Parallel Iso/Aniso-scale Surface Texturing Guided in Gabor Space

Bin Sheng*
Shanghai Jiao Tong University

Hanqiu Sun†
The Chinese University of Hong Kong

Yubao Wu‡
The Chinese University of Hong Kong

Daniel Thalmann§
EPFL& Nanyang Technological University

Abstract

This paper presents a parallel texture synthesis over arbitrary surfaces, generating consistent and spatially-varying visual appearances. A novel scaling field is represented to measure the geometry-aware appearance or geometric deformation, therefore the generated textures locally agree with the geometric structure and maintain the coherence during shape deformation. And we compute the Gabor feature space for the 2D exemplars, to determine the k-coherence candidate pixels. Such Gabor feature space is not only low dimensional but capturing multi-resolution texture structures, hence the k-coherence matching guided by Gabor space improves the performance and quality of pixel similarity measurement. We directly apply multi-pass correction for each vertex according to its local neighborhood for order-independent purpose. Experimental results demonstrate that our method produces significantly improved surface textures with parallel performance.

CR Categories: I.3.3 [Computer Graphics]: Three-Dimensional Graphics and Realism—Display Algorithms I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture;

Keywords: example-based synthesis, parallel processing, Gabor space, surface texture synthesis

1 Introduction

Textures, usually investigated in regular 2D grids and 3D volumes, does demonstrate its essential role in current rendering techniques only when it is mapped or synthesized on geometric surfaces. Surface texture enables augmenting geometric models with increased realism and visual richness. Over the past decade years, texture synthesis on surfaces has significantly increased the ease of designing complex image-based details on arbitrary meshes. A first category of algorithms achieved such a synthesis based on per-pixel non-parametric sampling[Turk 2001; Wei and Levoy 2001; Tong et al. 2002; Zelinka and Garland 2003; Ashikhmin 2001]. Other relevant texture synthesis techniques include an hybrid approach combining pixel-based and patch-based schemes [Nealen and Alexa 2003], texturing objects in photographs [Fang and Hart 2004], parallel controllable texture synthesis on GPU [Lefebvre and Hoppe 2005; Lefebvre and Hoppe 2006] which use coherent synthesis and and appearance space. Han at al. [Han et al. 2006] propose an expectation-maximization (EM) solver to compute surface texture with GPU implementation. Most previous methods extract neighbor grids by locally flattening and resampling surfaces. The major concerns/differences of these methods concentrate on how to measure and minimize the distortion in local parameterization [Zhou et al. 2005].

Despite the problem of texturing arbitrary surfaces has been extensively studied, real-time and parallel performance remains one of the major bottlenecks for interactive generation. With the overview of state-of-art techniques,we find that interactive use of multiple texture exemplars for surface decoration is still a labor-intensive task, as surface textures can often be overwhelming to compute and manipulate comparing to 2D texture images. Our proposed approach is simple and parallelizable, which allows spatially-varying texture patterns to be seamlessly synthesized as the texture mixture with smooth transition or platter. Such flexibility further allows artists to control the synthesized surface textures and effectively express their design intentions. The experimental results demonstrate the ease of use and reliable surface texture results provided by our system with GPU implementation.

Our parallel texture synthesis approach contribute the state-of-art techniques in the following aspects: firstly, the texture patterns are directly synthesized on the 3D surfaces without cumbersome manual operations and global parametrization. Gabor space is to capture texel's multiresolution characteristics in k-coherence matching, leading to structure-preserving synthesis. Secondly, the texture patterns are synthesized in coherence with the shape characteristics and deformation correspondence, due to the iso/aniso-scaling field-guided synthesis of texture coordinates. finally, the surface synthesis pipeline is fully order-independent, therefore it can be naturally parallelized and implemented on GPU, as shown in Figure 1.

2 Gabor Space

The similarity between a synthesized neighborhood and an exemplar neighborhood is typically measured by feature distance. Using such a measurement to find similar texture exemplars for a given synthesized surface neighborhood is crucial to improving the visual quality of surface textures. Previous methods have matched a neighborhood by measuring distance in either a standard or reduced-dimensional space [Lefebvre and Hoppe 2006]. The multiresolution character of exemplar pixels is, however, difficult to capture using SSD (Sum-of-Squared-Difference) distances in the texture's spatial neighborhood.

Here we introduce the feature space created by Gabor filtering, commonly known as Gabor feature space, as Gabor filters unify the frequency and multiresolution spatial analysis of texture features. Adopting the classification techniques of [Manjunath and Ma 1996], we use four scales four scales ($S = 4$) and six orientations ($K = 6$). The Gabor feature vector is therefore expressed as

$$T = [\mu_{0,0}\sigma_{0,1}\mu_{0,1}...\mu_{3,5}\sigma_{3,5}] \quad (1)$$

In our surface texture synthesis, a bank of Gabor filters is used to filter the luminance Y channel of exemplar textures. We then project

*e-mail: shengbin@sjtu.edu.cn
†e-mail:hanqiu@cse.cuhk.edu.hk
‡e-mail:wuyubao5@gmail.com
§e-mail: danielthalmann@ntu.edu.sg

SIGGRAPH Asia 2013, November 19 – 22, 2013, Hong Kong.

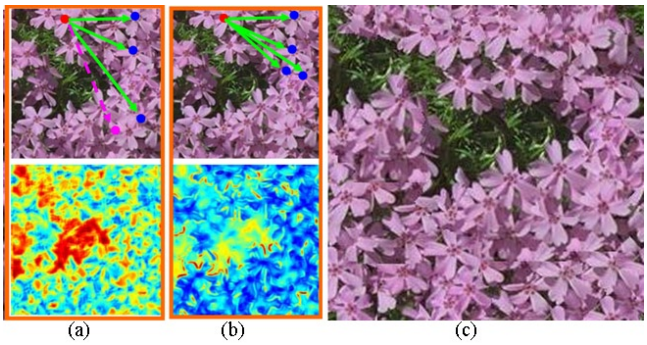

Figure 1: *Pixel similarity for constructing k-coherence similarity sets: k-coherence similarity set in (a) appearance space and (b) Gabor feature space; (c) the 2D synthesis result using Gabor space.*

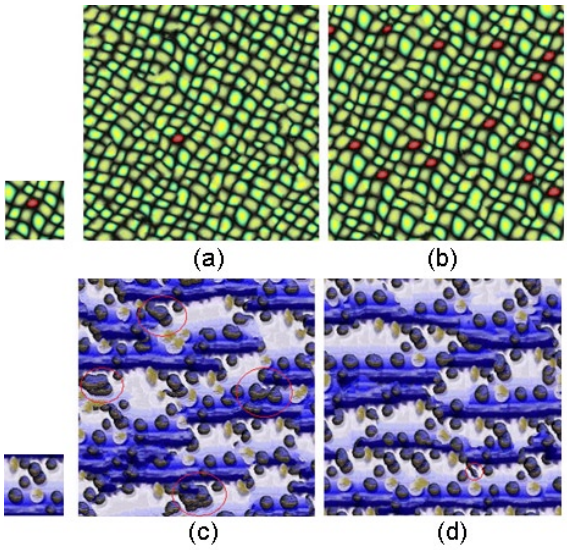

Figure 2: *2D synthesis quality comparisons: (a) results using the space defined by [Ashikhmin 2001]; (c) results using appearance space; (b) and (d) results using our Gabor feature space.*

this 48 dimensional space into a 6-dimensional subspace $(F_i)_{i=1..6}$ using PCA. Thus, the Gabor feature space for synthesis can be represented as a 9-dimensional space (R, G, B, F_i), where $i = 1...6$. Note that the low-dimensional Gabor feature space is generated for the exemplar texture as a precomputation.

In order to accelerate runtime synthesis, we have applied k-coherence searching based on Gabor feature distance, due to its constant and accelerated processing time per search [Tong et al. 2002]. The k-coherence method constructs a candidate-set for each exemplar pixel, containing k pixels with neighborhoods similar to the input pixels. These sets tend to be quite accurate, given the superiority of Gabor filters in feature and texture structure discrimination, compared to SSD measurement in texture spatial space. In Figure 1, the similarity measurement in Gabor space finds four similar pixels on the other flower centers in Figure 1b. The merit of our approach is that the synthesis results properly capture and preserve texture structure and features, as shown in Figure 2 and Figure 3.

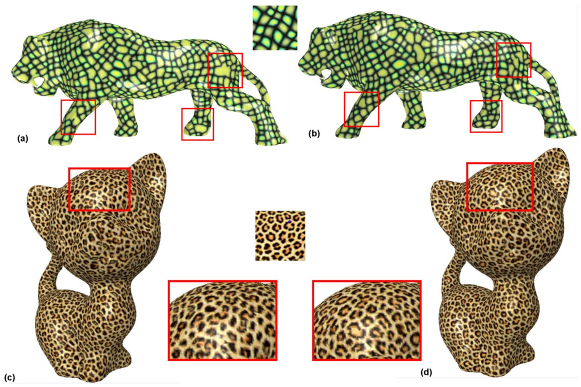

Figure 3: *Surface texturing results using k-coherence neighborhood matching in appearance space (a and c), and Gabor feature space (b and d).*

3 Iso-scale Synthesis on Surfaces

Since we synthesize textures as vertex colors directly on the target mesh surface, our synthesis primitives are vertices rather than texture pixels. In our approach, we build a multi-resolution pyramid for the surface mesh, and a Gaussian stack for the exemplar texture. We then apply traditional hierarchical texture synthesis, creating S_0, S_1,....,$(S_L = S)$ from lower to higher resolutions. At the initial (coarsest) resolution, we can just randomly assign texture coordinates to mesh vertices, without loss of fidelity. For each pyramid level above this, we will also perform three parallel operations on each vertex: *upsampling*, *jittering*, and *correction.*

We initialize the synthesis during the upsampling step, in which the vertex in the child mesh is directly assigned the texture coordinates of its parent vertex's texture coordinates in the coarser mesh level. To introduce spatially deterministic randomness, we can also apply the jittering function $J_l(i)$ to perturb the upsampled coordinates at each mesh level. This function is parameterized by a hash function $\mathcal{H} : \mathbb{Z} \rightarrow [-1, +1]^2$ and a user-specified per-level randomness value $r_l \in [0, 1]$.

The core of surface texture synthesis is a parallel correction of all local surface patches. The correction process consists of three steps: (1) For each surface vertex i, we gather the feature vectors of the 5×5 neighboring sample points on the surface patch N_i, The feature vector of each sample point is a 9×25dimensional Gabor feature vector. (2) We re-sample vertex i's 6-nearest neighboring vertices, denoted as $V_p(p = 1...6)$, onto the regular grid of surface patch N_i. Thus, the local offset coordinates relative to vertex i for these neighbors can be denoted by $N_i(V_p)_{p=1...6}$, or more formally, $N_i(V_p) = (\vec{u_i}\vec{v_i})^T \mathbf{V_p}$. (3) To facilitate K-coherence search [Tong et al. 2002], we precompute the candidate set $KC^l_{1...kc}(V_p)$ of kc exemplar pixels with similar Gabor feature vectors in the exemplar image at current Gaussian stake level l, (with kc experimentally set to 4). Then, during run-time synthesis, we find the pixels $E^l_{1...kc}(V_p)$ in the exemplar texture E^l. These pixels are the coherence candidates of i corresponding to $KC^l_{1...kc}(V_p)$ respectively. Their coordinates are computed as the candidate $KC^l_{1...kc}(V_p)$ offset by V_ps local resampled coordinates $N_i(V_p)$.

In principle, these synthesis steps should be processed in the same kernel of a CUDA structure. Pseudo-code for parallel execution of the above corrections is as follows:

Parallel Corrections on GPU

Calculate Coordinates Of 2-ring Neighbor Vertices;

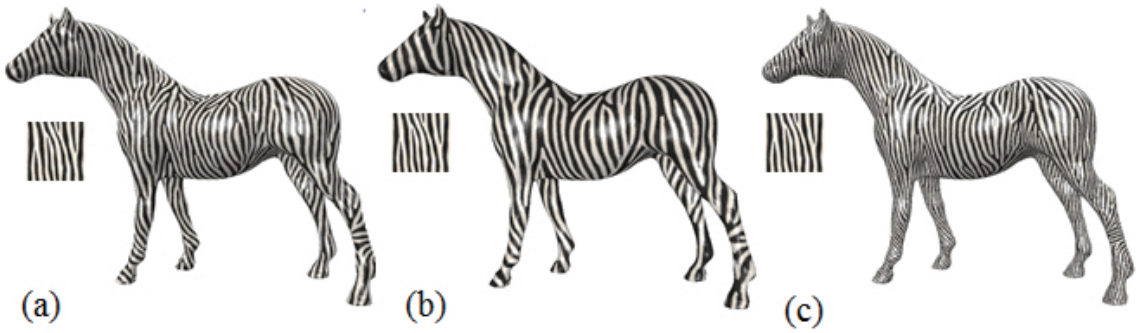

Figure 4: *Surface texture synthesis results with different scale fields: (a) iso-scale with $D = 1.0$, (b) iso-scale with $D = 0.6$, (c) aniso-scale with varying scale ($D = 0.62.5$).*

for Correction Iteration $Correction = 1$ to 8 **do**
  for K-Coherence number $kc = 1$ to 8 **do**
    Gather K-Coherence Candidates (Correction, kc);
    for Gabor Feature Channel $F_i = 1$ to 6 **do**
      Calculate Feature Distances (Correction, kc, F_i);
    end for
    Update Minimum Distance Candidate (Correction, kc);
  end for
  Gather Self-Similar Candidates;
  for Gabor Feature Channel $F_i = 1$ to 6 **do**
    Calculate Feature Distances (Correction, F_i);
  end for
  Update Minimum Distance Candidate ($Correction$);
end for

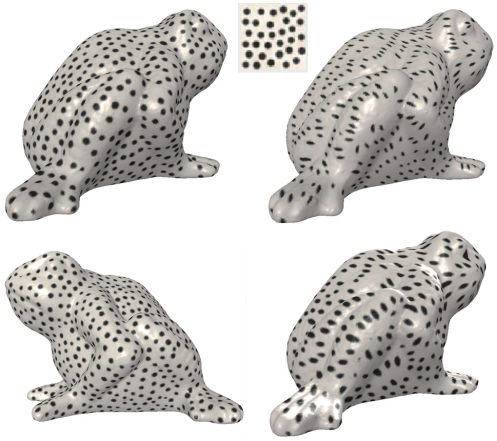

Figure 5: *(Left) iso-scale surface texture synthesis; (right) aniso-scale surface texture synthesis (top: $D_u = 1.0$, $D_v = 2.0$; middle: $D_u = 0.7$, $D_v = 1.4$; bottom: $D_u = 0.8$, $D_v = 1.2$).*

4 Aniso-scale Synthesis on Surfaces

We can anisometrically synthesize the spatially-variant texture over an arbitrary surface. While the Jacobian field J can be straightforwardly evaluated for a regular or parametric 2D domain [Lefebvre and Hoppe 2006], its definition on a curved surface domain without global parameterization is ambiguous with respect to the rotations of the surface vector field [Praun et al. 2000].

We implement the Jacobian transformation by warping the surface vertex neighborhood $N_i(\Delta)$. . Specifically, we define the aniso-scaling field $D_s = \binom{D_u}{D_v}$ and express the Jacobian transformed offset coordinates as an aniso-scaling offset:

$$N_i'(\Delta) = Jacobian_i^{-1} N_i(\Delta) = \begin{pmatrix} D_u(i) N_i(\Delta)_u \\ D_v(i) N_i(\Delta)_v \end{pmatrix} \quad (2)$$

Typically, if $D_u(i) = D_v(i)$ for each vertex i, the scaling offset becomes isometric, allowing the user to generate spatially-varying textures by interactively setting the variational scaling field, as shown in Figure 4. Figure 6 shows the iso-scale and aniso-scale synthesis results with a uniform scale of 1.3, and a texel scale that is the same for the head, feet, and body of the horse model. Using aniso-scale surface texture synthesis, the size of the texture can be reduced smoothly from head to feet, while patterns along the body grow larger.

5 Experimental Results and Summary

The proposed approach is developed on a 2.6GHz Intel Core(TM) 2.13Ghz CPU, together with nVidia GPU GTX285 and 1G RAM. We experiment a variety of input textures and output mesh models with $40,000 - 500,000$ vertices. Figure 7 shows the examples of parallel surface texturing results. Our approach can accurately capture and simply control the visual coherence of the examplar texture and the synthesized surface texturing. The original model and its deformed shape are synthesized for the same iso-scale texturing effect. To avoid distortions, we calculate the scale field to measure the deformation degree of the local surfaces. Our method can keep the visual similarity between the example image texture and the synthesized surface texture.

In this paper, we present a new surface texture synthesis framework that is adapted from parallel texture synthesis-from-sample methods to directly synthesize texture on a polygonal surface, creating the geometry-coherent texture appearance without the need for extra parametrization. Also, we introduce a Gabor feature space for capturing the texture appearance characteristics, leading to structure-preserving synthesis. In the future, we would like the users to control the texture orientation by specifying different kinds of vector fields on the surfaces.

Acknowledgment

The authors would like to thank all reviewers for their helpful suggestions and constructive comments. The 3D models used in this paper are from the shape repositories of AIM@SHAPE and Stanford. The work is supported by the National Natural Science Foundation of China (No. 61202154,61133009), Shanghai Pujiang Pro-

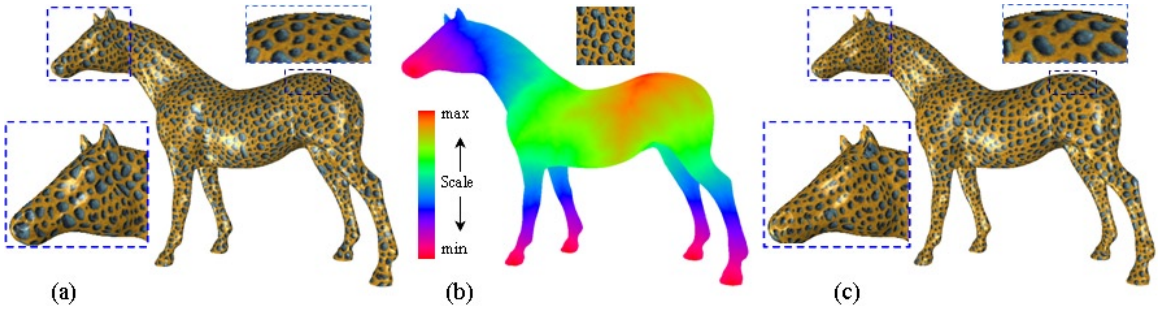

Figure 6: *Aniso-scale surface synthesis results: (a) isometric synthesis results, scale = 1.3; (b) scale fields and the example texture image, the scale field variants progressively from the head and feet regions to the horseback region; (c) anisometric synthesis results, scale varies (1.0–1.6).*

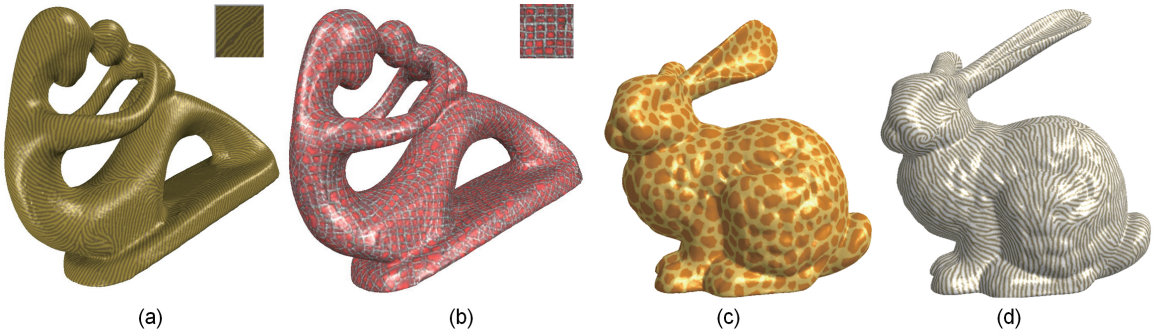

Figure 7: *Surface texturing of deformed surfaces with various exemplars. The scale-preserving surface texturing of deformable models is synthesized with the surface scale-field generated from the deformation measurement.*

gram (No. 13PJ1404500), and the National Basic Research Project of China (No. 2011CB302203), the Open Projects Program of National Laboratory of Pattern Recognition, RGC grant no. 416311, UGC direct grants for research (no.2050454, 2050485), and the Open Project Program of the State Key Lab of CAD&CG (Grant No. A1206), Zhejiang University.

References

ASHIKHMIN, M. 2001. Synthesizing natural textures. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, ACM, 217–226.

FANG, H., AND HART, J. 2004. Textureshop: texture synthesis as a photograph editing tool. *ACM Transactions on Graphics (TOG) 23*, 3, 354–359.

HAN, J., ZHOU, K., WEI, L., GONG, M., BAO, H., ZHANG, X., AND GUO, B. 2006. Fast example-based surface texture synthesis via discrete optimization. *The Visual Computer 22*, 9, 918–925.

LEFEBVRE, S., AND HOPPE, H. 2005. Parallel controllable texture synthesis. *ACM Transactions on Graphics (TOG) 24*, 3, 777–786.

LEFEBVRE, S., AND HOPPE, H. 2006. Appearance-space texture synthesis. In *ACM SIGGRAPH 2006 Papers*, ACM, 548.

MANJUNATH, B. S., AND MA, W.-Y. 1996. Texture features for browsing and retrieval of image data. *IEEE Trans. Pattern Anal. Mach. Intell. 18*, 8, 837–842.

NEALEN, A., AND ALEXA, M. 2003. Hybrid texture synthesis. In *Proceedings of the 14th Eurographics workshop on Rendering*, Eurographics Association, 105.

PRAUN, E., FINKELSTEIN, A., AND HOPPE, H. 2000. Lapped textures. In *Proceedings of SIGGRAPH 2000*, Citeseer, 465–470.

TONG, X., ZHANG, J., LIU, L., WANG, X., GUO, B., AND SHUM, H. 2002. Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Transactions on Graphics 21*, 3, 665–672.

TURK, G. 2001. Texture synthesis on surfaces. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM New York, NY, USA, 347–354.

WEI, L., AND LEVOY, M. 2001. Texture synthesis over arbitrary manifold surfaces. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM New York, NY, USA, 355–360.

ZELINKA, S., AND GARLAND, M. 2003. Interactive texture synthesis on surfaces using jump maps. In *Proceedings of the 14th Eurographics workshop on Rendering*, Eurographics Association Aire-la-Ville, Switzerland, Switzerland, 90–96.

ZHOU, K., WANG, X., TONG, Y., DESBRUN, M., GUO, B., AND SHUM, H. 2005. Texture-Montage: Seamless texturing of surfaces from multiple images. *ACM Trans. Graph.(Proc. of SIGGRAPH)*, 1148–1155.