Orientation Field Guided Texture Synthesis

Abstract We present a framework for example-based texture synthesis with feature directions aligned to vector fields with two way rotational symmetry, also known as orientation fields. Through a simple variational formulation, the framework allows the user to design the orientation field with intuitive controls, by interactively manipulating singularities and field directions. The resulting field is then used to guide a parallel synthesis. Our contribution is twofold: a design tool for orientation fields with a natural boundary condition, and a parallel texture synthesis adapted specifically for using such fields in feature alignment. We demonstrate the advantages of the procedure through examples on planar and curved patches with trivial topology.

Keywords orientation fields, natural boundary condition, texture synthesis

Beibei Liu[1] (刘蓓蓓), Yanlin Weng[2] (翁彦琳), Jiannan Wang[2] (王建南), and Yiying Tong[1] (童一颖), *Member, IEEE*

[1]*Computer Science and Engineering Department, Michigan Statue University, U.S.A.*

[2]*Computer Science Department, Zhejiang University, Hangzhou, 310058, China*

1 Introduction

Texture, information about geometric details or material properties stored as 2D images, is indispensable in decorating 3D surfaces. Texture synthesis is a popular method to acquire textures. Alignment of the feature directions in texture synthesis is often implemented by a user-specified guidance direction field. Such a direction field is also mandatory for most methods when the synthesis is performed on surfaces.

For textures with two way rotational symmetry (2-RoSy), the guidance fields do not have be continuous everywhere. Instead, nearby vectors should be allowed to have nearly opposite direc-

Corresponding Authors: Yanlin Weng (weng@cad.zju.edu.cn) and Yiying Tong (ytong@msu.edu). Supported in part by NSF grants IIS-0953096 and CMMI-1250261, NSF of China (No.61003145) and the National High-Tech R&D program of China (No.2012AA011503).

tions to have natural singularities. Such fields are called 2-RoSy fields, or orientation fields. For instance, fingerprints are oblivious to whether the direction is forward or backward along the ridges. The principal curvature direction fields on surfaces is another extremely important example of 2-RoSy. In this paper, we specifically target at developing a method for handling such fields.

Widely used in various applications such as rendering, image editing and video synthesis, texture synthesis techniques should meet some common key requirements to be practical. Among others, intuitive control is often a life-saver in the design process. For instance, if the nonintuitive Dirichlet or Neumann boundary conditions are used, an artist may have to spend extra time on figuring out their influences when preparing the feature alignment fields, while a natural boundary condition can lead to an expected smooth field without additional user intervention. Smoothness in the final texture is another basic requirement for seamless appearance of the objects being decorated. However, generating seamless textures can be a challenge for orientation fields in regions with discontinuity in the choice of forward or backward directions. Special care must be given in the treatment of neighborhoods containing such discontinuity, which is inevitable for generic orientation fields. Furthermore, the algorithm should be efficient enough to make the system interactive and flexible.

Aiming at the above design goals, we introduce a novel framework for the entire pipeline of orientation field guided texture synthesis. Our main contributions include

- A tangent vector field design tool with the natural boundary condition induced by minimizing the Dirichlet energy.
- An orientation field design tool based on an associated vector field, with straightforward control over oriented singularities as well as directions.
- A parallel texture synthesis adapted to handle the discontinuity in orientation fields.

In the following, we first briefly discuss the most relevant related work on orientation field design and on texture synthesis. We then present our vector field design algorithm for natural/free boundary condition in Sec. 2. The singularity control in orientation fields is presented in Sec. 3. Next, we elaborate on the nontrivial modifications to make parallel appearance texture synthesis applicable to orientation fields in Sec. 4. Examples demonstrating the capability of our system are shown in Sec. 5. We conclude the paper with a discussion on future work in Sec. 6.

Related work

Vector field design In a flexible and intuitive texturing system, users should be able to control the orientation and sizing of textures on surface. Such controls are often achieved by designing a vector field prescribing one of the axes of the local coordinate frames. Many graphics applications other than texture synthesis, such as non-photorealistic-rendering, parameterization and visualization, also rely on smooth vector fields. Some vector field design tools used interpolation from scattered user-specified directions (e.g., [1],[2]), and others also allow singularity control (e.g., [3],[4]). Zhang et al.[3] proposed to use geodesic polar maps and parallel transport to create radial basis functions given users' requirements. Fisher et al. [4] employed the tools from discrete exterior calculus, by representing the field as discrete 1-forms and solving linear equations with user-defined constraints. Our vector field design is based on ,[4] with one main difference on the treatment of the free boundary condition. The change is necessary as the efficient tools from discrete exterior calculus cannot express the vector field Dirichlet energy of vector fields as a direct combination of the basic operators in exterior calculus for surfaces with boundaries. More precisely, in this case, the Dirichlet energy of a tangent vector field is different from the sum of the squared sum of the L_2-norms of its divergence field and its curl field, as detailed in Sec 2.

Fields with symmetries In cases where the textures or features contain N-way rotational symmetries, the feature alignment vector at any given point in the guidance vector field can be seen as a representative of the N feature alignment vectors obtained by rotating the vector by integer multiples of $2\pi/N$, which form equivalence classes of vectors at each point. If the representative vector field is not continuous directly, but continuous if we can freely choose any of the N vectors in the equivalence classes when comparing nearby vectors, it is called an N-way rotational symmetry (N-RoSy) field. The special case of 2-RoSy field has been known as orientation field, and long been used in fingerprint research, e.g., in,[5] since the ridges and valleys on a fingerprint image cannot distinguish forward or backward directions along them. The models used for detecting singularities commonly used in fingerprints are often with few parameters, but more accurate orientation fields are proven important in enhancing latent fingerprints, those collected at crime scenes.[6] In graphics, Zhang et al.[7] introduced interactive 2-RoSy design on surfaces, and Palacios et al.[8] extended this idea to N-RoSy fields with $N \geq 3$. Building N-RoSy fields can also be achieved through specifying compatible singularities and modifying parallel transport: Lai et al.[9] focused on designing Riemannian metrics compatible with the local symmetry of N-RoSy fields, while Crane et al.[10] proposed to directly design a connection that is flat almost everywhere except for the singularities, instead of using the Levi-Civita connection induced by the Riemannian metric. Ray et al.[11] introduced the concept of turning numbers and offered another equivalent definition for singularities of N-RoSy fields. Affinity-based edit propagation can potentially be used to modified orientation fields with user strokes.[12]

Texture synthesis The literature on generating large texture patches automatically from given exemplars is vast, as such example-based texture synthesis techniques, among the state-of–the-art texture acquisition methods, are easy to use and capable of producing results without unnatural artifacts or periodicity. It has been applied extensively in practice, e.g. in game engines and feature films.[13] Most of these example-based methods are based on the Markov Random Field theory, assuming that the combined probability distribution of pixels has stationarity and locality. The actual implementation can be pixel-based, patch-based, or more generally, optimization-based. Patch-based algorithms[14]–[16] extract consistent patches from the exemplar and glue them together to create large textures. They can be highly efficient with neighborhoods faithful to those in the exemplar. However, they do not provide large variation and can be inefficient for runtime synthesis due to the sequential nature of the process. Some pixel-based algorithms, on the other hand, are able to generate high quality results interactively through multi-scale Gaussian image stacks and parallel texture synthesis.[17],[18] Extensions to perform texture synthesis on surfaces by forming seamless texture across atlas charts can be found in, e.g., .[18],[19] Texture synthesis can also be applied for bidirectional texture functions,[20] gradient solid textures,[21] semi-regular solid texture,[22] image mixing,[23] and even for fire animation.[24] It is also possible to mix symmetry from one texture to guide the synthesis of another.[25] Refer to recent surveys [26],[27] for more information on texture synthesis.

Relation to our work For our task of orientation field guided texture synthesis, we use a modified version of the tangent field design method.[4] The vector field design through a weighted least squares method leads to a straightforward Poisson-equation-like linear system. When the singularities are moved, there is no need to rebuild spanning trees as in,[10] or rerun discrete Ricci flows.[9] For index-1 singularities (poles), only the right hand side of the linear system is modified; for index-−1 singularities (saddles), an efficient increment to the Cholesky factorization of the left hand side can be performed. Furthermore, while texture synthesis depends only on the unit direction of the field in most cases, the true vector field design in engineering applications could benefit from a method that allows direct control over divergence and curl as in.[4] Our texture synthesis stage is based on [17] and ,[18] which manipulate pixel coordinates to overcome the issue of lack of efficiency in order-independent neighborhood matching. However, for orientation field guided synthesis, the upsampling and correction steps in the top-down multiscale approach must be substantially modified.

2 Vector Field Design with Natural Boundary Conditions

Before presenting our modification to the natural boundary condition, we briefly recap the method described in.[4] We demonstrate the necessity of our modification through examples.

The tangent vector field design problem is formulated as a weighted least squares problem in .[4] The design constraints are specified through user-controlled curl and divergence of the vector field, as well as direct constraints on the vectors, all at scattered locations. The curl is only non-zero at user-specified vortices, and the divergence is only non-zero at user-specified sources or sinks. The direct constraints can be any prescription of the vector at selected locations, but are often specified in batches through user sketch strokes.

When the relevant fields are expressed as discrete differential forms, the above weighted least squares problem has a straightforward implementation through discrete exterior calculus (DEC), a computational tool for performing calculus on meshes.[28] The resulting linear system is essentially a Poisson equation combined with terms from soft constraints.

2.1 Setup

The computation is carried out on a 2-manifold with boundary, represented by a triangle mesh M , with vertex set V, edge set E, and triangle set T. A vector field $\boldsymbol{u}$ can be stored as a 1-form, i.e. one scalar value per oriented edge $e_i \in E$, denoting the line integral along the edge $c_i = \int_{e_i} \boldsymbol{u}$. A scalar field s can be stored as a 0-form, one value per vertex $v_i \in V$, $s_i = s(v_i)$, or as a 2-form, one value per triangle $t_j \in T$, $s_j = \int_{t_j} s_j$. In particular, the divergence of a vector field can be represented by a 0-form, while its curl can be represented by a 2-form.

The usual differential operators in vector field analysis can be implemented through two basic operators in DEC. The first operator *differential*, or exterior derivative, $\boldsymbol{d}_k$ maps k-forms to $k{+}1$-forms, and the second operator *Hodge star* $*_k$ maps k-forms to $2{-}k$-forms in 2D. We may interpret $\boldsymbol{d}_0$ as ∇, $\boldsymbol{d}_1$ as $\nabla\times$, $*_0$ as multiplication by the area form, $*_1$ as rotation by 90° on the tangent plane, and $*_2$ as division by the area form. Other differential operators can be assembled from $\boldsymbol{d}$ and $*$, e.g. *co-differential* $\boldsymbol{\delta}_k = *_{k-1}^{-1}\boldsymbol{d}_{k-1}^T *_k$, which are adjoint to differentials. In particular, divergence $\nabla\cdot$ can be implemented by $\boldsymbol{\delta}_1$. On a triangle mesh, the $\boldsymbol{d}$'s are transposes of the signed incidence matrices, and the $*$'s are the ratios between the sizes of dual mesh cells and the corresponding primal mesh cells. We omit the subscripts when they can be determined from the context.

Assume that $\boldsymbol{U}$ represents the vector field, $\boldsymbol{S}$ the divergence field, $\boldsymbol{C}$ the curl field, $\boldsymbol{U_Z}$ the constraints on selected edges, where $\boldsymbol{Z}$ is the matrix projecting an array representing a 1-form onto an array assembled by one value per user-selected edge. The desired vector field can be computed by the weighted least squares solution of the following equations,

$$\boldsymbol{\delta U} = \boldsymbol{S},\ \ \boldsymbol{dU} = \boldsymbol{C},\ \ \boldsymbol{ZU} = \boldsymbol{U_Z},$$

leading to

$$\begin{aligned}&[*(\boldsymbol{d\delta}+\boldsymbol{\delta d}) + \boldsymbol{Z^T W Z}]\boldsymbol{U}\\ &= *\boldsymbol{dS} + *\boldsymbol{\delta C} + \boldsymbol{Z^T W U_Z},\end{aligned}$$

where $\boldsymbol{W}$ specifies the weighting of the direct constraints. Aside from the term induced by $\boldsymbol{Z}$, the resulting symmetric linear system is simply the vector field Poisson equation, where the Laplace-Beltrami operator $\boldsymbol{d\delta}+\boldsymbol{\delta d}$ is equivalent, up to a sign, to what can be obtained from the vector calculus identity

$$\nabla^2 = \nabla\nabla\cdot - \nabla\times\nabla\times\ .$$

2.2 Natural Boundary Conditions

In,[4] the free boundary condition, for the case when the vector field is not restricted to be at a certain angle with the boundary, is implemented by adding a term to properly include the integral of divergence for the partial Voronoi cells at the boundary. However, this still leaves a high-dimensional kernel for the resulting linear operator. Numerically the operator is likely to be positive definite due to the discretization, but it leads to spurious singularities, unless sufficient direct constraints are included or when the much denser bi-Laplacian is included.

Our remedy to the above problem is based on the observation that the free boundary condition should be obtained through minimizing the Dirichlet energy $\int_M |\nabla \boldsymbol{u}|^2$. Choosing a local orthonormal frame $\{\boldsymbol{e}_1, \boldsymbol{e}_2\}$ at each point, we denote the partial derivatives of the components of $\boldsymbol{u}$ by $u_{\alpha,\beta} = \frac{\partial u_\alpha}{\partial x_\beta}$. Assuming trivial connection, we ignore the curvature-related term (see the Weitzenböck formula in, e.g., [29]), and focus on the influence of the boundary

$$\int_M |\nabla \boldsymbol{u}|^2 = \int_M u_{1,1}^2 + u_{1,2}^2 + u_{2,1}^2 + u_{2,2}^2$$
$$= \int_M (u_{1,1}+u_{2,2})^2 + (u_{2,1}-u_{1,2})^2 \quad (1)$$
$$- 2(u_{1,1}u_{2,2}-u_{1,2}u_{2,1})$$
$$= \int_M (\nabla\cdot\boldsymbol{u})^2 + (\nabla\times\boldsymbol{u})^2 - 2\int_{\partial M} u_1 du_2 - u_2 du_1 \quad (2)$$

The boundary term in the last row is a result of Stokes' theorem. In standard calculus, we may rewrite the term as

$$-\int_{\partial M} (\boldsymbol{u} \times d\boldsymbol{u}) \cdot \boldsymbol{n},$$

where $\boldsymbol{n}$ is the surface normal.

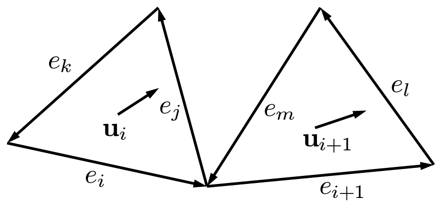

Fig. 1: Two consecutive edges along the boundary.

In the discrete setting, we express the Dirichlet energy as $\boldsymbol{U}^T \boldsymbol{L} \boldsymbol{U}$, where $\boldsymbol{U}$ is the discrete 1-form representation of $\boldsymbol{u}$, and $\boldsymbol{L}$ is the Laplacian-like matrix to be constructed. We initialize $\boldsymbol{L}$ to the sum of the divergence and curl terms, and then add the boundary term. To discretize the boundary term, we first turn the boundary integral into a summation over boundary edges

$$\sum_{e_i \in \partial M} (\boldsymbol{u}_i \times (\boldsymbol{u}_{i+1} - \boldsymbol{u}_i)) \cdot \boldsymbol{n}_i = \sum_{e_i \in \partial M} (\boldsymbol{u}_i \times \boldsymbol{u}_{i+1}) \cdot \boldsymbol{n}_i,$$

where we assume that e_{i+1} is the edge following e_i along the boundary, and $\boldsymbol{n}_i$ is the surface normal at their shared vertex.

Assuming the discrete curl for boundary triangles to be close to zero as in ,[4] we have a constant vector within each triangle, which allows us to simply choose any point (in particular, the barycenter) of the triangle for the evaluation of $\boldsymbol{u}_i$. A discrete 1-form is one value per edge, so $\boldsymbol{U} = (c_1, c_2, \ldots, c_n)$, where n is number of edges. For the pair of boundary triangles shown in Figure 1, we have

$$\boldsymbol{u}_i = c_i\boldsymbol{\phi}_i + c_j\boldsymbol{\phi}_j + c_k\boldsymbol{\phi}_k,$$
$$\boldsymbol{u}_{i+1} = c_{i+1}\boldsymbol{\phi}_{i+1} + c_l\boldsymbol{\phi}_l + c_m\boldsymbol{\phi}_m,$$

where $\boldsymbol{\phi}_i = \frac{1}{3}(\nabla\phi_{v_2} - \nabla\phi_{v_1})$ is the basis function for edge i pointing from v_1 to v_2 evaluated at the barycenter of the corresponding triangle, and ϕ_v is the linear basis function for vertex v. The update to $\boldsymbol{L}$ involves 18 terms in 9 pairs, e.g.

$$L_{jl} += (\boldsymbol{\phi}_j \times \boldsymbol{\phi}_l) \cdot \boldsymbol{n}_i, \; L_{lj} += (\boldsymbol{\phi}_j \times \boldsymbol{\phi}_l) \cdot \boldsymbol{n}_i.$$

On a curved patch, we may obtain the surface normal at the vertex from any reasonable weighted averaging.

As shown in Figure 2, any harmonic vector field can be added to a field without changing the target function in,[4] leading to spurious singularities, while in our case, the Dirichlet energy minimization produces the expected results.

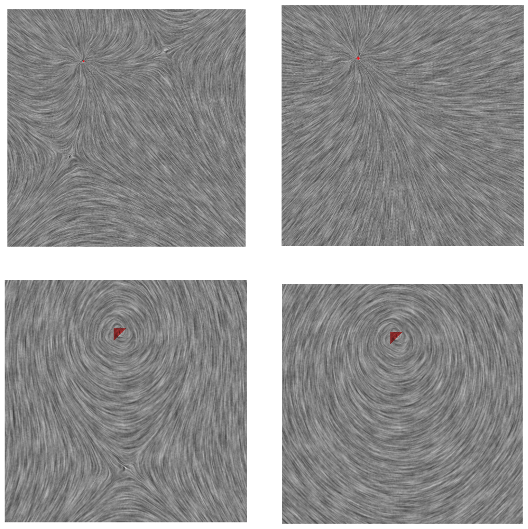

Fig. 2: Comparison of results. User input: a single source (top); a single vortex (bottom). Fisher et al.'s design method produced multiple spurious singularities (left); our method produced the minimizer of the Dirichlet energy (right).

3 Orientation Field Design

We follow the practice in ,[7] and represent the direction orientation field by the angle θ denoting the deviation from the x-axis of a local coordinate frame. We can construct the frame field by specifying the x-axis through the solution of the above natural boundary condition vector field design obtained by fixing a single vector, in the case of trivial topology. Otherwise, it can be computed by first specifying some singularities consistent with the Poincaré-Hopf index theorem, and use trivial connection [10] or discrete Ricci flow[9] to construct the frame field.

The orientation field can then be represented by a smooth vector field with the angle 2θ, since $2(\theta + \pi)$ leads to the same angle. Using a complex number to represent the vector in the local frame, we can use the square and square root operations for complex numbers to convert the orientation field and vector field from each other.

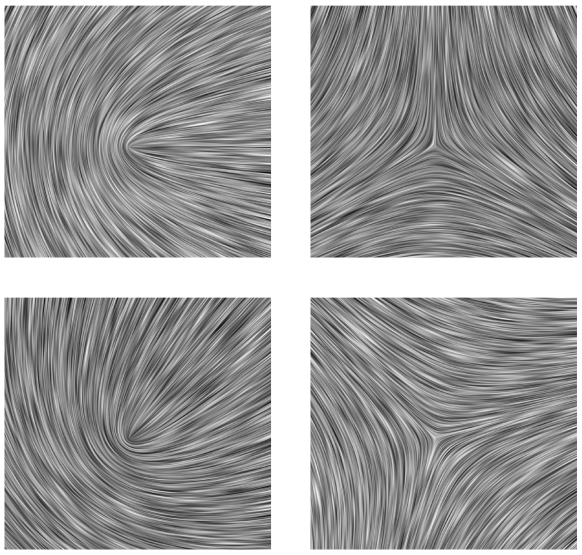

Fig. 3: Basic singularities for orientation fields: wedge (left) and trisector (right). Our system also provides control over the orientations. Top: original; Bottom: 45° rotated.

The major features in an orientation field are determined by the singularities. Two basic singularities, wedge and trisector (Figure 3) with index-$\frac{1}{2}$ and index-$-\frac{1}{2}$, respectively, can be used to produce singularities of arbitrary indices in the orientation field.

We present a simple method of specifying not only the singularity types and locations, but also their orientations. Wedges correspond to sources/sinks and vortices, for example, a source correspond to wedge with a horizontal flow line connected to the singularity. As described in,[3] we can see that the local field in a small neighborhood around a source vertex v have the form of $e^{i\theta}$, where θ is the angle between the displacement from v to the point in the neighborhood and the local x-axis direction. Thus, the corresponding orientation field is of the form $e^{i\theta/2}$, the expected wedge. For a vortex, the vector field is of the form $e^{i(\theta+\pi/2)}$. Combining the two with weights $\cos 2\alpha$ and $\sin 2\alpha$, we have the orientation field of the form $e^{i(\theta/2+\alpha)}$, i.e. a rotated wedge. The trisectors correspond to saddle points, which can be constructed by controlling the one-ring of a vertex v to have a vector of $e^{-i(\theta+2\alpha)}$ in each incident triangle. See the 45° rotated wedge and trisector in the bottom row of Figure 3.

With a proper combination of singularities of positive and negative indices, there would rarely be any uncontrolled singularities with our proper boundary condition. This behavior can be understood by following the same argument as in :[9],[10] additional vector field constraints can be seen as just smooth deviation from an initial orientation field satisfying the singularity constraints.

4 Texture Synthesis for 2-Rosy Field

Traditional controllable texture synthesis often uses a designed smooth vector field as user input. This is not the same as using an orientation field, since the field of representative vectors chosen from one of the two direction is inevitably discontinuous in the presence of at least one wedge or trisector. Thus as we adapt the strategy from,[17],[18] and perform the coarse-to-fine texture synthesis, we must introduce an “upside-down” mapping style to enforce the continuous appearance. We restrict our discussion to the planar case, as the curved patch case is treated by combining the Jacobian of the parameterization as in[18]

Anisometric texture synthesis

We first briefly summarize the appearance space synthesis in [18] before discussing the modifications. In the preparation stage, a Principal Component Analysis (PCA) is performed on the set of all 5×5-neighborhoods in the exemplar Gaussian stack at each level, to create a 8D appearance vector space, turning the exemplar into a 2D array of appearance vectors $\tilde{E}$. The parallel synthesis then repeats three main steps, namely upsampling, jittering, and correction, until the finest level texture is generated.

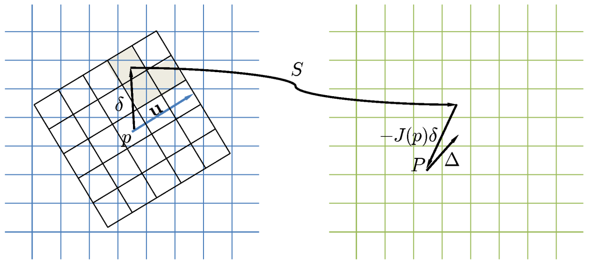

Fig. 4: One of the three predicted texture locations for the upper-right corner in the four-corner neighborhood.

The appearance vector at each output pixel is represented through a mapping to a point in the exemplar $\tilde{E}$. Denoting the texture exemplar coordinates for a pixel $\boldsymbol{p}$ in the output pyramid at level L by $S_L[p]$, the upsampling pass for the levle-L+1 pixels corresponding to level-L pixel $\boldsymbol{p}$ is rather straightforward, when there is a guidance field:

$$S_{L+1}[2\boldsymbol{p} + \tilde{\triangle}] = S_L(\boldsymbol{p}) + \boldsymbol{J}(\boldsymbol{p})\frac{1}{2}\triangle,$$

where

$$\triangle \in \{\begin{pmatrix} -1 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix}\},$$

$$\tilde{\triangle} = \frac{1}{2}(\triangle + \begin{pmatrix} 1 \\ 1 \end{pmatrix}),$$

and $\boldsymbol{J}(\boldsymbol{p})$ is the Jacobian matrix for local S to follow the guidance field, which is essentially a rotation matrix aligning x-axis to the given guidance direction combined with a possible scaling.

In the correction step, a four-corner neighborhood $N_S(\boldsymbol{p};\triangle)$ is sufficient due to the use of appearance vectors. For better convergence in the parallel synthesis, [18] suggested to average the appearance vector for each corner $\triangle$ from the corner values predicted from three offset locations $\delta(\triangle, \boldsymbol{M}) = \hat{\varphi}(\triangle) + \hat{\varphi}(\boldsymbol{M}\triangle), \boldsymbol{M} \in \mathcal{M}$, where

$$\mathcal{M} = \{\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}\},$$

and $\hat{\varphi}(\triangle) = \lfloor \varphi(\triangle)/\|\varphi(\triangle)\| + 1/2 \rfloor$ is the normalized version of warped offset $\varphi(\triangle) = \boldsymbol{J}^{-1}(\boldsymbol{p})\triangle$. As the predicted texture location is $P(\boldsymbol{p}, \delta) = S[\boldsymbol{p} + \delta] - \boldsymbol{J}(\boldsymbol{p})\delta$, shown in Figure 4, the final formula is

$$N_S(\boldsymbol{p};\triangle) = \frac{1}{3}\sum_{M\in\mathcal{M}} \tilde{E}[P(\boldsymbol{p}, \delta(\triangle, \boldsymbol{M})) + \triangle].$$

The neighborhood is then compared with the precomputed neighborhoods in the exemplar. For fast comparison on Graphics Processing Unit (GPU), the neighborhood can be further compressed through another PCA.

For efficiency, the search of best-matching exemplar pixel is limited to the k-coherent set

$$\mathcal{C}(\boldsymbol{p}) = \{C(\boldsymbol{p}, \triangle, i) | i = 1 \ldots k, \|\triangle\| < 2\},$$

where the candidates are predicted from nearby points $p+\triangle$ with the precomputed k-coherent offset C'_i,

$$C(\boldsymbol{p}, \triangle, i) = S[\boldsymbol{p}+\triangle] + C'_i(S[\boldsymbol{p}+\triangle]) - \boldsymbol{J}(\boldsymbol{p})\triangle.$$

We follow their practice of choosing $k = 2$.

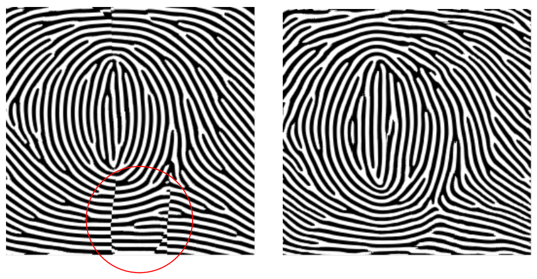
Fig. 5: Comparison of the results from the parallel anisometric texture synthesis method without (left) and with (right) our modifications. The representative vector field has discontinuity within the red circle.

Handling orientation

When the anisometric parallel synthesis is applied to orientation fields, there are visible artifacts when the representative vectors are changing to the opposite directions (Figure 5). Increasing the amount of jittering or rearranging the texture exemplar in a more symmetric way would not solve the problem. Cutting the output into charts according the field and use indirection map would not be less costly and less effective than our solution.

Our modification is based on the observation that the discontinuity in the texture is mainly due to the incorrectly predicted texture coordinates $P(\boldsymbol{p}, \delta)$ as shown in Figure 6, which affects both the neighborhood construction and the candidate sets. This issue can be fixed by modifying the prediction to

$$\hat{P}(\boldsymbol{p}, \delta) = S[\boldsymbol{p} + \delta] - \boldsymbol{J}(\boldsymbol{p} + \delta)\delta.$$

The four-corner neighborhood is also modified to

$$\begin{aligned}\hat{N}_S(\boldsymbol{p}; \triangle) &= \frac{1}{3} \sum_{\boldsymbol{M} \in \mathcal{M}} \tilde{E}[\hat{P}(\boldsymbol{p}, \delta(\triangle, \boldsymbol{M})) \\ &+ (-1)^{c(\boldsymbol{p}+\delta, \boldsymbol{p})} \triangle],\end{aligned}$$

where the consistency $c(\boldsymbol{p} + \delta, \boldsymbol{p})$ is defined as $\boldsymbol{u}(\boldsymbol{p}+\delta) \cdot \boldsymbol{u}(\boldsymbol{p}) < 0$, a binary indicator of the presence of a 180° rotation.

If we are using color pixels, this would have sufficed. However the appearance vectors represent 5×5-neighborhoods. So they would be containing the wrong appearance if we use the appearance vector at the flipped predicted location directly. To handle the issue without losing much efficiency, we build the 8D appearance space from the set containing both the 5×5-neighborhoods and their rotated images. Then we store two appearance images for the exemplar, one for the original $\tilde{E}_0$, the other for the rotated $\tilde{E}_1$. We store one boolean variable $I(\boldsymbol{p})$ for each output point, indicating whether the rotated appearance is used.

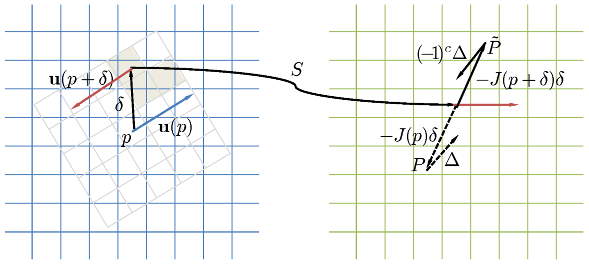

Fig. 6: Following the dashed direction would have produced a wrong prediction, while $\tilde{P}$ properly takes into account the mutual orientation.

Putting these together, we have the final formula for the neighborhood construction

$$\tilde{N}_s(\boldsymbol{p};\triangle) = \frac{1}{3}\sum_{M\in\mathcal{M}} \tilde{E}_{\alpha(\pm p,\delta)}[\tilde{P}(\boldsymbol{p},\delta(\triangle,M)) + (-1)^{\alpha(\boldsymbol{p},\delta)}\triangle], \quad (3)$$

where

$$\alpha(\boldsymbol{p},\delta) = I(\boldsymbol{p}+\delta) + c(\boldsymbol{p}+\delta,\boldsymbol{p})$$

combines the effects of the consistency and the current indicator, and

$$\tilde{P}(\boldsymbol{p},\delta) = S[\boldsymbol{p}+\delta] - (-1)^{\alpha(\boldsymbol{p},\delta)}\boldsymbol{J}(\boldsymbol{p})\delta$$

is the modified prediction.

A modification is also in place for the k-coherent candidates

$$C(\boldsymbol{p},\triangle,i) = S[\boldsymbol{p}+\triangle] + C_i'(S[\boldsymbol{p}+\triangle]) - (-1)^{\alpha(\boldsymbol{p},\triangle)}\boldsymbol{J}(\boldsymbol{p})\triangle.$$

When comparing the neighborhood information constructed with the candidates, $\tilde{E}_{\alpha(\boldsymbol{p},\triangle)}(C(\boldsymbol{p},\triangle,i))$ should be used to account for the possible relative rotations.

When the best match is found at the candidate predicted by offset $\triangle$, the indicator I is updated as well as S,

$$I(\boldsymbol{p}) = \alpha(\boldsymbol{p},\triangle).$$

Finally, the upsampling step is also adapted to

$$S_{L+1}[2\boldsymbol{p}+\tilde{\triangle}] = S_L(\boldsymbol{p}) + (-1)^{I(\boldsymbol{p})}\boldsymbol{J}(\boldsymbol{p})\frac{1}{2}\triangle,$$

and

$$I_{L+1}[2\boldsymbol{p}+\tilde{\triangle}] = I_L(\boldsymbol{p}) + c_{L,L+1}(\boldsymbol{p},2\boldsymbol{p}+\tilde{\triangle}),$$

where $c_{L,L+1}(\boldsymbol{p},\boldsymbol{q})$ is defined to be $\boldsymbol{u}_L(\boldsymbol{p})\cdot\boldsymbol{u}_{L+1}(\boldsymbol{q}) < 0$, a binary valued function for checking the consistency of the orientation field between different scales, as the coarse and fine levels of the output image may choose different representatives when downsampling from the original orientation fields.

5 Results

The tests of our algorithm on examples were performed on a regular laptop with Intel Core2Dual with 4GB memory. In all of our tests, the method took no more than a fraction of a second, allowing for interactive manipulation of the singularity and direction constraints, even though our implementation is not optimized. In theory, we can reach the same efficiency of [4] and [18] in the respective stages, as only negligible overhead is incurred by the modifications to the 1-form-based tangent design method and parallel control texture synthesis.

In Figure 7, we show that our system can easily create fingerprint-like images imitating the five main categories of singularity layouts in human fingerprints. We show the method applied to more exemplars for planar regions with orientation fields in Figure 8. We use the same procedure in [18] for generating the texture in texture domain while taking into account the Jacobian of the parameterization, and some results are shown in Figure 9.

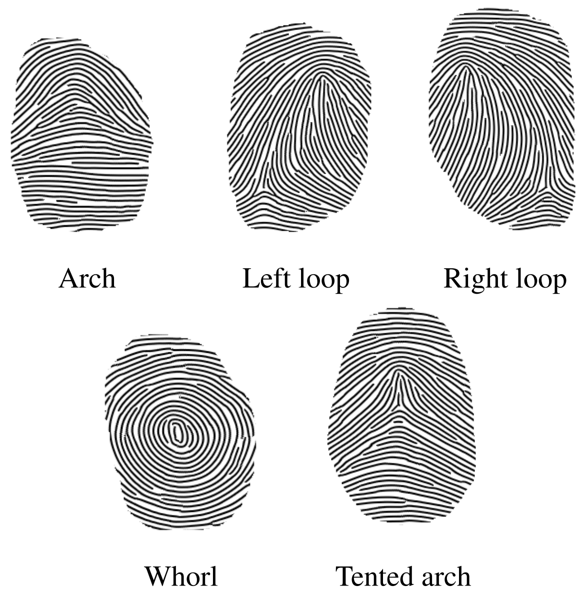

Fig. 7: Examples for the five major categories of fingerprints generated by our texture synthesis

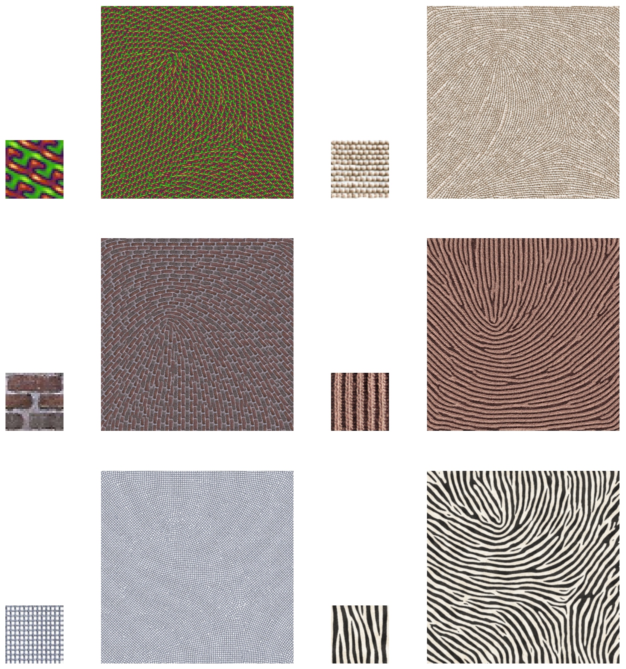

Fig. 8: Results with various textures on planar regions

Limitations There is no strict guarantee that additional saddle points would not emerge in our vector field design, if, e.g., we place two sources close to each other. However the same could happen for methods such as:[9],[10] if one specifies some vector direction constraints as in our saddle point placement, extra singularities would have to be generated in addition to those used in constructing the almost everywhere flat metrics or connections. On the other hand, in practice, with a proper mixture of positive and negative singularities, which does not produce excessively large indices in local regions, it would take some strong vector direction constraints to produce additional singularities with any method with proper free boundary condition. Another issue is that our texture synthesis does not provide direct control over the bifurcation and ending of the features contained the exemplar (See, e.g. Figures 7 and 8), but this is common to many anisometric texture synthesis methods.

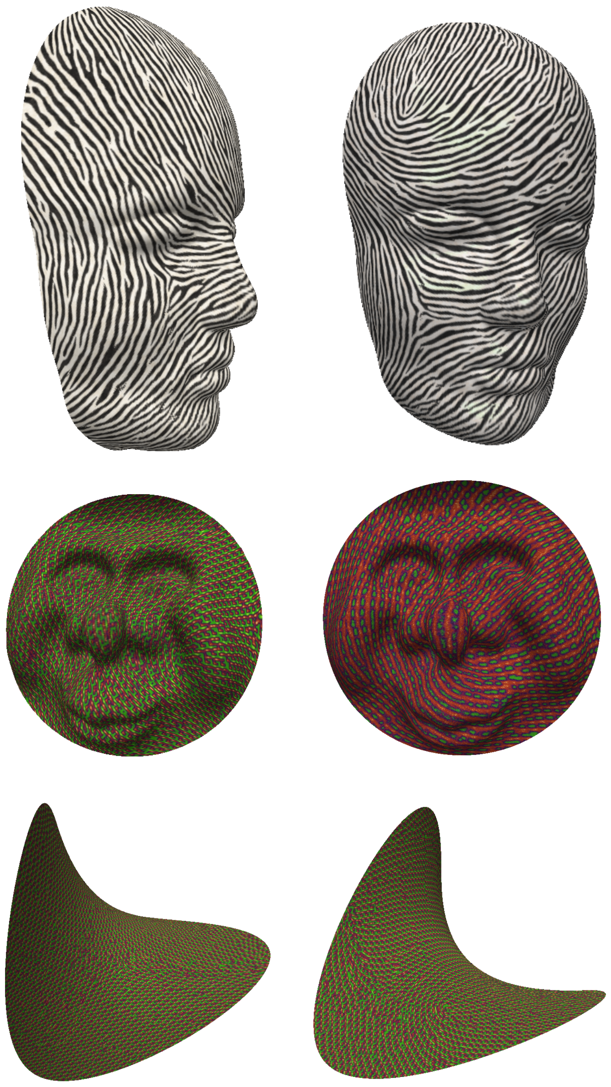
Fig. 9: Results for orientation fields on curved patches

6 Conclusion

We presented a simple framework based on tangent vector field design. We eliminated the spurious singularities produced by the free boundary condition through including the missing boundary term. Given a local frame field with trivial connection, we covert the vector field into an orientation field by taking the square root of the complex representation of the vector in the local frame, which halves the angle to the x-axis. We also provide control over the orientations of the wedge and trisector singularities. The designed orientation field can then be used in a parallel texture synthesis, adapted for orientation fields. Such texture synthesis allows on-the-fly synthesis and is GPU-friendly, due to its order-independence.

In the future, we will explore the effects of combining trivial connection or metric-driven N-RoSy on surfaces with arbitrary topology, generalize the parallel texture synthesis to N-RoSy fields, and implement applications of the method to latent fingerprint enhancement.

References

[1] Praun E, Finkelstein A, Hoppe H. Lapped textures. Proceedings of the 27th annual conference on Computer graphics and interactive techniques, 2000, pp.465–470. 3

[2] Turk G. Texture synthesis on surfaces. Proceedings of the 28th annual conference on Computer graphics and interactive techniques, 2001, pp.347–354. 3

[3] Zhang E, Mischaikow K, Turk G. Vector field design on surfaces. *ACM Trans. Graph.*, 2006, 25(4):1294–1326. 3, 9

[4] Fisher M, Schröder P, Desbrun M, Hoppe H. Design of tangent vector fields. *ACM Trans. Graph.*, 2007, 26(3). 3, 4, 5, 6, 7, 13

[5] Sherlock B G, Monro D M. A model for interpreting fingerprint topology. *Pattern Recognition*, 1993, 26(7):1047 – 1055. 3

[6] Feng J, Zhou J, Jain A. Orientation field estimation for latent fingerprint enhancement. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2013, 35:925–940. 3

[7] Zhang E, Hays J, Turk G. Interactive tensor field design and visualization on surfaces. *Visualization and Computer Graphics, IEEE Transactions on*, 2007, 13(1):94–107. 3, 8

[8] Palacios J, Zhang E. Rotational symmetry field design on surfaces. *ACM Trans. Graph.*, 2007, 26(3). 3

[9] Lai Y K, Jin M, Xie X, He Y, Palacios J, Zhang E, Hu S M, Gu X. Metric-driven rosy field design and remeshing. *Visualization and Computer Graphics, IEEE Transactions on*, 2010, 16(1):95–108. 3, 4, 8, 9, 13

[10] Crane K, Desbrun M, Schröder P. Trivial connections on discrete surfaces. *Computer Graphics Forum*, 2010, 29(5):1525–1533. 4, 8, 9, 13

[11] Ray N, Vallet B, Li W C, Lévy B. N-symmetry direction field design. ACM Transactions on Graphics, 2008. Presented at SIGGRAPH. 4

[12] Xu K, Li Y, Ju T, Hu S M, Liu T Q. Efficient affinity-based edit propagation using kd tree. ACM Transactions on Graphics (TOG), 2009, p.118. 4

[13] Eisenacher C, Tappan C, Burley B, Teece D, Shek A. Example-based texture synthesis on disney's tangled. ACM SIGGRAPH 2010 Talks, Los Angeles, California, 2010, pp.32:1–32:1. 4

[14] Kwatra V, Schdl A, Essa I, Turk G, Bobick A. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics, SIGGRAPH 2003*, 2003, 22(3):277–286. 4

[15] Wu Q, Yu Y. Feature matching and deformation for texture synthesis. ACM SIGGRAPH

2004 Papers, Los Angeles, California, 2004, pp.364–367. 4

[16] Barnes C, Shechtman E, Finkelstein A, Goldman D. Patchmatch: a randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics-TOG*, 2009, 28(3):24. 4

[17] Lefebvre S, Hoppe H. Parallel controllable texture synthesis. *ACM Trans. Graph.*, 2005, 24(3):777–786. 4, 5, 9

[18] Lefebvre S, Hoppe H. Appearance-space texture synthesis. *ACM Trans. Graph.*, 2006, 25(3):541–548. 4, 5, 9, 10, 13

[19] González F, Patow G. Continuity mapping for multi-chart textures. *ACM Trans. Graph.*, 2009, 28(5):109:1–109:8. 4

[20] Xu K, Wang J, Tong X, Hu S M, Guo B. Edit propagation on bidirectional texture functions. Computer Graphics Forum, 2009, pp.1871–1877. 4

[21] Zhang G X, Lai Y K, Hu S M. Efficient synthesis of gradient solid textures. *Graph. Models*, 2013, 75(3):104–117. 4

[22] Du S P, Hu S M, Martin R R. Semiregular solid texturing from 2d image exemplars. *IEEE Transactions on Visualization and Computer Graphics*, 2013, 19(3):460–469. 4

[23] Zhang F L, Cheng M M, Jia J, Hu S M. Imageadmixture: Putting together dissimilar objects from groups. *IEEE Transactions on Visualization and Computer Graphics*, 2012, 18(11):1849–1857. 4

[24] Lever J, Komura T. Real-time controllable fire using textured forces. *The Visual Computer*, 2012, 28(6-8):691–700. 4

[25] Kim V G, Lipman Y, Funkhouser T A. Symmetry-guided texture synthesis and manipulation. *ACM Trans. Graph.*, 2012, 31(3):22. 4

[26] Turk G. Part ii: texturing surfaces and geometry creation. ACM SIGGRAPH 2007 courses, San Diego, California, 2007. 4

[27] Wei L Y, Lefebvre S, Kwatra V, Turk G. State of the art in example-based texture synthesis. Eurographics 2009, State of the Art Report, EG-STAR, 2009. 4

[28] Desbrun M, Kanso E, Tong Y. Discrete Differential Forms for Computational Modeling. In *Discrete Differential Geometry*, Bobenko, A Schröder, P Sullivan, J Ziegler G (ed.), Springer, 2008, pp.287–324. 5

[29] Petersen P. Riemannian geometry. Springer New York, 2006. 6