$ (where $\tilde{d}'_j$ are the gaps of $\tilde{u}'$).

The remaining problem is to find this scalar function with given gaps. In order to minimally distort the initial parameterization, we let ψ be a harmonic function, as they are the smallest functions with given gaps in L^2 norm. ψ is found via minimizing the Dirichlet energy $E_D = \int_M \|\nabla \psi\|^2 dA$ under the constraint of given gaps.

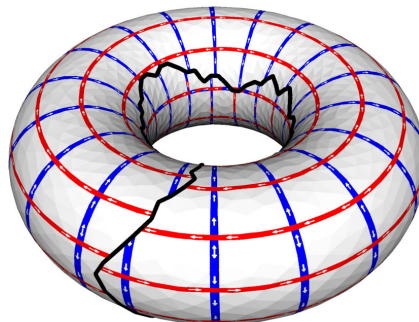The second stage of stripe parameterizer has the following outline:

1. Compute cut paths γ_i.
2. Measure gaps d_i.
3. Find harmonic map ψ with gaps $\mathrm{round}(d_i) - d_i$.
4. Add ψ to u'.

In step 3, the gaps are rounded to the closest integer. Rounding the gaps such that the distortion is minimized is an NP hard combinatorial problem. As we do not solve this problem exactly, the rounding behavior slightly depends on the choice of cut paths.

5   Results

We have tested our method on different tube-like surfaces. Simple examples are the knots in Fig. 7 without branchings. The principal curvature directions are stable and can be computed very accurate in each point, so the algorithm produces a parameterization of high quality.
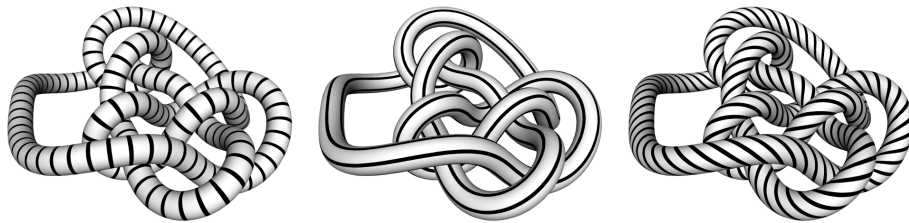
Fig. 7. Left: Only u coordinates are used to map stripes on the knot surface. Middle: Only v coordinates are used. Right: u and v coordinates are used. The texture image is a diagonal line which connects two opposite corners.
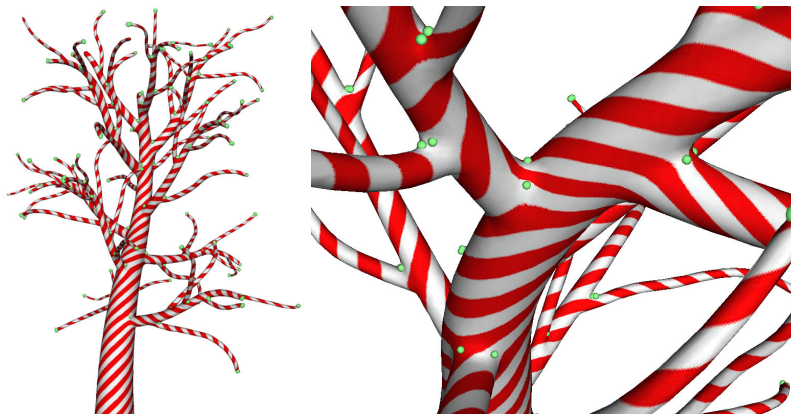

Fig. 8. The tree model of Fig. 1 with diagonal stripe pattern generates a candy cane. Singularities are marked in green.

The tree in Fig. 8 has a more complicated shape. It bifurcates and the thickness of the twigs vary. Note the accurate placement of branch points. There are two branch points at each bifurcation, allowing the circular stripes to split.

Fig. 10 shows a complex neuron model of genus 23, captured using confocal microscopy. The produced parameterization has very little distortion even on this complicated object.

The unshaded version in the top demonstrates how a stripe pattern helps to perceive the complicated shape of the neuron. But also in the fully shaded images, the stripes help to capture the object more easily.

The surface in Fig. 9 shows a human blood vessel which contains parts with a very large tube radius as well as very filigrane branches. Regardless of this difference in the scaling, the stripe density stays nearly constant everywhere.
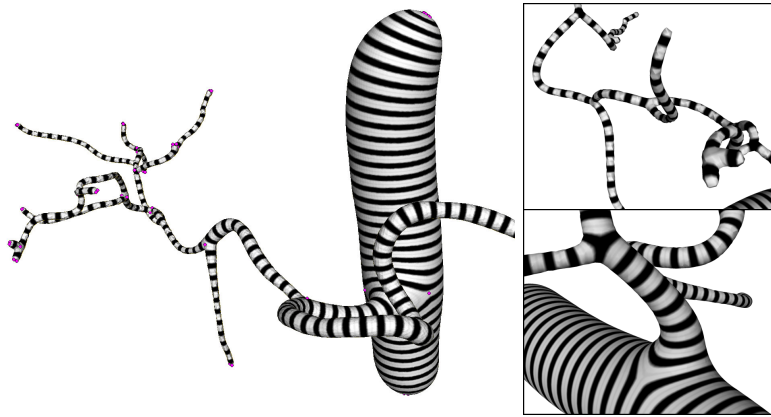

Fig. 9. Parameterized blood vessel, captured by MRT. Courtesy of Fraunhofer MEVIS.

The parameterization of these models was fully automatic. We only chose the density of the lines and the amount of curvature field smoothing. The models had approximately 20k to 40k triangles and the algorithm terminated in less than a minute.
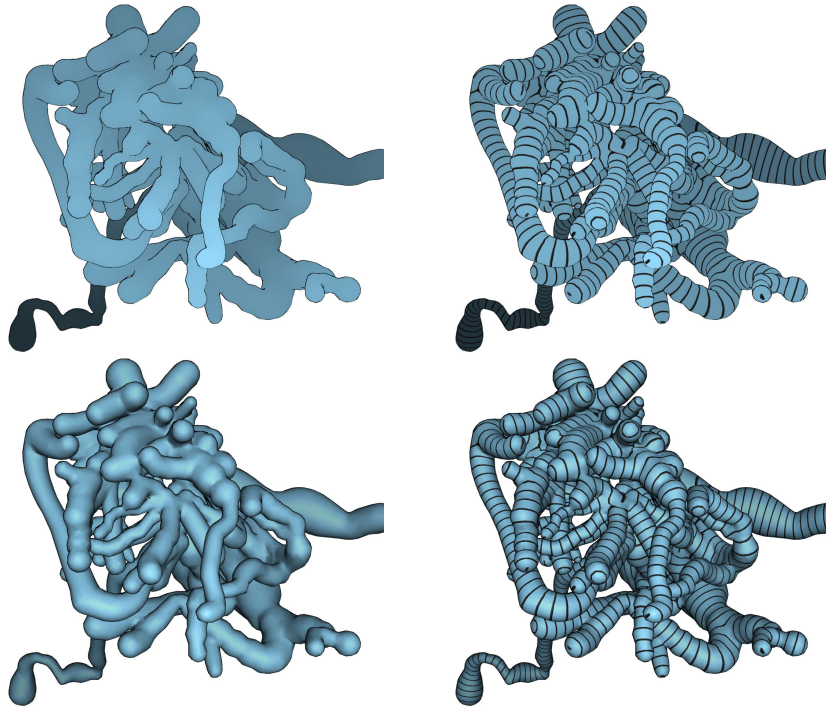
6 Acknowledgements

Fig. 10. Parameterized neuron by courtesy of Freie Universität Berlin, Department of Neurobiology. Top: depth shading. Bottom: full shading.

References

1. Floater, M.S., Hormann, K.: Surface parameterization: a tutorial and survey. In Dodgson, N.A., Floater, M.S., Sabin, M.A., eds.: Advances in multiresolution for geometric modelling. Springer Verlag (2005) 157–186
2. Hormann, K., Polthier, K., Sheffer, A.: Mesh parameterization: Theory and practice. In: SIGGRAPH Asia 2008, Course Notes. Number 11 (2008)
3. Tutte, W.T.: How to draw a graph. Proc. London Math. Soc. **s3-13**(1) (1963) 743–767
4. Haker, S., Angenent, S., Tannenbaum, A., Kikinis, R., Sapiro, G., Halle, M.: Conformal surface parameterization for texture mapping. IEEE Trans. on Visualization and Computer Graphics **6**(2) (2000) 181–189
5. Gu, X., Yau, S.T.: Global conformal surface parameterization. In: Symp. on Geom. Proc. (2003) 127–137
6. Jin, M., Wang, Y., Yau, S.T., Gu, X.: Optimal global conformal surface parameterization. In: In IEEE Visualization. (2004) 267–274
7. Kharevych, L., Springborn, B., Schröder, P.: Discrete conformal mappings via circle patterns. ACM Trans. on Graphics **25**(2) (2006)
8. Tong, Y., Alliez, P., Cohen-Steiner, D., Desbrun, M.: Designing quadrangulations with discrete harmonic forms. In: Eurographics Symp. on Geom. Proc. (2006)

9. Dong, S., Bremer, P.T., Garland, M., Pascucci, V., Hart, J.: Spectral surface quadrangulation. ACM SIGGRAPH (2006)

10. Ray, N., Li, W.C., Lévy, B., Sheffer, A., Alliez, P.: Periodic global parameterization. ACM Trans. Graph. **25**(4) (2006) 1460–1485

11. Kälberer, F., Nieser, M., Polthier, K.: Quadcover - surface parameterization using branched coverings. Comput. Graph. Forum **26**(3) (2007) 375–384

12. Ray, N., Vallet, B., Li, W.C., Lévy, B.: N-symmetry direction field design. ACM Trans. Graph. **27**(2) (2008) 1–13

13. Palacios, J., Zhang, E.: Rotational symmetry field design on surfaces. ACM Trans. on Graphics **26**(3) (2007) 55:1–55

14. Zhang, E., Hays, J., Turk, G.: Interactive Tensor Field Design and Visualization on Surfaces. IEEE Trans. on Visualization and Computer Graphics (2007) 94–107

15. Lai, Y.K., Jin, M., Xie, X., He, Y., Palacios, J., Zhang, E., Hu, S.M., Gu, X.D.: Metric-driven rosy fields design. Technical report, Tsinghua Univ., Beijing (2008)

16. Huysmans, T., Sijbers, J., Verdonk, B.: Parametrization of tubular surfaces on the cylinder. In: WSCG (Journal Papers). (2005) 97–104

17. Antiga, L., Steinman, D.: Robust and objective decomposition and mapping of bifurcating vessels. IEEE Trans. on Medical Imaging **23**(6) (2004) 704–713

18. Zhu, L., Haker, S., Tannenbaum, A.: Flattening maps for the visualization of multi-branched vessels (2005)

19. Farkas, H.M., Kra, I.: Riemann Surfaces. Springer Verlag (1980)

20. Fulton, W.: Algebraic Topology, A first course. Springer Verlag (1995)

21. Jost, J.: Compact Riemann Surfaces. Springer (2002)

22. Cohen-Steiner, D., Morvan, J.M.: Restricted delaunay triangulations and normal cycle. In: Proc. of Symp. on Comp. Geom., ACM Press (2003) 312–321

23. Hildebrandt, K., Polthier, K.: Anisotropic filtering of non-linear surface features. Computer Graphics Forum **23**(3) (2004) 391–400

24. Erickson, J., Whittlesey, K.: Greedy optimal homotopy and homology generators. In: Proc. 16th ACM-SIAM Symp. on Discrete Algorithms. (2005) 1038–1046

25. Polthier, K., Preuss, E.: Identifying vector field singularities using a discrete Hodge decomposition. In: Visualization and Mathematics III. Springer (2003) 113–134

26. Praun, E., Hoppe, H., Webb, M., Finkelstein, A.: Real-time hatching. In: SIGGRAPH. (2001) 581

27. Zöckler, M., Stalling, D., Hege, H.C.: Fast and intuitive generation of geometric shape transitions. The Visual Computer **16**(5) (2000) 241–253