Feature-Aligned Shape Texturing

Kai Xu[⋆§∗] Daniel Cohen-Or[⁎] Tao Ju[†] Ligang Liu[‡∗] Hao Zhang[⋆] Shizhe Zhou[‡] Yueshan Xiong[§]

[⋆]Simon Fraser University [⁎]Tel-Aviv University [†]Washington University at St. Louis
[‡]Zhejiang University [§]National University of Defense Technology

Abstract

The essence of a 3D shape can often be well captured by its salient feature curves. In this paper, we explore the use of salient curves in synthesizing intuitive, shape-revealing textures on surfaces. Our texture synthesis is guided by two principles: matching the direction of the texture patterns to those of the salient curves, and aligning the prominent feature lines in the texture to the salient curves exactly. We have observed that textures synthesized by these principles not only fit naturally to the surface geometry, but also visually reveal, even reinforce, the shape's essential characteristics. We call these *feature-aligned shape texturing*. Our technique is fully automatic, and introduces two novel technical components in vector-field-guided texture synthesis: an algorithm that orients the salient curves on a surface for constrained vector field generation, and a feature-to-feature texture optimization.

Keywords: Texture synthesis, salient features, feature alignment

1 Introduction

Salient curves have been explored extensively in non-photorealistic rendering of 2D images and 3D surface models. An underlying assumption is that a small set of salient lines can capture well the geometric features of a complex shape. For example, a 2D image can be abstracted by lines at strong local maxima of image gradients, and a 3D surface can be intuitively visualized by lines highlighting the protrusions and indentations. The expressiveness of these salient curve features further suggests that they may play important roles in the decoration of shapes, such as by colors or textures. That is, the decorative features may better serve the purpose of enriching the shape appearance when they are aligned with the salient curves. Indeed, this is observed in the 2D case in the recent work of Orzan et al. [2008], who showed that adding colors that are diffused from a set of planar curves dramatically enhances the vividness of the shape that is conveyed by those curves.

In much of the same spirit as [Orzan et al. 2008], we explore ornamenting a 3D surface model with synthesized texture patterns that are aligned with salient curves on the surface. Previously, texture synthesis was mostly guided by user inputs that indicate the placement and orientation of textures. While user guidance is important when the surface geometry does not provide strong hint for the layout of textures, a large class of models do possess perceptually prominent shape features that should be utilized. Salient curve features, in particular, are well suited for guiding the synthesis, not only because they capture the essence of the 3D shapes, but also due to the availability of automatic algorithms, such as [Yoshizawa et al. 2005], that can robustly detect curve features on surfaces.

Our synthesis is guided by two principles. First, we make the synthesized texture patterns follow the direction of the salient curves on the input surface. Second, when prominent line features exist in the original texture pattern, we exactly line up these texture features with the salient surface features. We have observed, on many models with prominent curve features (e.g., the one in Figure 1), that syntheses guided by these two principles, which we call *feature-aligned shape texturing*, not only fit the texture to the surface more naturally than those without, but also tend to enrich the overall perception of the shape itself. Such synthesis could be offered as an automated option to a modeler to improve texture appearance on surfaces that are rich in curve features.

We implement feature-aligned texturing within the optimization-based framework that has been successful in synthesis from texture exemplars [Kwatra et al. 2005; Wexler et al. 2007]. In this framework, texture is synthesized along a user-guided vector field on the surface, maximizing the similarity between the texture pattern in the oriented neighborhood of each vertex and the appearance of a texture examplar. We introduce two novel algorithmic components into this framework that are essential for feature-aligned texturing. To orient the textures by the direction of salient curves, we propose an algorithm for determining the orientation of vectors along the salient curves, which are used as constraints to obtain a smooth curve-guided vector field with reduced singularities. To enforce exact placement of texture features along the surface features, we augment the texture optimization formulation by an exact feature-to-feature alignment. The complete synthesis method is fully automatic, and works in conjunction with any existing methods that extract feature curves on surfaces.

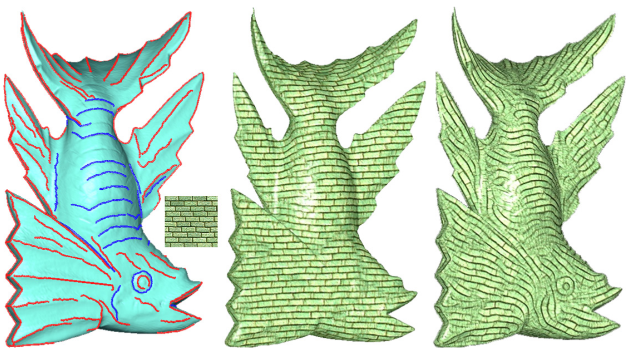

Figure 1: Decorating a shape with textures without user assistance often conceals the shape's rich geometric details (b). Our fully automatic algorithm synthesizes textures that better reveal, and even enhance, the shape appearance (c) — here the fish has "awoken". This is done by orienting the texture patterns with the direction of the salient curves on the shape (a) and enforcing exact feature-to-feature alignment between the salient curves and texture features.

[∗]Corresponding authors: ligangliu@zju.edu.cn, kaixu@nudt.edu.cn

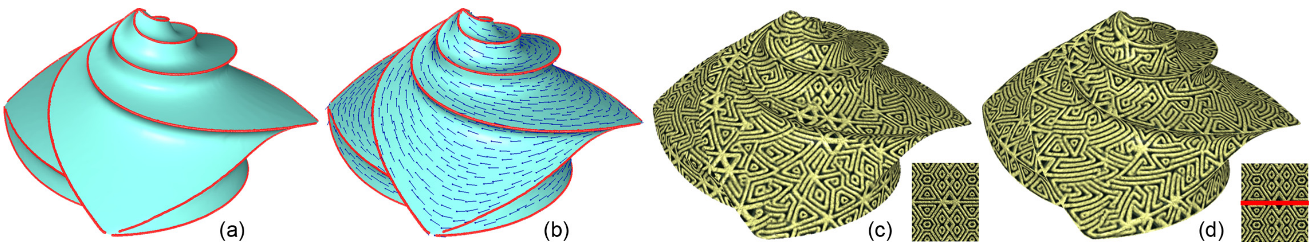

Figure 2: Our texture synthesis pipeline: starting from a surface with extracted salient curves (a), we first compute a smooth vector field that follows the curve directions (b), and then synthesize the texture along the vector field from a given exemplar (c). If prominent features exist in the exemplar, we enforce exact alignment of these features with the salient curves (d).

Contributions: The driving goal of feature-aligned shape texturing is to reveal and enhance the intrinsic geometric features of the 3D shape using textures. In this respect, our work makes a first step in a direction that complements existing efforts in texture synthesis that are mostly concerned with the continuity, low distortion of the textures or their conformation to user constraints. We make two main contributions in this direction:

- While salient feature curves are known to play a critical role in *perceiving* a shape, we harness these curves to *decorate* the shape's surface. In particular, we observe that synthesis guided by two principles (directional agreement between texture and salient curves on a surface as well as exact feature alignment) provides a natural fit of texture to an object while revealing the shape's essential characteristics.
- We develop a fully automatic texture synthesis method, and introduce two novel algorithm components. We propose an algorithm for computing a smooth vector field from a set of un-oriented curve constraints, and a feature-to-feature texture optimization technique for 3D surfaces.

2 Related work

Our work is concerned with texture synthesis guided by surface features, and draws inspirations from three related areas: texture synthesis, vector field design, and feature-based shape perception.

Texture synthesis: There is a rich body of work on synthesizing textures directly on surfaces (see [Wei et al. 2009]). The pixel-based approach was simultaneously generalized by [Wei and Levoy 2001] and [Turk 2001] to surfaces. The patch-based approach was first introduced in lapped texture by Praun et. al. [2000], where user-specified irregular texture patches are randomly pasted onto a surface. Texture optimization [Kwatra et al. 2005] has recently been extended to surfaces [Kwatra et al. 2007; Chen et al. 2009], multiscale texture synthesis from multiple texture exemplars [Han et al. 2008] and inverse texture synthesis [Wei et al. 2008]. Fu and Lung [2005] have extended Wang tiles [Cohen et al. 2003] to surfaces. There have been recent works on synthesizing solid texture from 2D texture exemplars [Kopf et al. 2007; Takayama et al. 2008; Dong et al. 2008] as well.

Vector field design: Vector fields play an essential role in controlling the appearance of details and textures over surfaces. Methods have been proposed for generating vector fields on planar and 2-manifold domains. Praun et. al. [2000] use radial basis functions to interpolate a sparse set of vector constraints at selected points, while others use hierarchical low pass filtering [Turk 2001], interpolation in tangent spaces [Wei and Levoy 2001], or tools from Discrete Exterior Calculus [Fisher et al. 2007]. Zhang et. al. [2006] design smooth tangent vector fields by allowing precise placement of different types of singularities. Recently, Chen et. al. [2008] develop an interactive technique to design a tensor field which guides the generation of street networks that conform to the tensor field.

Feature based shape perception: The use of feature lines on surfaces has become popular in conveying and depicting 3D shapes. Examples include smooth silhouettes [Hertzmann and Zorin 2000], suggestive contours [Decarlo et al. 2003], geometric ridges and valleys [Ohtake et al. 2004], apparent ridges [Judd et al. 2007], and demarcating curves [Kolomenkin et al. 2008]. Recently, [Cole et al. 2008] provides a statistical analysis of the locations where artists draw lines and the geometric, viewpoint, and lighting characteristics of the underlying 3D scene. On the other hand, the perception of a 3D shape can be significantly affected by texture information. The work of Gorla et. al. [2003] studies the accuracy of judging a shape by the orientation of the texture pattern anisotropy with respect to the principal curvature directions over the shape's surface. The work of [Narain et al. 2007] generates texture along features on continuous flows.

Other related works: The works of [Alliez et al. 2003; Kälberer et al. 2007; Marinov and Kobbelt 2004] propose approaches to generate quad meshes whose quadrangles are aligned with two orthogonal direction fields. Dong et. al. [2006] propose a quad remeshing method which connects extrema of Laplacian eigenfunctions via gradient flow. In the realm of 2D images, the recent work of diffusion curves [Orzan et al. 2008] uses a set of feature lines in an image to represent the original image via Poisson equation, and our study draws much inspiration from this work.

3 Overview

To compute feature-aligned shape textures, our synthesis is guided by two principles. First, we would like the orientation of the synthesized texture to match with the direction of the salient curves, so that the texture "flows" with surface features. Second, to better reveal the shape at the salient curves (e.g., sharp edges in Figure 1), we would like any prominent line features in the texture exemplar to be placed exactly along those curves. Given a 3D surface with extracted salient curves and a texture exemplar, our synthesis proceeds in two fully automated steps, which are illustrated in Figure 2 and detailed in the next two sections:

Vector field generation: We first compute a tangent vector field on the surface that specifies the orientation of the textures, as shown in Figure 2(b). To align the vector field with the direction of salient curves, we propose an incremental algorithm for determining the orientation of the salient curves, and an efficient way to compute a smooth, constrained vector field by energy minimization. The main goal here is to obtain proper curve orientations which would lead to a smooth curve-guided vector field with reduced singularities.

Texture optimization: Next, we compute the texture along the vector field following the patch-based texture optimization framework of [Kwatra et al. 2005; Kwatra et al. 2007]. To align the texture features exactly along the salient surface features, we augment the optimization formulation with an exact feature-to-feature alignment constraint. Figures 2(c) and 2(d) compare the synthesized textures without and with such a constraint.

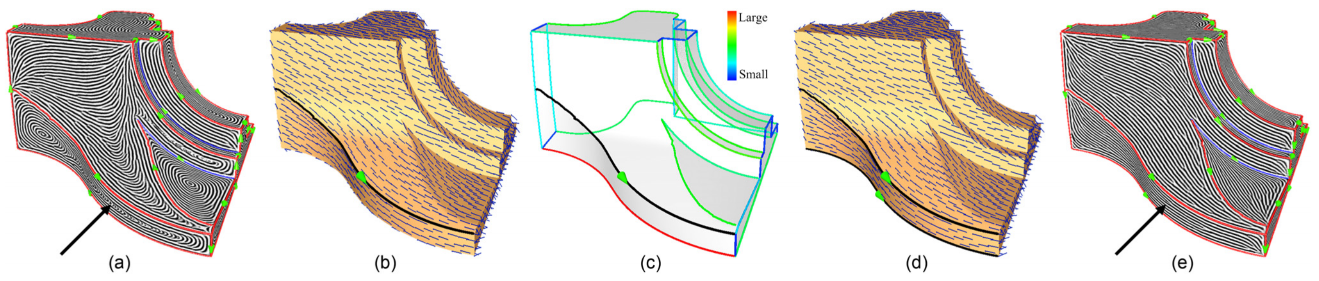

Figure 3: Vector field generated (via harmonic diffusion) from arbitrarily oriented salient curves exhibits undesirable bending and singularities (a). Our orientation algorithm works iteratively where at each step, it computes the vector field from curves that have already been oriented (b), measures the consistency of remaining curves with the field (c), and orients the curve which has the highest consistency score (d). The resulting assignment of curve orientations results in a much smoother field (e) with reduced singularities.

4 Curve-guided vector field generation

Texture synthesis on surfaces typically requires a vector field that determines the local orientation of textures. To obtain continuous, feature-aligned textures, we desire a smooth vector field that follows the direction of the salient curves. Methods capable of computing a smooth vector field that is constrained by user-provided vectors exist [Praun et al. 2000; Fisher et al. 2007]. The main challenge of adopting salient curve constraints in these methods is that these curves are *un-oriented* — a tangent vector on the curve can assume either one of two possible orientations. Arbitrarily oriented tangent vector constraints may produce excessive bending and singularities in the resulting vector field, e.g., see Figure 3(a).

We present, in Section 4.1, an algorithm for determining the orientation of tangent vectors along a set of un-oriented curves on a surface, so as to minimize the bending and singularities in a vector field constrained by these tangent vectors, as shown in Figure 3(e). Our algorithm can work in conjunction with any method that computes a smooth vector field from oriented vector constraints, and further empowers these algorithms to intake un-oriented constraints. In our implementation, we adopted a harmonic-guided diffusion technique for vector field generation, which is simple to implement, efficient to run, and producing the desired fields for subsequent texture synthesis. This will be presented in Section 4.2.

4.1 Curve orientation

Given a set of curves on a 3D surface, each represented as a sequence of vertices connected by edges, we wish to assign an orientation to the tangent vector at each vertex, so that a vector field that interpolates these oriented tangents (using any particular algorithm for vector field generation) is as smooth as possible. One could formulate the problem as a combinatorial optimization task, optimizing some smoothness measure of the vector field over all possible choices of tangent orientations (two at each vertex). However, such a global smoothness measure for a vector field is difficult to formulate. Instead, we found that a greedy algorithm guided by a local, incremental measure works very well in practice, resulting in smooth fields automatically and efficiently.

Our algorithm works incrementally and assigns one orientation to all tangent vectors along a feature curve at a time. The reason that we consider entire curves instead of individual tangent vectors is to avoid undesirable flipping of the orientations along a feature curve, which could happen if individual tangent vectors exhibit significant variations due to, for example, noise or discretization. To identify individual feature curves, we pre-process the salient curves given on the input surface to group close-by, disconnected curve segments along a smooth salient feature and break long curves at high curvature points. Grouping and breaking are controlled by user-specified distance and curvature thresholds, respectively.

We illustrate our algorithm in Figure 3. To start, one feature curve is picked and assigned an arbitrary orientation. We choose the longest feature curve, the black curve in (b), based on the premise that this curve is expected to be the most visually dominant. A smooth vector field, shown by blue vectors, is then computed that interpolates this single curve constraint. At each subsequent step, one un-oriented curve is picked and assigned an orientation, in a way that the vector field interpolating this expanded set of oriented curves introduces least ”distortion” to the vector field that interpolates the current set of oriented curves. This is done by computing a score for each un-oriented curve, as shown in color in (c). The curve with the highest score is picked and oriented, and the vector field is updated to interpolate the newly oriented curve, as in (d). The algorithm repeats until all curves are oriented, as in (e). In hindsight, we can achieve smoothness of the final vector field, which interpolates all the feature curves, if at each step of the algorithm, the intermediate interpolating field is as smooth as possible.

The key to drive our incremental orientation algorithm is a measure of how much “distortion” will need to be introduced to a vector field, which interpolates a set of oriented curves, in order to interpolate an additional oriented curve. Our measure is based on the following intuition. Consider a vector field $\mathbf{u}$ interpolating some oriented vectors, and the field $\mathbf{u}'$ that interpolates an additional vector v_p at vertex p. If v_p coincides with $\mathbf{u}(p)$, the vector at p in the original field $\mathbf{u}$, then $\mathbf{u}'$ would be identical with $\mathbf{u}$. Otherwise, $\mathbf{u}'$ would contain some bending around p in order to accommodate the new constraint v_p. Intuitively, the larger the angle between v_p and $\mathbf{u}(p)$, the more severe the bending would be.

Thus we define the score of an un-oriented curve with respect to a given vector field to reflect the accumulated *consistency* between the curve and the vector field, where each term in the accumulation is inversely related to an angular deviation measured at a point along the curve. Specifically, given a curve c and a vector field $\mathbf{u}$, the consistency score is expressed as

$$g(c, \mathbf{u}) = \max \left\{ \sum_{p \in V(c)} \frac{\pi - \theta_p}{\pi + \theta_p}, \sum_{p \in V(c)} \frac{\theta_p}{2\pi - \theta_p} \right\}. \quad (1)$$

Here $\theta_p \in [0, \pi]$ is the angle between the tangent vector at vertex p along the curve c and the vector in the vector field $\mathbf{u}$ at p, and the denominator normalizes each term of the sum to within $[0, 1]$. The curve c is given an arbitrary orientation when defining $g(c, \mathbf{u})$; in fact, $g(c, \mathbf{u})$ is invariant to the change of orientation of c. With the max operation, we are ensured that the orientation of c that is more consistent with the existing field is chosen to define its score. It is also worth noting that the motivation of using accumulated consistency, instead of other alternatives such as the average, is to favor long curves that are consistent with the vector field.

Limitation Our automatic curve orientation algorithm works well for most 3D models tested, including all examples in this paper. However, its greedy nature may not yield a satisfactory result in some cases. For example, with a set of feature lines radiating out symmetrically from a center (e.g., the spokes of a wheel), our algorithm would attempt to assign the same orientation to two opposite lines, while a more natural, symmetric field would have a singularity at the center and interpolate opposite lines in opposite orientations. Such global information as symmetry has not been considered currently but will be interesting to explore in future work.

4.2 Constrained vector field diffusion

The above curve orientation algorithm works along with any method that generates a smooth vector field from oriented vector constraints, such as [Zhang et al. 2006; Fisher et al. 2007]. Due to the iterative nature of our orientation algorithm, it is desirable to be able to update the vector field at a fast speed. With these requirements in mind, we consider an alternative vector field generation method that is both efficient to run and producing vector fields of comparable quality to these other methods.

Our method is inspired by previous works that consider harmonic functions, whose smoothness is inherited from their physical interpretation as heat diffusion, for data interpolation [Orzan et al. 2008; DeRose and Meyer 2006]. We compute the vector field as three independent scalar functions (one for each x, y, z component), each subject to constraints at a set of vertices. To efficiently compute and update these functions, as needed by our curve orientation algorithm, we consider the penalty method [Xu et al. 2009] to enforce constraints since the implied linear system admits fast computation as well as updating. The final vector field is projected onto the surface tangent planes before texture optimization, as similarly done in [Praun et al. 2000].

Let $\mathbf{x}$ be the x component of the unknown vector field (other components are treated similarly), and $\mathbf{b}$ the x component of the vector constraints. We compute $\mathbf{x}$ as the minimizer of the following energy that combines harmonicity with a quadratic penalty term:

$$\mathbf{x} = \text{argmin}_{\hat{\mathbf{x}}}\{\frac{1}{2}\hat{\mathbf{x}}^{\mathrm{T}}\mathbf{L}\hat{\mathbf{x}} + ||\mathbf{P}^{1/2}(\hat{\mathbf{x}} - \mathbf{b})||^2\}. \tag{2}$$

Here, $\mathbf{L}$ is the cotan Laplacian matrix defined as $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where $\mathbf{W}_{ij} = \frac{1}{2}(\cot\alpha_{ij} + \cot\beta_{ij})$ if vertex pair $\{i, j\}$ forms an edge and α_{ij}, β_{ij} are opposite angles to the edge, and $\mathbf{W}_{ij} = 0$ otherwise, and $\mathbf{D}$ is a diagonal matrix of the row sums of $\mathbf{W}$. The *penalty* matrix $\mathbf{P}$ is a diagonal matrix where $\mathbf{P}_{ii} = \alpha \neq 0$ if and only if there is a constraint vector at vertex i. For the penalty weight, we choose $\alpha = 1.0 \times 10^8$ for all the examples in this paper.

Implementation The minimization can be obtained by solving the linear system $(\mathbf{L} + \mathbf{P})\mathbf{x} = \mathbf{Pb}$ using Cholesky factorization. To avoid re-solving the system at each step of our orientation algorithm, we only *update* the previously solved vector field. To do so, we utilize the efficient super-nodal algorithm [Davis and Hager 2006; Davis 2006], as in [Xu et al. 2009], for updating a sparse Cholesky factorization. The algorithm requires the modifications to the coefficient matrix to be low-rank. This is ensured in our case as the number of added vector constraints each time is much smaller than the total number of vertices in the input mesh.

Comparison Our diffusion-based method generates vector fields with comparable quality to existing methods under the same constraints, as shown in Figure 4 by a comparison to [Zhang et al. 2006] and [Fisher et al. 2007]. When plugged into our curve orientation algorithm for vector field generation, all three methods resulted in the same curve orientations as shown in the figure. On the other hand, our method achieves better efficiency. For this example, each vector field generation step took 0.33 seconds, 0.67 seconds, and over 11 minutes respectively using our method, [Fisher et al. 2007] (using implementation provided by the authors), and [Zhang et al. 2006] (using our own implementation). Note that the latter two methods work directly on tangent vectors on the surface while ours projects non-tangent vectors onto the surface afterwards.

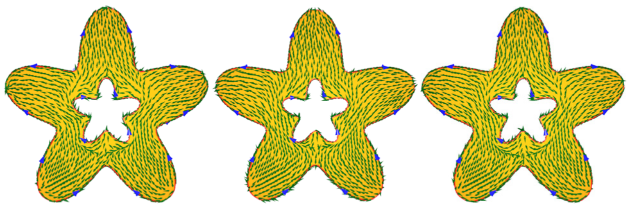
[Fisher et al. 2007]. [Zhang et al. 2006]. Ours.

Figure 4: Smooth vector fields generated by three methods from a given set of oriented curves (orientation shown by blue arrows).

5 Feature-to-feature texture optimization

With a smooth vector field that naturally "flows" with the curve features on the surface, we next synthesize texture patterns guided by the field. Vector field guided synthesis has been previously achieved in an optimization-based framework in both 2D [Kwatra et al. 2005] and 3D [Kwatra et al. 2007; Chen et al. 2009]. We observe that while the result of such un-constrained optimization conforms to the overall surface shape, as in Figure 2(c), when the texture exemplar contains prominent feature lines, such as those in Figure 5(a), being able to align these texture features exactly to surface features would greatly enhance the shape appearance, as can be seen in Figure 2(d). After reviewing the optimization framework for texture synthesis on surfaces, we will introduce our modification in the energy formulation to enforce exact feature-to-feature alignment.

5.1 Texture optimization on surfaces

Given a texture exemplar, the synthesis technique of [Kwatra et al. 2005; Kwatra et al. 2007] seeks colors at points on the output image that minimize the matching error between each local patch on the output image with the most similar patch in the exemplar.

Specifically, denote by Z the input exemplar, and by X the synthesized output on the surface. We minimize the global energy

$$E_t(\mathrm{X}; \{\mathbf{z}_p\}) = \sum_{p \in X^\dagger} \lambda_p \omega_p \left|\mathbf{x}_p - \mathbf{z}_p\right|^2, \tag{3}$$

where $X^\dagger$ is a subset of points in X, $\mathbf{x}_p$ refers to the vectorized colors of the points in a local grid-patch around the surface point $p \in X^\dagger$, $\mathbf{z}_p$ refers to the vectorized neighborhood in Z whose appearance is most similar to $\mathbf{x}_p$, and ω_p is set as in Section 3.1 of [Kwatra et al. 2005]. In the un-constrained scenario, weights λ_p are set to be 1. These notations are illustrated in Figure 5(b,c). The minimization is solved by iterating between searching for the best matching exemplar neighborhoods $\mathbf{z}_p$ for each $\mathbf{x}_p$ and minimizing E_t with respect to X using a linear system.

The energy is minimized over a subset $X^\dagger$ of mesh vertices. Given the size w of the local grid-patch, the vertices $\mathbf{x}_p$ in $X^\dagger$ should be carefully selected so that all the local grid-patches of $\mathbf{x}_p$ should together cover the whole surface. Therefore, we have to ensure that the distance between neighboring sample vertices in $X^\dagger$ is less than w. We follow the sampling method in [Chen et al. 2009] to obtain $X^\dagger$. Specifically, we start by uniformly sampling the vertices using distance rw along the salient curves, where r is a user-chosen parameter in $[0, 1]$ to control the sample spacing. Then we select the vertices near the salient curves and finally randomly select the other vertices in-between the curves so that the distance between two neighboring vertices is less than rw. The parameter r is chosen empirically as $r = 0.25$.

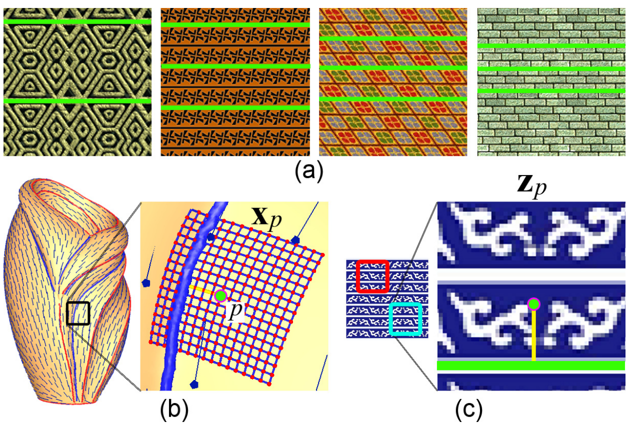

Figure 5: (a) Texture patterns containing prominent feature lines (green). (b-c) Notations for describing patch-based optimization.

5.2 Feature-to-feature constrained optimization

We next constrain the texture optimization above to enforce exact alignment between texture features and surface features. Given a texture exemplar, we extract the prominent features in the exemplar as piecewise linear segments using a heuristic approach [Wu and Yu 2004], optionally allowing user sketching. When more than one feature lines are detected, we select one as the principal feature line. Example feature lines are shown in Figure 5(a).

To achieve feature alignment in the above patch-matching paradigm, we enforce those patches on the surface that are centered at points near salient curves to match with only those patches in the texture exemplar that are centered at pixels near feature lines. That is, we constrain the texture matching for those points $p \in X^\dagger$ which are close to the salient curves. Specifically, consider the set of all salient curves $\mathcal{C}$ on the surface, and denote $d(p, \mathcal{C})$ the closest geodesic distance from a point $p \in X^\dagger$ to the salient curves. If $d(p, \mathcal{C}) < w/2$, which means that the local patch $\mathbf{x}_p$ at point p covers some feature line, then we consider only those $\mathbf{z}_p$ whose center pixel's distance to the nearest texture feature differs from $d(p, \mathcal{C})$ by less than a user-specified threshold when searching for the most similar patch $\mathbf{z}_p$ in the texture exemplar to the surface patch $\mathbf{x}_p$. As an example, the patch with a blue outline in the exemplar in Figure 5(c) will be considered when matching with the surface patch in (b), while the patch with a red outline in the exemplar will not. To ensure small matching error near salient curves, we re-define the weights in Equation (3) as

$$\lambda_p = 1/(d(p, \mathcal{C}) + \epsilon_0), \tag{4}$$

where $\epsilon_0 = 1.0 \times 10^{-6}$ is a small number to avoid division by zero.

6 Results

We show examples demonstrating our feature-aligned texturing technique on a range of models. In all examples, we adopt the algorithm and available implementation of Yoshizawa et al. [2005] for fast and robust detection of crest lines on meshes. Given an input model, its crest lines, and a user-selected texture exemplar, texturing is then performed fully automatically.

Figure 6 compares textures synthesized on the same model guided by vector fields constrained by feature curves with arbitrary orientations (a) and orientations computed using our incremental algorithm (b). Observe that the distortions and singularities are much less severe in (b) than in (a). Similar observations can be made in Figure 3. Note that even with proper curve orientation, singularities are not entirely avoidable. However, our orientation algorithm tends to push the remaining singularities to "corners" where two or more feature curves meet, as shown in Figure 3(e). These corner singularities appear to have less visual impact compared to those occurring over a flat region, e.g., compare with Figure 3(a).

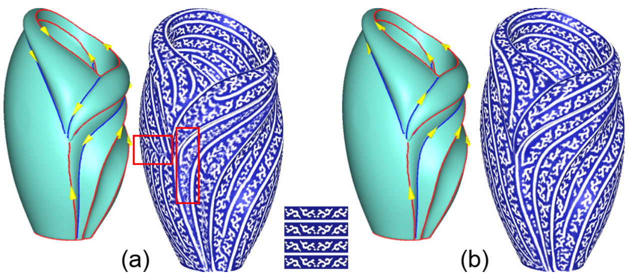

Figure 6: Texturing of a surface with arbitrary (a) vs. incrementally computed (b) orientation of feature curves. Curve orientations are shown by green arrows (top). Note the singularities indicated in (a).

To further demonstrate the effectiveness of our texturing technique in decorating and enhancing shape appearance, we show a gallery of examples in Figure 7. For each model, we compare between texturing results synthesized without and with feature alignment. Observe that the latter present a more natural fit to the surface and better reveal the geometry or structure of the shapes by enhancing the appearance of their salient feature curves. In Figure 8, we show texturing aligned with multiple levels of feature curves on the Bimba model. One easily observes the clearer depiction of the mouth at a low feature resolution and further around the eyes at a high feature resolution, due to alignment of texture patterns with features identified over those regions.

Performance Our tests are performed on a 2.5GHz Intel Core 2 Duo PC with 2GB of RAM. Using fast harmonic field update, as discussed in Section 4.2, vector field generation is performed under one minute for a model with 400K triangles, where updating of the vector field at each step of curve orientation takes about 1 second. The optimization-based texture synthesis step currently takes about 20 minutes for a model at a similar size. We anticipate that this performance can be substantially improved by performing synthesis at multiple mesh resolutions, as successfully demonstrated in [Kwatra et al. 2007].

7 Conclusion and future work

The essence of a 3D shape is well reflected by its salient feature curves and the essence of a *textured* shape is precisely revealed only when its feature lines exactly align with those in the associated texture. We achieve the latter with a novel and fully automatic technique combining orientation computation of salient feature curves, curve-guided harmonic vector diffusion, and example-based texture synthesis with exact feature-to-feature alignment.

There are a number of venues of future work. On the technical side, our curve orientation algorithm can be further improved by considering global structure of the surface features, such as symmetry. Our un-optimized texture synthesis implementation could also benefit from hierarchical techniques [Kwatra et al. 2007] and GPU implementations [Huang et al. 2007]. On the perceptual end, it would be most interesting and informative to conduct user studies on the effectiveness of texturing that exploits shape features.

Acknowledgments

We would first like to thank the anonymous reviewers for their helpful comments. We are also grateful to Matthew Fisher for sharing his code on vector field generation, to Renjie Chen for providing his codes on texture synthesis, and to Xiaokun Wu for his help with making the video. All the 3D models used in this paper are from the AIM@SHAPE shape repository. This work

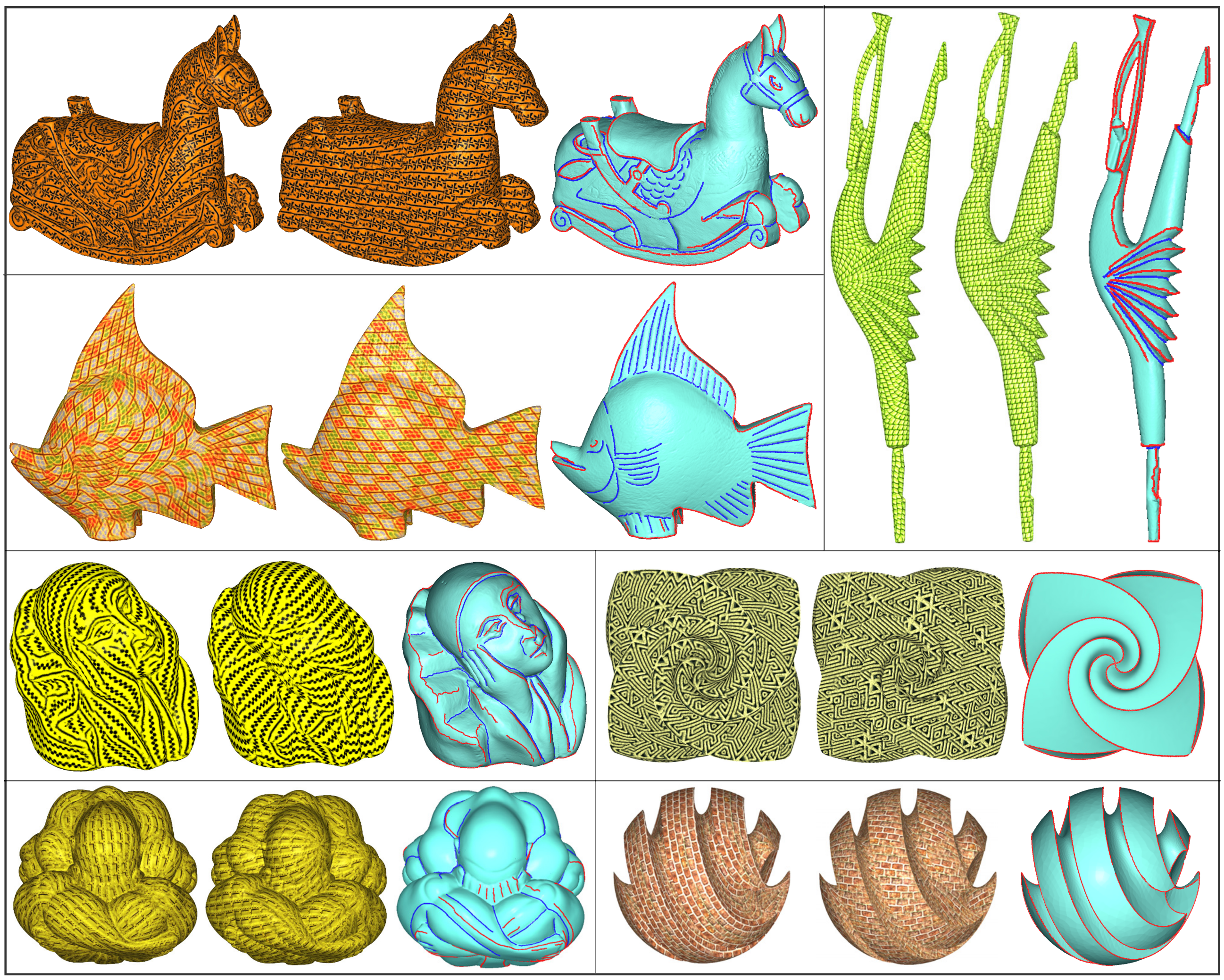
Figure 7: A gallery of feature-aligned texturing results. In each triplet, from left to right: our texturing result, textures synthesized by a smooth vector field but without feature alignment, and model showing the extracted feature curves (blue: valley lines; red: ridge lines).

is supported in part by grants from NSERC (No. 611370), NSF (No. CCF-0702662), the Israeli Ministry of Science, the Israel Science Foundation, the 973 National Key Basic Research Foundation of China (No. 2009CB320801), the 863 Program of China (No. 2007AA01Z313), NSFC (No. 60773022), the China Scholarship Council and the joint grant of the NSFC and Microsoft Research Asia (No. 60776799).

References

ALLIEZ, P., COHEN-STEINER, D., DEVILLERS, O., LEVY, B., AND DESBRUN, M. 2003. Anisotropic polygonal remeshing. *ACM Trans. on Graphics 22*, 3, 485–493.

CHEN, G., ESCH, G., WONKA, P., MÜLLER, P., AND ZHANG, E. 2008. Interactive procedural street modeling. *ACM Trans. on Graphics 27*, 3, 1–10.

CHEN, R., LIU, L., AND DONG, G. 2009. Local resampling for patch-based texture synthesis in vector fields. *International Journal of Computer Applications in Technology*, to appear.

COHEN, M., SHADE, J., HILLER, S., , AND DEUSSEN, O. 2003. Wang tiles for image and texture generation. *ACM Trans. on Graphics 22*, 3, 287–294.

COLE, F., GOLOVINSKIY, A., LIMPAECHER, A., BARROS, H. S., FINKELSTEIN, A., FUNKHOUSER, T., AND RUSINKIEWICZ, S. 2008. Where do people draw lines. *ACM Trans. on Graphics 27*, 3, 88:1–11.

DAVIS, T. A., AND HAGER, W. W. 2006. Modifying a sparse Cholesky factorization. *SIAM Journal on Matrix Analysis and Applications 20*, 3, 606–627.

DAVIS, T. A. 2006. User guide for CHOLMOD. Tech. rep., University of Florida.

DECARLO, D., FINKLSTEIN, A., RUSINKIEWICZ, S., AND SANTELLA, A. 2003. Suggestive contours for conveying shape. *ACM Trans. on Graphics 22*, 3, 848–855.

DEROSE, T., AND MEYER, M. 2006. Harmonic coordinates. Tech. rep., Pixar Technical Memo #06-02.

DONG, S., BREMER, P.-T., GARLAND, M., PASCUCCI, V., AND HART, J. 2006. Spectral surface quadrangulation. *ACM Trans. on Graphics 24*, 3, 1057–1066.

DONG, Y., LEFEBVRE, S., TONG, X., AND DRETTAKIS, G. 2008. Lazy solid texture synthesis. In *Computer Graphics Forum*

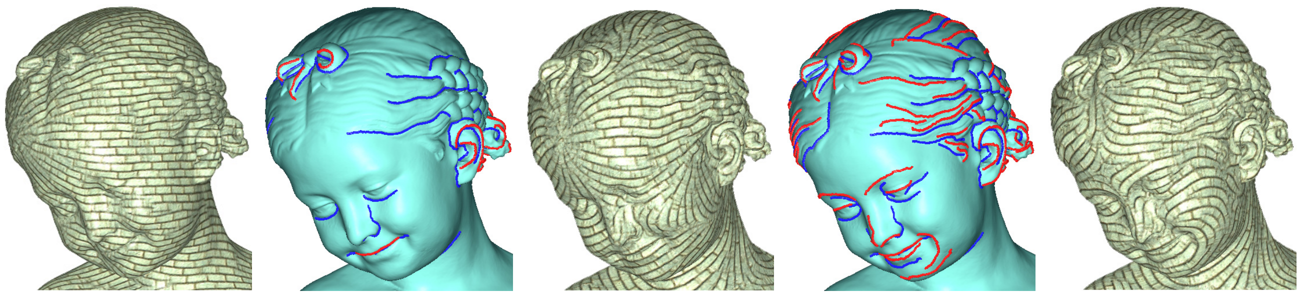
(a) (b) (c) (d) (e)

Figure 8: Influence of feature set on feature-aligned texturing. (a) No feature-alignment. (b)-(c) Lower feature resolution aligns the texture to larger features such as the mouth, but leads to a glaring singularity in the forehead. (d)-(e) Higher feature resolution (additional feature curves detected) leads to a more satisfying result and the aligned texture reveals finer features such as the eyes.

(Proc. of the Eurographics Symposium on Rendering), vol. 27, 1165–1174.

FISHER, M., SCHRÖDER, P., DESBRUN, M., AND HOPPE, H. 2007. Design of tangent vector fields. *ACM Trans. on Graphics 26*, 3, 56:1–9.

FU, C.-W., AND LEUNG, M.-K. 2005. Texture tiling on arbitrary topological surfaces. In *Proc. of Eurographics Symposium on Rendering*, 99–104.

GORLA, G., INTERRANTE, V., AND SAPIRO, G. 2003. Texture synthesis for 3D shape representation. *IEEE Trans. Vis. & Comp. Graphics 9*, 4, 512–524.

HAN, C., RISSER, E., RAMAMOORTHI, R., AND GRINSPUN, E. 2008. Multiscale texture synthesis. *ACM Trans. on Graphics 27*, 3, 51:1–8.

HERTZMANN, A., AND ZORIN, D. 2000. Illustrating smooth surfaces. *ACM Trans. on Graphics 19*, 3, 517–526.

HUANG, H., TONG, X., AND WANG, W. 2007. Hardware accelerated parallel texture optimization. *Journal of Computer Science and Technology 22*, 5, 761–769.

JUDD, T., DURAND, F., AND ADELSON, E. 2007. Apparent ridges for line drawing. *ACM Trans. on Graphics 26*, 3, 19:1–7.

KÄLBERER, F., NIESER, M., AND POLTHIER, K. 2007. Quadcover - surface parameterization using branched coverings. *Computer Graphics Forum 26*, 3, 375–384.

KOLOMENKIN, M., SHIMSHONI, I., AND TAL, A. 2008. Demarcating curves for shape illustration. *ACM Trans. on Graphics 27*, 5, 157:1–9.

KOPF, J., FU, C., COHEN-OR, D., DEUSSEN, O., LISCHINSKI, D., AND WONG, T. 2007. Solid texture synthesis from 2D exemplars. *ACM Trans. on Graphics 26*, 3, 2:1–9.

KWATRA, V., ESSA, I., BOBICK, A., AND KWATRA, A. 2005. Texture optimization for example-based synthesis. *ACM Trans. on Graphics 24*, 4, 795–802.

KWATRA, V., ADALSTEINSSON, D., KIM, T., KWATRA, N., CARLSON, M., AND LIN, M. 2007. Texture fluids. *IEEE Trans. Vis. & Comp. Graphics 13*, 5, 939–952.

MARINOV, M., AND KOBBELT, L. 2004. Direct anisotropic quad-dominant remeshing. In *Proc. of Pacific Graphics*, 207–216.

NARAIN, R., KWATRA, V., LEE, H.-P., KIM, T., CARLSON, M., AND LIN, M. 2007. Feature-guided dynamic texture synthesis on continuous flows. In *Proc. of Eurographics Symposium on Rendering*, 361–370.

OHTAKE, Y., BELYAEV, A., AND SEIDEL, H.-P. 2004. Ridge-valley lines on meshes via implicit surface fitting. *ACM Trans. on Graphics 23*, 3, 609–612.

ORZAN, A., BOUSSEAU, A., WINNEMÖLLER, H., BARLA, P., THOLLOT, J., AND SALESIN, D. 2008. Diffusion curves: A vector representation for smooth-shaded images. *ACM Trans. on Graphics 27*, 3, 92:1–8.

PRAUN, E., FINKELSTEIN, A., AND HOPPE, H. 2000. Lapped textures. *ACM Trans. on Graphics 19*, 3, 465–470.

TAKAYAMA, K., OKABE, M., IJIRI, T., AND IGARASHI, T. 2008. Lapped solid textures: filling a model with anisotropic textures. *ACM Trans. on Graphics 27*, 3, 53:1–8.

TURK, G. 2001. Texture synthesis on surfaces. *ACM Trans. on Graphics 20*, 3, 347–354.

WEI, L.-Y., AND LEVOY, M. 2001. Texture synthesis over arbitrary manifold surfaces. *ACM Trans. on Graphics 20*, 3, 355–360.

WEI, L., HAN, J., ZHOU, K., BAO, H., GUO, B., AND SHUM, H. 2008. Inverse texture synthesis. *ACM Trans. on Graphics 27*, 5, 52:1–9.

WEI, L.-Y., LEFEBVRE, S., KWATRA, V., AND TURK, G. 2009. State of the art in example-based texture synthesis. *Eurographics 2009 State-of-the-art Report*.

WEXLER, Y., SHECHTMAN, E., AND IRANI, M. 2007. Space-time completion of video. *IEEE Trans. Pat. Ana. & Mach. Int. 29*, 3, 463–476.

WU, Q., AND YU, Y. 2004. Feature matching and deformation for texture synthesis. *ACM Trans. on Graphics 23*, 3, 364–367.

XU, K., ZHANG, H., COHEN-OR, D., AND XIONG, Y. 2009. Dynamic harmonic fields for surface processing. *Computers and Graphics (Special Issue of SMI 2009) 33*, 3, 391–398.

YOSHIZAWA, S., BELYAEV, A., AND SEIDEL, H.-P. 2005. Fast and robust detection of crest lines on meshes. In *Proc. of ACM Symposium on Solid and Physical Modeling*, 227–232.

ZHANG, E., MISCHAIKOW, K., AND TURK, G. 2006. Vector field design on surfaces. *ACM Trans. on Graphics 25*, 4, 1294–1326.