Synthesis of Progressively-Variant Textures on Arbitrary Surfaces

Jingdan Zhang* Tsinghua Univ.
Kun Zhou MSR Asia
Luiz Velho† IMPA
Baining Guo MSR Asia
Heung-Yeung Shum MSR Asia

Abstract

We present an approach for decorating surfaces with *progressively-variant textures*. Unlike a homogeneous texture, a progressively-variant texture can model local texture variations, including the scale, orientation, color, and shape variations of texture elements. We describe techniques for modeling progressively-variant textures in 2D as well as for synthesizing them over surfaces. For 2D texture modeling, our *feature-based warping* technique allows the user to control the shape variations of texture elements, making it possible to capture complex texture variations such as those seen in animal coat patterns. In addition, our *feature-based blending* technique can create a smooth transition between two given homogeneous textures, with progressive changes of both shapes and colors of texture elements. For synthesizing textures over surfaces, the biggest challenge is that the synthesized texture elements tend to break apart as they progressively vary. To address this issue, we propose an algorithm based on texton masks, which mark most prominent texture elements in the 2D texture sample. By leveraging the power of texton masks, our algorithm can maintain the integrity of the synthesized texture elements on the target surface.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism–color, shading, shadowing, and texture; I.2.10 [Artificial Intelligence]: Vision and Scene Understanding–texture; I.3.3 [Computer Graphics]: Picture/Image Generation.

Keywords: Texture synthesis, texture mapping, surfaces

1 Introduction

Textures are extensively used to increase the realism of surfaces. Recently, a number of techniques including [Turk 2001; Wei and Levoy 2001; Ying et al. 2001; Gorla et al. 2001; Soler et al. 2002] have been developed for synthesizing textures on arbitrary surfaces. Given a sample texture, these techniques can create a similar texture that fits the target surface naturally and seamlessly.

Most of the previous work on surface texture synthesis concentrated on homogeneous textures. These textures are stationary in that they are characterized by stationary stochastic models. Homogeneous textures, however, only account for a limited class of real-world textures. Many textures, including the coating patterns of various animals such as the tiger, cannot be described by stationary models [Turk 1991; Tonietto and Walter 2002]. These patterns exhibit complex variations resulting from a biological growth process [Walter et al. 2001]. Nonetheless, these patterns have a well defined structure. Intuitively, their texture elements change in a progressive fashion. The texture is stationary in a small neighborhood around each point, but the overall texture characteristics vary continuously over the texture domain. We call such textures progressively-variant textures. Figure 1 shows a progressively-variant texture mimicking that of a leopard [Kingdon 1977].

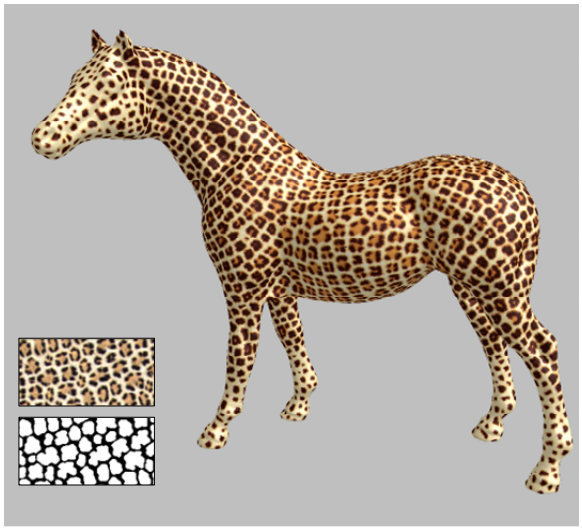


Figure 1: A progressively-variant texture mimicking that of a leopard. Notice the non-trivial progressive transition from the rosette patterns on the main body to the spots on the legs. In the lower left corner we show the original homogeneous texture sample and a texton mask.

In this paper, we present an approach for decorating arbitrary surfaces with progressively-variant textures. The key to harnessing the modeling power these textures is placing the local variations under user control. With this goal in mind we developed techniques for modeling progressively-variant textures in 2D and for synthesizing such textures on surfaces.

For 2D texture modeling, we introduce techniques for creating progressively-variant textures from homogeneous texture samples. The user can control the scale, orientation, color, and shape variations of texture elements at different levels. At the pixel level, our field distortion synthesis allows the user to control the scale and orientation variations of texture elements. This is similar to [Tonietto and Walter 2002], but with the additional ability of orientation control. At the feature level, our feature-based warping provides user control of shape variations of texture elements. This control is important for capturing complex variations such as those seen in animal coat patterns, yet no existing techniques support such control. We also introduce feature-based blending, which uses a

*This research was done when Jingdan Zhang was an intern at Microsoft Research Asia (MSR Asia). http://research.microsoft.com/ig/

†This research was done when Luiz Velho was visiting MSR Asia.

blend texture to provide a smooth transition (i.e., smooth changes of both shape and color of texture elements) between two given homogeneous textures. In many ways, our feature-based techniques are similar to feature-based morphing for images [Beier and Neely 1992]. With our techniques, user-specified features are represented by a texton mask, which is a labeling map that marks the prominent texture elements.

For synthesizing progressively-variant textures onto surfaces, the biggest challenge is that the synthesized texture elements tend to break apart as they progressively change. This is a serious problem since the main advantage of progressively-variant textures is the smooth variation of texture elements, and this advantage will be lost if texture elements break apart as they vary. To prevent breaking, we propose an algorithm which synthesizes a texton mask in conjunction with the target texture. The texton masks we use have extremely sparse histograms (e.g., binary images), making them easier to synthesize by existing techniques such as [Efros and Leung 1999; Wei and Levoy 2000]. Leveraging the power of a texton mask so synthesized, we can maintain the integrity of texture elements on the target surface.

The rest of the paper is structured as follows: Section 2 reviews related work. Section 3 gives an overview of the pipeline for creating progressively-variant textures on surfaces. Section 4 describes 2D texture modeling techniques. Section 5 discusses texture synthesis over surfaces. Section 6 presents results. Section 7 concludes the paper with suggestions for future work. Finally, Appendixes A and B in the CD-ROM provide additional experimental results.

2 Related Work

Several algorithms have been proposed for synthesizing homogeneous textures on surfaces, including [Gorla et al. 2001; Turk 2001; Wei and Levoy 2001; Ying et al. 2001]. The synthesis quality these algorithms depends on the performance of the underlying non-parametric sampling techniques [Efros and Leung 1999; Wei and Levoy 2000]. Ashikhmin [2001] pointed out a special type of textures, called "natural textures", that cannot be synthesized well by [Efros and Leung 1999; Wei and Levoy 2000]. He noted that the L2-norm they used is a poor measure for perceptual similarity and proposed a special-purpose algorithm for "natural textures". Building on [Ashikhmin 2001], Ying et al. [2001] presented an algorithm for synthesizing textures on surfaces. Hertzmann et al. [2001] combined [Wei and Levoy 2000; Ashikhmin 2001] to get the benefits of both.

Non-parametric sampling can also be done at patch level as in [Efros and Freeman 2001; Liang et al. 2001]. Soler et al. [2002] extended this approach for synthesizing textures on surfaces.

Dischler et al. [2002] proposed a technique for generating textures on surfaces. They first extract "texture particles" from the sample texture by color thresholding and then paste "texture particles" onto surfaces. They also showed an example of changing the scale of texture particles.

The algorithms reviewed so far are designed for homogeneous textures, which are usually described by stationary Markov random field (MRF) models (e.g., see [Zhu et al. 1998]). One way to create textures with local variations is through chemical or biological simulations. Turk [1991] generated textures on surfaces using reaction-diffusion differential equations. To change the size of the spots or stripes on an animal coat, he varied the diffusion rates on the target surfaces. Witkin and Kass [1991] obtained complex spatial variations in reaction-diffusion patterns by space-varying diffusion. Like other procedural textures, reaction-diffusion textures are only suitable for modeling certain textures, and much parameter tweaking is necessary to achieve a desired result. Recently, Walter et al. [2001] proposed a technique for generating mammalian coat patterns by biological simulation. Compared with the simulation-based techniques, our approach can produce a wider class of textures. In addition, our approach usually generates more realistic textures because of our use of real images as texture samples.

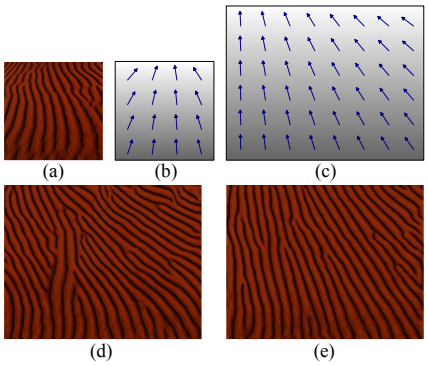


Figure 2: The need for input orientation field. (a) Input texture. (b) Transition function and orientation field on the input texture. (c) Transition function and orientation field on the output texture. (d) Synthesis result without input orientation field. Notice that the orientation of texture elements do not follow the desired orientation field in (c). (e) Synthesis result with input orientation field.

We consider a texture progressively-variant if texture characteristics change smoothly over the texture domain. More specifically, at each point of the texture domain there should be a neighborhood over which the texture is stationary. Hertzmann et al. [2001], Harrison [2001], and Zalesny et al. [2002] studied a different type of non-stationary texture, which is piecewise stationary in the sense that the texture domain can be divided into patches and textures are stationary on individual patches. Fig. 16 in [Hertzmann et al. 2001] touches upon the theme of progressive variation but with no attention paid to the variation of texture elements. Progressively variant textures are related to locally-stationary stochastic processes [Mallat et al. 1998]. Research in this recent area has been limited to 1D processes and at present there is no universally accepted definition of local stationarity [Mallat 2003].

Our texton mask is similar to the texton channel proposed by Malik et al. [1999] and the texton map presented by Guo et al. [2001]. Unlike texton channels and texton maps which are extracted by visual learning, our texton map is interactively specified by the user. We experimented with texton channels but found texton masks more suitable for our goal. Texton masks are easy to specify and they provide a lot of user control.

3 Overview

We represent a progressively-variant texture by a tuple (T, M, F, V) with a texture image T and three user-specified control channels. The *texton mask* M is a labeling map that marks the prominent texture elements in T. At each pixel p, the texton map indicates which type of texture elements p belongs to. The *orientation field* is a vector field defining a unit vector at each pixel of T, whereas the *transition function* is a continuous scalar function F whose gradient determines how fast the texture T is changing. Fig. 1 contains a texture image and its texton mask. Fig. 2 (a) and (b) show a texture with its transition function and orientation field.

For creating a progressively-variant texture in 2D, our input consists of a homogeneous texture sample and user-specified texton mask M, transition function F and orientation field V. The specification of F and V is similar to the familiar task of specifying vector fields for synthesizing (homogeneous) textures on surfaces [Turk 2001]). Specification of texton mask T is based on a simple but

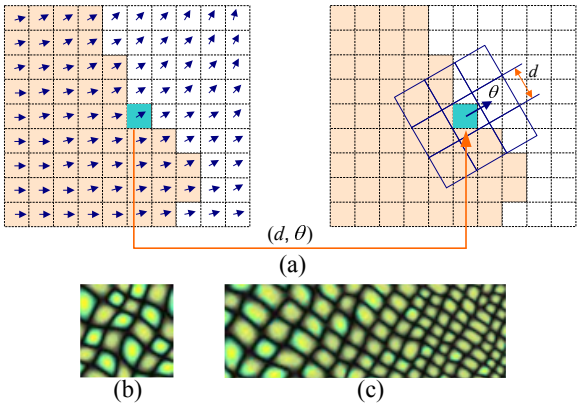


Figure 3: (a) Field distortion synthesis. (b) Homogeneous texture sample. (c) Synthesis result. The green blobs are made progressively smaller towards the right.

effective color thresholding method described in Section 4.

From the above input, a progressively-variant 2D texture can be created by our field distortion or feature-based techniques. The field distortion algorithm generates a texture by scaling and rotating the local coordinate frame at each pixel according to the transition function and orientation field. The key to feature-based techniques is to create texton masks first, which then guide the synthesis of the target textures by providing a rough sketch of the distribution and shapes of the synthesized texture elements. Specifically, the feature-based warping algorithm first warps the texton mask and then uses that to synthesize a new texture. The feature-based blending algorithm takes two homogeneous textures with texton masks as input and generates first a blend texton mask and then a blend texture.

To synthesize a progressively-variant texture on a mesh, we start with a 2D progressively-variant texture sample (T_o, M_o, F_o, V_o). The user needs to specify a transition function F_s and orientation field V_s on the target mesh [Turk 2001]. On the mesh, the synthesis algorithm controls the scale and orientation variation of texture elements by matching F_s and V_s with their 2D counterparts. Most importantly, our algorithm synthesizes a texton mask M_s in conjunction with the target texture T_s and uses M_s to prevent the breaking of texture elements.

4 2D Texture Modeling

Techniques for modeling progressively-variant 2D textures from homogeneous texture samples include field distortion synthesis and feature-based techniques.

4.1 Field Distortion Synthesis

The field distortion algorithm synthesizes a progressively-variant texture T_o from a homogeneous sample texture by controlling scale and orientation variations of the texture elements in T_o. For texture elements in T_o to change size and orientation smoothly, the user needs to supply a continuous transition function F_o and a continuous orientation field V_o of the size of T_o. The user specifies scale and orientation vectors at a few locations. Our system then automatically interpolates these "key" scales and orientations to generate the entire F_o and V_o by using radial basis functions. This technique is widely used in previous work for generating vector fields on surfaces (e.g., in [Praun et al. 2000]).

Figure 3 (a) illustrates the synthesis process. Essentially the field distortion algorithm extends [Wei and Levoy 2000] by incorporating scale and orientation variations controlled F_o and V_o. The algorithm synthesizes T_o pixel-by-pixel. To synthesize a pixel p, the algorithm first finds all neighborhoods in the sample texture that are similar to p's neighborhood $N(p)$ and then chooses the neighborhood that is the most similar, taking its center to be the newly synthesized pixel p. F_o and V_o control the target texture through the construction of the neighborhood $N(p)$. As shown in Figure 3 (a), the pixels in $N(p)$ are not of the same size as the pixels of the target texture T_o; instead they are scaled by the scalar $d = F_o(p)$. In addition, $N(p)$ is oriented according to the vector $V_o(p)$. The pixels in $N(p)$ are resampled from that of T_o. In our implementation, we compute a pixel in $N(p)$ from the four nearest pixels in T_o by bilinear interpolation. This is a simple approximation that works well for a wide range of scales and orientations. Ideally, the resampling should be weighted by the coverage of $N(p)$'s pixels in T_o. However, computing pixel coverage is a costly operation and should be avoided whenever possible.

The synthesis order has a large effect on the synthesis quality. An order established as in [Turk 2001] generates the best results. Note also that the self-similarity based texture editing [Brooks and Dodgson 2002] can be easily adapted for interactive editing of progressively-variant textures.

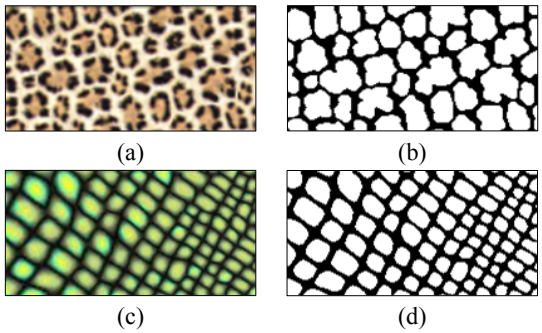


Figure 4: Specifying texton masks by color thresholding. (a) and (c): Textures. (b) and (d) Texton masks. In both cases the texton mask is binary. Specifying a texton mask takes a few seconds.

4.2 Texton Mask Specification

To apply feature-based techniques, the user must specify a texton mask on a given texture. The goal of texton masks is to mark prominent features or texture elements. Texton mask construction may be regarded as an image segmentation problem. Fully automatic segmentation of images remains a challenging problem (e.g., see [Malik et al. 1999]). We do not attempt to solve this problem; instead we build a user interface that allows the user to mark prominent texture elements easily. Our experiences suggest that a texton mask indicating one or two types of the most prominent texture elements is sufficient for modeling and synthesis of progressively-variant textures.

Our user interface is based on color thresholding. The user picks one or two pixel colors and for each color a threshold for color differences. The texture is then partitioned accordingly. Fig. 4 provides examples of texton mask specification. Similar techniques are used for the "magic wand" tool in Adobe Photoshop and by Dischler et al. [2002]. This is a simple and effective technique for textures in which meaningful patterns can be discriminated by colors. We found texton masks produced by this technique work well for most textures, mainly for the reason that our feature-based techniques and surface texture synthesis algorithm have very low requirements for texton masks and are not sensitive to errors in texton masks. More sophisticated segmentation methods such as [Leung and Malik 1996] can be used to generate better texton masks

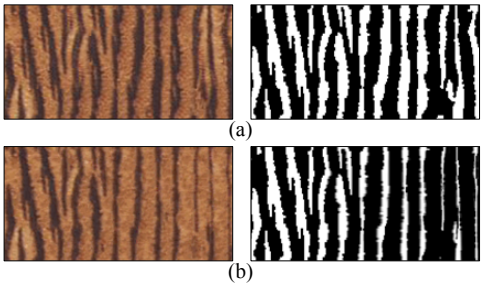


Figure 5: Feature-based warping. (a) A homogeneous tiger skin texture and its texton mask. (b) Result after warping. The stripes are made progressively thinner towards the right.

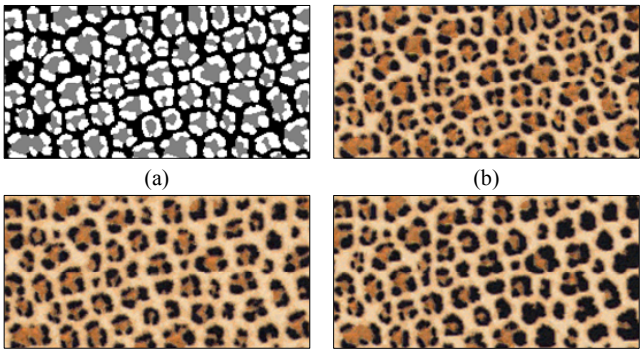


Figure 6: Another example of feature-based warping. (a) Texton mask of the texture in (b). This mask has three colors (black, white, and grey). (b) A homogeneous leopard skin texture. (c) Warping result of (b) by progressively shrinking the white and grey areas together. (d) Warping of (c) by expanding the white areas.

when necessary.

In addition to color thresholding, our user interface also supports morphological operations such as dilation and erosion for refining texton masks.

4.3 Feature-Based Warping and Blending

Warping: To apply this technique, the user first uses a texton mask M_i to mark the features or texture elements in the input texture T_i and then performs editing operations on this mask, producing a new mask M_o. The transition function F_o controls the parameters in the editing operations to achieve a smooth progressive variation of patterns in the mask M_o. Finally, our system synthesizes a progressively-variant texture T_o using two texton masks, M_i and M_o, and known texture T_i.

The synthesis of the texture T_o can be formulated as an application of image analogies [Hertzmann et al. 2001]. Specifically, we want to find an analogous texture T_o that relates to the new texton mask M_o the same way as original texture T_i relates to its texton mask M_i. Using the terminology of image analogies, T_i and T_o are filtered results of the texton masks M_i and M_o respectively. For this reason, we refer to the step of creating T_o from M_i, M_o, and T_i as texton mask filtering. Texton mask filtering bears some resemblance to the texture-by-numbers technique in [Hertzmann et al. 2001]. The latter technique synthesizes a non-homogeneous texture consisting of patches of homogeneous textures from an input non-homogeneous texture that has been segmented into homogeneous patches. The main difference is that texton mask filtering uses fine-grain partitions of the textures (down to the texture element level).

A variety of editing operations can be applied to the original texton mask M_i to generate a desired target mask M_o. The texton masks we use have few colors: All masks used in this paper have fewer than four colors and usually the mask is binary. For this reason we can easily apply morphological operations such as dilation, erosion, and their combinations. We can also apply image warping techniques such as mesh warping, field warping, and warping using radial basis functions [Gomes et al. 1998]. These techniques often require feature points and feature lines to be specified. We found [Suzuki and Abe 1985] very useful for this purpose. As shown in Fig. 4 (b) and (d), a texton mask often has isolated patterns repeating over the mask. We use [Suzuki and Abe 1985] to extract the boundary contours of these patterns. For image warping, these contours can be used as feature lines and the contour centers as feature points.

Fig. 5 shows an example of feature-based warping. We warp the texton mask by progressively shrinking the stripes horizontally but not vertically. Fig. 6 provides another example, in which we generate a progressively-variant texture mimicking the complex variation of texture elements as seen on a leopard skin [Kingdon 1977]. The specific transition we try to capture in this example is that from the rosette patterns on the animal's main body to the spots on its legs.

Blending: This technique takes two homogeneous textures T_0 and T_1 as input and generates a progressively-variant texture T_b that provides a smooth transition from T_0 to T_1 as shown in Fig. 7. For simplicity, we assume T_0, T_1, and T_b are all of the same size and are defined on the unit square $[0,1]\times[0,1]$. We also use a simple linear transition function $F_b(x,y)=x$ defined on $[0,1]\times[0,1]$. Finally, we use binary texton masks M_0 and M_1 for T_0 and T_1, respectively.

Suppose that we can construct the texton mask M_b of the blend texture T_b. Then we can obtain T_b by color blending two textures T_0' and T_1' according to the transition function F_b. T_0' is synthesized from two texton masks, M_0 and M_b, and texture T_0 by using texton mask filtering. T_1' is synthesized from M_1 and M_b, and texture T_1 in the same way. Because T_0' and T_1' share the same texton mask M_b, features in T_0' and T_1' are aligned and thus the color blending of T_0' and T_1' will not cause "ghosting" [Gomes et al. 1998].

The key to generating T_0' and T_1' is the construction of M_b. This can be done in a few different ways. A simple technique is the following. Ideally, we want $M_b(x,y)$ to be like M_0 when $x\approx 0$ and like M_1 when $x\approx 1$. To achieve this goal, we interpolate 2D binary functions M_0 and M_1 in 3D and take a diagonal 2D slice, obtaining $M(x,y)=xM_1(x,y)+(1-x)M_0(x,y)$. $M(x,y)$ behaves like M_0 when $x\approx 0$ and like M_1 when $x\approx 1$. Then we Gaussian blur $M(x,y)$ to smooth out the discontinuity inherited from M_1 and M_2. Finally, we convert $M(x,y)$ to the binary mask M_b using a user-selected threshold. Fig. 7 (middle row) is a blending example generated this way. Note that simply averaging the texton masks, e.g., by setting $M(x,y)=\frac{1}{2}(M_1(x,y)+M_0(x,y))$, will generate inferior blending results, as Fig. 7 (bottom row) demonstrates (e.g., pay attention to the discontinuities at the border of the blend texture). Fig. 8 analyzes the construction of M_b for Fig. 7.

It is also possible to construct M_b by applying Wei's texture mixture algorithm [Wei 2001] to texton masks M_0 and M_1, although none of examples in this paper was created this way. Wei designed his algorithm for direct application to textures. Unfortunately this algorithm often fails to produce good blend textures. See Fig. 7 (top row) for an example. For blending two texton masks, the performance of Wei's algorithm is comparable to the simple algorithm described above. An advantage of using Wei's algorithm is that it can be used to blend any number of textures.

In addition to Wei's texture mixture algorithm, there are other texture mixture approaches such as [Bar-Joseph et al. 2001]. These general texture mixture methods often focus on mixing texture elements instead of their progressive variation. In Appendix B in the CD-ROM, we also compare our feature-based blending with an-

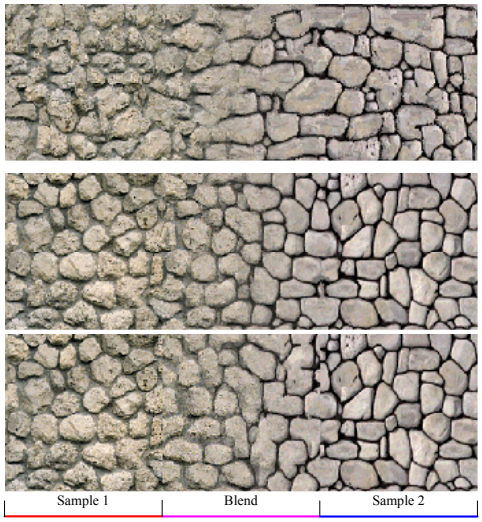


Figure 7: Feature-based blending. Top row: Result by Wei's texture mixture algorithm. For the middle and bottom rows, the input textures are on the left and right and the blend texture in the middle. Middle row: Our result. Notice the progressive transition of both the shape and color of texture elements. Bottom row: Blending result using the average mask. For the middle and bottom examples, the borders of the homogeneous textures and the blend texture are marked by the color-coded line segments at the bottom.

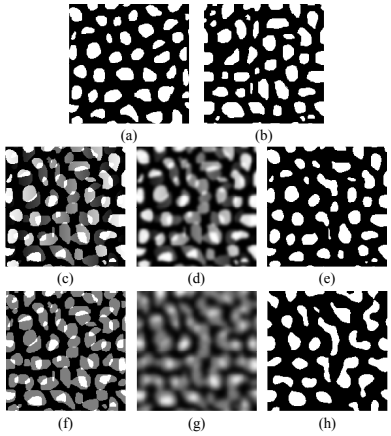


Figure 8: Generating texton mask M_b for the example in Fig. 7. Top row: Texton masks M_0 and M_1 of the left and right textures, respectively. Middle row: Generating M_b with our technique. Images (c), (d), and (e) are $M(x,y)$, Gaussian-blurred $M(x,y)$, and M_b. Notice that the left part of M_b looks like M_0 whereas the right part of M_b looks like M_1. This is not the case for the corresponding images in the bottom row, in which M_b is generated by simple averaging.

other related approach called pattern-based texture morphing [Liu et al. 2002].

5 Surface Texture Synthesis

Given a surface mesh and a progressively-variant 2D texture T_o with transition function F_o, orientation field V_o, and texton mask M_o, we wish to synthesize a progressively-variant texture T_s on the mesh. As in the case of homogeneous texture synthesis, the user needs to specify an orientation field V_s on the mesh by providing the orientations at some key locations. For progressively-variant textures, the

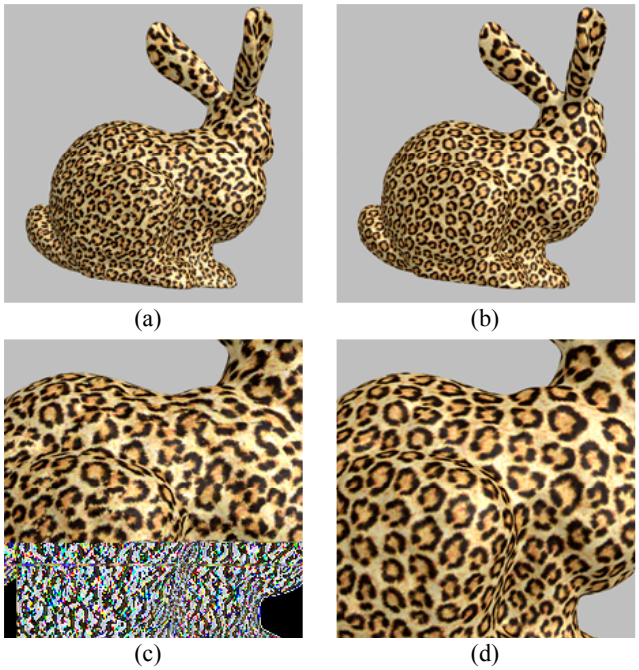


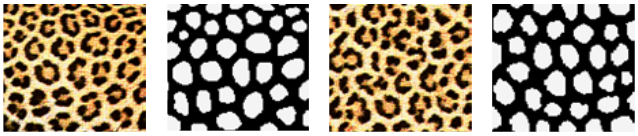
Figure 9: (a) Texture synthesis without texton masks. (b) Synthesis result of our algorithm, which is based on texton masks. (c) Zoomed view of (a). (d) Zoomed view of (b). The progressively-variant 2D texture sample is generated from the homogeneous sample shown in Fig. 10 (a).

user must also specify a transition function F_s on the mesh in a similar fashion. From the user-specified values, our system interpolates the entire V_s and F_s using Gaussian radial basis functions, where the radius is the distance over the mesh as in [Praun et al. 2000].

5.1 Using Texton Masks

Our synthesis algorithm is based on texton masks. In a progressively variant texture synthesized without texton masks, texture elements tend to break apart as Fig. 9 (a) demonstrates. The breaking of texture elements is caused by the fact that the standard L2-norm is a poor perceptual measure for neighborhood similarity, which is at the heart of all texture synthesis algorithms following the non-parametric sampling approach of [Efros and Leung 1999; Wei and Levoy 2000]. Ideally, a good perceptual measure should account for the fact that the human visual system is most sensitive to edges, corners, and other high-level features in images. The simple L2-norm cannot account for this fact and the associated synthesis process often smoothes out the edges in the output texture, leading to breaking and re-mixing of texture elements. Ashikhmin [2001] noted this behavior and proposed a coherence-based synthesis algorithm to address this problem for a special class of textures. In general, due to our incomplete understanding of the human visual system at present, there is no perceptual measure available that is reliable and computationally efficient enough for texture synthesis.

To prevent the breaking of texture elements, our algorithm synthesizes a texton mask M_s in conjunction with the texture T_s. This algorithm is based on two ideas. First, texton masks are resistant to damage caused by deficiencies in the L2-norm. Specifically, the non-parametric sampling of [Efros and Leung 1999] and [Wei and Levoy 2000] can only generate pixel values that already exist in the input texture and thus cannot smooth out edges when the input texture has few colors, which is the case for texton masks. Fig. 10 demonstrates this idea with an example. Our second idea is to leverage the damage resisting property of texton masks. By synthesizing



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

Figure 10: Texton masks are resistant to the damages caused by deficiencies in the L2-norm. (a) Input texture. (b) Texton mask of (a). (c) A patch of the synthesized texture. Many texture elements break apart. (d) The corresponding texton mask synthesized from (b). No texture element breaks apart. The same synthesis procedure is used for (c) and (d).

a texton mask on the target surface, we can use the mask to guide the texture synthesis process by encouraging it to pick, for each mesh vertex v, the color of a pixel p in the 2D texture sample such that v and p have the same texton mask value.

5.2 The Algorithm

Our algorithm synthesizes the texton mask M_s along with the target texture T_s on the mesh. A single-resolution and single-pass version of our algorithm proceeds as follows. We make a pass through the vertices of the mesh and pick a color and a texton mask value for every vertex. Following [Turk 2001], we use the orientation field V_s to determine the processing order of mesh vertices. We found that this technique noticeably improves the synthesis quality. At each vertex v, the color is obtained through the following steps. First, we construct neighborhoods $N_c(v)$ and $N_m(v)$ in the tangent plane of the surface at v. Within the tangent plane, $N_c(v)$ and $N_m(v)$ have the same orientation, which is determined by the surface orientation field value $V_s(v)$. The neighborhood $N_c(v)$ contains color values resampled from that of the vertices near v. Likewise, the neighborhood $N_m(v)$ holds texton mask values resampled from that of the vertices near v.

In the next step, we collect all candidate pixels for vertex v in the 2D texture T_o and put them in the candidate pool $C(v,\varepsilon)$. To qualify for $C(v,\varepsilon)$, a candidate pixel p must satisfy the condition

$$|F_o(p) - F_s(v)| < \varepsilon, \tag{1}$$

where ε is a prescribed tolerance for mismatch in transition function values. For all the examples in this paper, we set $\varepsilon = 0.1$ for transition function values normalized to lie in the interval $[0,1]$.

Finally, we carry out two searches, one for texton mask value $M_s(v)$ and the other for vertex color $T_s(v)$. For $M_s(v)$, we find a pixel p in $C(v,\varepsilon)$ such that the distance between the neighborhoods $N_m(v)$ and $N_m(p)$ is the smallest, where $N_m(p)$ is a neighborhood of p in the 2D texton mask M_o. Searching for $T_s(v)$ is slightly more complex. We need to look for a pixel p in the candidate pool $C(v,\varepsilon)$ such that the following sum of two distances

$$dist(N_c(v),N_c(p)) + dist(N_m(v),N_m(p))$$

is the smallest, where $N_c(p)$ is a neighborhood of p in the 2D texture T_o. The neighborhoods $N_c(p)$ and $N_m(p)$ have the same orientation, which is determined by the 2D orientation field value $V_o(p)$.

The pseudo-code of our algorithm is as follows.

For each vertex v on surface
  construct neighborhoods $N_c(v)$ and $N_m(v)$
  build candidate pool $C(v,\varepsilon)$
  smallest_match = INFTY
  For each pixel $p = (a,b)$ in $C(v,\varepsilon)$
    construct neighborhoods $N_m(p)$
    new_match = $dist(N_m(v),N_m(p))$
    If (*new_match* $<$ *smallest_match*)
      smallest_match = *new_match*
      mask_value = $M_o(p)$
  $M_s(v)$ = *mask_value*
  smallest_match = INFTY
  For each pixel $p = (a,b)$ in $C(v,\varepsilon)$
    construct neighborhoods $N_c(p)$ and $N_m(p)$
    new_match = $dist(N_c(v),N_c(p))$ + $dist(N_m(v),N_m(p))$
    If (*new_match* $<$ *smallest_match*)
      smallest_match = *new_match*
      color = $T_o(p)$
  $T_s(v)$ = *color*

As in [Wei and Levoy 2001; Turk 2001], our system uses two-pass multi-resolution synthesis to improve the synthesis quality. In addition, we perform a refinement pass with a small neighborhood size. This refinement pass is very fast and noticeably improves synthesis results.

Neighborhood Construction: In our implementation, we choose larger (i.e., having more pixels) neighborhoods $N_m(v)$ and $N_m(p)$ when we search for the texton mask value at v. When we search for the color value at v, we choose a smaller neighborhood size for both the color neighborhoods $N_c(v)$ and $N_c(p)$ and the texton mask neighborhoods $N_m(v)$ and $N_m(p)$. The rationale behind this choice is that texton masks determine the layout of texture elements whereas the synthesis of pixel colors is simply a step to fill in the details.

Another technical issue is the local adaptation of the size of the neighborhood $N_c(p)$. One would expect that as texture elements change sizes, the neighborhood used to capture them should also change. This is indeed the case and $N_c(p)$ should really be $N_c(p,s)$ where $s = F_o(p)$ is the scale at p. Let $N_c(p,s_{min})$ be the smallest neighborhood and $N_c(p,s_{max})$ be the largest. We determine the size of $N_c(p,s)$ by linearly interpolating between that of $N_c(p,s_{min})$ and $N_c(p,s_{max})$ and rounding the result up to the nearest integer. The same local adaptation technique applies to neighborhoods $N_m(p)$, $N_c(v)$ and $N_m(v)$.

Matching Transition Functions: The synthesis algorithm requires that we match the transition function values when searching for the color and texton mask value of a vertex v. We fulfill this requirement by confining the search to the candidate pool $C(v,\varepsilon)$. This approach is very different from most existing algorithms for synthesizing stationary textures on surfaces, as these algorithms simply search all pixels of the 2D sample texture T_o. An advantage of searching all pixels is that hierarchical data techniques such as kd-trees and tree-structured vector quantization (TSVQ) can be pre-computed and used to accelerate the search. Unfortunately, the pixel p found by the search may not satisfy the condition in Equation (1). This is a price we cannot afford to pay for progressively-variant textures.

In our algorithm, the matching of transition function values is guaranteed since a qualified candidate p in $C(v,\varepsilon)$ is required to satisfy the condition in Equation (1). With $C(v,\varepsilon)$ so constructed, we can no longer use kd-trees or TSVQ to accelerate the search because $C(v,\varepsilon)$ changes from vertex to vertex, making pre-computation of kd-trees and TSVQ impossible. To address this problem, we search $C(v,\varepsilon)$ using the k-coherence technique proposed by [Tong et al. 2002]. According to the standard k-coherence criteria, $C(v,\varepsilon)$ should be populated with pixels that are appropriately "forward-shifted" with respect to pixels already used for synthesis. In addition to each "forward-shifted" pixel, its $(k-1)$ nearest neighbors by the neighborhood distance should also be included in $C(v,\varepsilon)$. Our algorithm builds $C(v,\varepsilon)$ using the k-coherence technique with an additional check for the condition in Equation (1) for each candidate pixel p. Following [Tong et al. 2002], we accelerate the k-coherence technique by pre-computing the k nearest neighbors for each pixel of the 2D sample texture. For all the examples in this paper, we set $k = 20$.

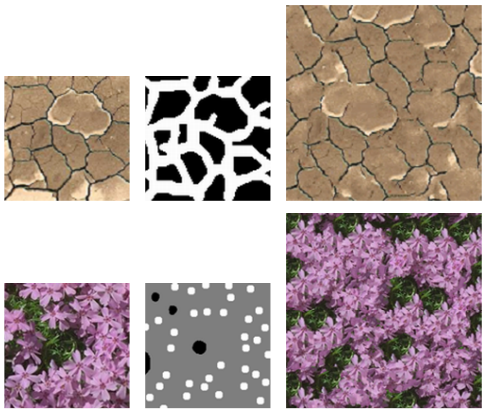


Figure 11: Homogeneous texture synthesis using texton masks. Left column shows sample textures. Middle column shows input texton masks (see Appendix A in the CD-ROM for the target texton masks). Right column shows synthesis results. In the bottom row, color thresholding fails and the texton mask is hand painted (white dots for flowers and black dots for leaves).

An alternative approach to handling the transition functions is to put the transition function values in the alpha channel and compute the L2-norm of RGBA-vectors during texture synthesis. This is the approach taken by [Hertzmann et al. 2001] and it works for texture-by-numbers applications. For progressively-variant textures, this technique has the drawback that the pixel found by the search procedure may not satisfy the condition in Equation (1).

Local Orientation Control: For synthesis of stationary textures on a mesh, it is standard to have an orientation field on the mesh. The difference with a progressively-variant texture is that it needs an orientation field on the input 2D texture as well. Fig. 2 explains why an input orientation field is necessary.

5.3 Discussion

Texton masks are also useful for homogeneous texture synthesis, as Fig. 10 shows. A case in which texton masks are extremely useful is illustrated by the pepper example in Fig. 13. This is a challenging example for existing algorithms. Wei and Levoy's method [2000] would have difficulty because of the L2-norm. The tiling method [Efros and Freeman 2001; Liang et al. 2001] will also have trouble because it is based on L2-distance matching of boundary zones of texture patches. When applied to the pepper image, the tiling method can create "non-peppers" near texture patch boundary.

The top example in Fig. 11 is also a challenge for existing algorithms. Even with interactively extracted texture elements, Dischler et al. [2002] reported difficulty with this texture. Our algorithm can handle this example well because we combine the strengths of both non-parametric sampling and user-specified texton masks.

6 Results

Fig. 1 shows a leopard-horse generated by our system. This example was inspired by the homogeneous leopard-horse in [Turk 1991]. The progressively-variant 2D texture sample was generated by feature-based warping as described in Fig. 6. Fig. 12 exhibits a collection of examples. The top left example is the head of a venus statue. We blended between two kinds of marble by feature-based blending to create the progressively-variant 2D texture sample. Notice that the color and structure of the marble veins progressively change over the whole surface. The homogeneous texture images of marble are shown next to the statue. The top right example is the

Figure 12: Progressively-variant textures on surfaces. The initial homogeneous texture samples are shown next to each example.

Mesh	Sample size	Vertex number	Time (minutes)
Venus	512×128	192k	44
Bunny	512×128	300k	104
Giraffe	384×96	106k	33
Tiger	256×64	170k	51
Horse	448×128	250k	39

Table 1: The timings are measured on a 2.4 GHz Xeon workstation.

pepper bunny. The transition between red peppers to yellow peppers is produced by feature-based blending. Observe that individual peppers are packed together without breaking. The homogeneous texture samples of the red and yellow peppers are shown next to the bunny. The bottom left example is a giraffe. Note that the shape and size of the texture elements gradually change along the giraffe's legs and neck. The bottom right example is a tiger. Notice the stripes on the tiger's shoulder are made progressively thinner. This is achieved by applying feature-based warping to a homogeneous texture sample as described in Fig. 5. The homogeneous texture sample used in this example is shown near the tiger. The texture for the entire body of the animal was generated by our system. The texture of the tiger's face was painted by an artist. Table 1 provides timings for synthesizing progressively-variant textures onto surfaces.

Although color thresholding may not always generate a good segmentation in the traditional sense, the resulting texton masks are usually good enough for our purposes because of the low requirements of our techniques on texton masks. Fig. 13 demonstrates the robustness of our synthesis algorithm to variations in texton masks. Fig. 11 (bottom row) shows an example for which color thresholding fails to extract a texton mask. In this example, we hand painted a texton mask and our algorithm was able to produce good results

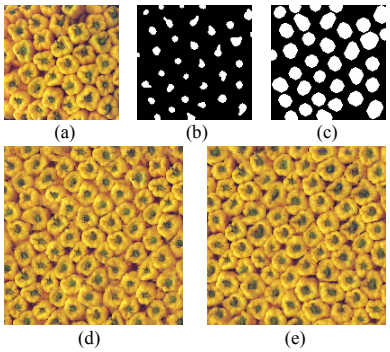


Figure 13: Robustness of our synthesis algorithm to variations in texton masks. (a) Texture sample. (b) and (c): two significantly different texton masks of the sample. (d) Texture synthesized using the mask in (b). (e) Texture synthesized using the mask in (c).

with this low-quality texton mask.

Appendix A and Appendix B in the conference CD-ROM provide additional experimental results.

7 Conclusion

Our main contribution in this paper is a framework for progressively variant textures on arbitrary surfaces. The most novel aspects of our work are feature-based warping and blending of textures, as well as synthesis of progressive variant textures on arbitrary surfaces. We expect our work to inspire a lot of creative methods for decorating surfaces with textures. We feel that our contribution may not lie so much in the technical solutions to each individual problem per se, but rather in the overall framework of progressively variant textures. The general framework we propose should be applicable to most textures, even though individual technical solutions may be restricted to certain types of textures.

One area of future work is to add more user control to feature-based blending. Our current method computes a texton mask for the blend texture such that texture elements change shape smoothly. However, the user may want more control on the way they change for achieving the most "natural" blend between the two given textures. Another topic is to explore the multi-way transition among more than two textures. Finally, we are interested in other ways to control the local variations of textures.

Acknowledgments: We want to thank all reviewers for their constructive critiques, which have significantly improved our paper. In particular, we appreciate reviewer #3 and the senior reviewer's insightful comments and excellent (and detailed!) revision suggestions. Many thanks to Ke Deng for implementing and testing many parts of our system, to Xin Tong for useful discussions, and to Steve Lin for his help in video production and proofreading of this paper.

References

ASHIKHMIN, M. 2001. Synthesizing natural textures. *ACM Symposium on Interactive 3D Graphics*, 217–226.

BAR-JOSEPH, Z., EL-YANIV, R., LISCHINSKI, D., AND WERMAN, M. 2001. Texture mixing and texture movie synthesis using statistical learning. *IEEE Transactions on Visualization and Computer Graphics 7*, 2, 120–135.

BEIER, T., AND NEELY, S. 1992. Feature-based image metamorphosis. *Computer Graphics (Proceedings of SIGGRAPH 92) 26*, 2 (July), 35–42.

BROOKS, S., AND DODGSON, N. 2002. Self-similarity based texture editing. *ACM Transactions on Graphics 21*, 3 (July), 653–656. (Proceedings of ACM SIGGRAPH 2002).

DISCHLER, J., MARITAUD, K., LÉVY, B., AND GHAZANFARPOUR, D. 2002. Texture particles. *Computer Graphics Forum 21*, 3, 401–410.

EFROS, A. A., AND FREEMAN, W. T. 2001. Image quilting for texture synthesis and transfer. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, 341–346.

EFROS, A. A., AND LEUNG, T. K. 1999. Texture synthesis by non-parametric sampling. In *Proceedings of International Conference on Computer Vision*.

GOMES, J., COSTA, B., DARSA, L., AND VELHO, L. 1998. *Warping and Morphing of Graphics Objects*. Morgan Kaufmann.

GORLA, G., INTERRANTE, V., AND SAPIRO, G. 2001. Growing fitted textures. *SIGGRAPH 2001 Sketches and Applications* (August), 191.

GUO, C., ZHU, S. C., AND WU, Y.-N. 2001. Visual learning by integrating descriptive and generative methods. In *Proceedings of International Conference on Computer Vision (ICCV)*, 370–377.

HARRISON, P. 2001. A non-hierarchical procedure for synthesis of complex textures. In *Proceedings of Winter School of Computer Graphics*.

HERTZMANN, A., JACOBS, C. E., OLIVER, N., CURLESS, B., AND SALESIN, D. H. 2001. Image analogies. *Proceedings of SIGGRAPH 2001* (August), 327–340.

KINGDON, J. 1977. *East African Mammals*, vol. III A. The University of Chicago Press.

LEUNG, T. K., AND MALIK, J. 1996. Detecting, localizing and grouping repeated scene elements from an image. In *Proceedings of European Conference on Computer Vision*.

LIANG, L., LIU, C., XU, Y., GUO, B., AND SHUM, H.-Y. 2001. Real-time texture synthesis using patch-based sampling. *ACM Transactions on Graphics 20*, 3 (July).

LIU, Z., LIU, C., SHUM, H.-Y., AND YU, Y. 2002. Pattern-based texture metamorphosis. In *Proceedings of Pacific Graphics*, 184–191.

MALIK, J., BELONGIE, S., SHI, J., AND LEUNG, T. 1999. Textons, contours and regions: cue integration in image segmentation. In *Proceedings of International Conference on Computer Vision (ICCV)*, 918–925.

MALLAT, S., PAPANICOLAOU, G., AND ZHANG, Z. 1998. Adaptive covariance estimation of locally stationary processes. *Annals of Statistics 26*, 1, 1–47.

MALLAT, S. 2003. Personal communication.

PRAUN, E., FINKELSTEIN, A., AND HOPPE, H. 2000. Lapped textures. *Proceedings of SIGGRAPH 2000* (July), 465–470.

SOLER, C., CANI, M.-P., AND ANGELIDIS, A. 2002. Hierarchical pattern mapping. *ACM Transactions on Graphics 21*, 3 (July), 673–680. (Proceedings of ACM SIGGRAPH 2002).

SUZUKI, S., AND ABE, K. 1985. Topological structural analysis of digital binary images by border following. *Computer Vision, Graphics, and Image Processing 30*, 1, 32–46.

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

TONIETTO, L., AND WALTER, M. 2002. Towards local control for image-based texture synthesis. In *Proceedings of SIBGRAPI 2002 - XV Brazilian Symposium on Computer Graphics and Image Processing*.

TURK, G. 1991. Generating textures for arbitrary surfaces using reaction-diffusion. *Computer Graphics (Proceedings of SIGGRAPH 91) 25*, 4 (July), 289–298.

TURK, G. 2001. Texture synthesis on surfaces. *Proceedings of SIGGRAPH 2001* (August), 347–354.

WALTER, M., FOURNIER, A., AND MENEVAUX, D. 2001. Integrating shape and pattern in mammalian models. *Proceedings of SIGGRAPH 2001* (August), 317–326.

WEI, L.-Y., AND LEVOY, M. 2000. Fast texture synthesis using tree-structured vector quantization. *Proceedings of SIGGRAPH 2000* (July), 479–488.

WEI, L.-Y., AND LEVOY, M. 2001. Texture synthesis over arbitrary manifold surfaces. *Proceedings of SIGGRAPH 2001* (August), 355–360.

WEI, L.-Y. 2001. *Texture Synthesis by Fixed Neighborhood Searching*. Ph.D. Dissertation, Stanford University.

WITKIN, A., AND KASS, M. 1991. Reaction-diffusion textures. *Computer Graphics (Proceedings of SIGGRAPH 91) 25*, 4 (July), 299–308.

YING, L., HERTZMANN, A., BIERMANN, H., AND ZORIN, D. 2001. Texture and shape synthesis on surfaces. *Proceedings of 12th Eurographics Workshop on Rendering* (June), 301–312.

ZALESNY, A., FERRARI, V., CAENEN, G., DER MAUR, D. A., AND GOOL, L. V. 2002. Composite texture descriptions. In *Proceedings of ECCV*, vol. 3, 341–346.

ZHU, S., WU, Y., AND MUMFORD, D. 1998. Filters random fields and maximum entropy (frame). *International Journal of Computer Vision 27*, 2 (March), 107–126.