FEATURE-BASED TEXTURE SYNTHESIS AND EDITING USING VORONOI DIAGRAMS

Muath Sabha
Department of Multimedia Technology
Arab American University-Jenin, Palestine
email: msabha@aauj.edu

Philip Dutré
Department of Computer Science
Katholieke Universiteit Leuven, Belgium
email: philip.dutre@cs.kuleuven.be

Abstract
In this paper, we present a technique to synthesize featured textures easily and interactively. The main idea is to synthesize a new texture by copying irregular patches from the source to the target texture, each of which contains a complete feature. The interior part of the feature is not touched while the cutting and stitching is performed on the background texture between the features.

The technique starts by selecting a feature in the source texture by the user, after which the algorithm finds the positions of other features, generates a similar distribution of features, and finally synthesizes the target texture by copying and stitching patches of the target’s Voronoi cellular shapes from the source texture. The technique is fast enough to be used interactively to edit textures in a simple and easy way.

KEY WORDS
Texture editing, Voronoi diagrams, Image synthesis.

1 Introduction
Computer graphics applications often use textures to decorate virtual objects to increase realism without modifying geometric details. In recent years, texture synthesis has been a useful tool to generate large textures starting from a sample texture. It is also used to fill holes in damaged pictures, or to generate limitless textures in interactive applications like games and simulators applications. Sample-based texture synthesis schemes are based on analyzing the sample texture to create visually similar images.

From a sample image taken from nature by a photographer or made by an artist, our technique generates an infinite aperiodic texture by copying irregular patches from the sample to the target texture. Figure 1 outlines the stages of the technique. Using a suitable editor, the user can modify the generated texture easily. The user can add, move, remove and merge features from different textures as shown in figure 6.

This paper starts by a quick overview of the recent contributions and most related to our work in the field of texture synthesis in section 2. In section 3, our technique will be explained in detail. Our work is composed of two main components, (1) the first is analyzing the source texture and extracting the features’ positions, and then generating a similar distribution in the target texture and that is explained in the subsections 3.1 and 3.2. (2) The second and major component, which we consider the main contribution in our work, is the shape of the patch and its placement in the target texture, which is discussed in subsection 3.3. Texture editing is an important application of our technique, and that is explained in 3.4. In section 4, we show some results generated from different textures, followed by discussion. Finally, section 5 concludes this work and puts some recommendations for future work.

2 Related work
There are various sample-based texture synthesis approaches which start from an exemplar to generate new textures:

Texture Synthesis by Analysis generates a new texture depending on statistics extracted from the sample texture [2, 15, 27].

Pixel-based Texture Synthesis techniques depend on neighborhood matching to generate the target texture pixel by pixel. For each pixel, the already synthesized spatial neighborhood is compared to exemplar neighborhoods to identify the most elaborate pixels [6, 24, 19, 11, 8]. Most of the pixel-based techniques follow the Markov Random Field model.

Texture morphing [14] works better with Pixel-Based techniques, and [20] uses the texton mask for texture morphing. [22] and [25] used the pixel-based approach to synthesize texture on surfaces, and [28] used a a data structure called jump-map to synthesize pixels over surfaces.

Efros and Freeman [5], Xu et al. in [26] and Liang et al. in [12] suggested copying one patch-at-a-time instead of one pixel-at-a-time as a Patch-based Texture Synthesis model. This maintains the interior part of the patch. The idea was stitching random patches from the sample texture together and modifying them in a consistent way. The GraphCut [9] and GrabCut [17] techniques improved the way of cutting and stitching the patches. Liu et al. [13] worked on synthesizing the near regular textures and for that, the user inputs are important in assigning the latices and the technique extracts the deformations in color and geometry. The lapped texture technique [16] tries to stitch random patches over 2D and 3D meshes,while [18] tried to speed up the patch selection process. [21] uses wavelets to
select suitable patches. Some techniques try to build different square tiles from the source texture [3, 10], and then perform the synthesis by matching edges or corners. The main drawback of tile-based texture synthesis is the limitation of the variety due to the fixed tile set and often the tiling appears regular if the texture is viewed from afar, especially for non-homogeneous textures.

A new class of image editing tools has emerged which employs this from texture synthesis to perform sophisticated image editing operations including Texture-By-Numbers (Image Analogies) [7], region filling and object removal from scenes [4] and photo montage [1].

3 Our Technique

We try to synthesize a range of textures where "clear and relatively big" features are present in front of a background, which is often a stochastic texture in our examples. Those textures can be found commonly in nature such as flower fields, sand with stone pebbles, animals and birds in fields, on water and in sky. We target this category of textures because most of the existing texture synthesis algorithms do not work well with input textures of two different categories (featured and stochastic), or they would need different parameters for each part of the texture. Our approach is also motivated by the observation that the human eye notices mostly main features in an image, while at the same time being sensitive to cuts in these main features [23].

Previous Patch-Based texture synthesis techniques do not take care of the patch shape or its position relative to the features. Most of the previous techniques (i.e. [5, 17, 18, 13]) use rectangular shape patches which is mostly suitable for near regular textures, while others use randomly irregular patch shapes by finding minimal cuts [9].

In our paper, the expression feature refers to the big repeated parts in the texture, where the user is attracted in the first look.

3.1 Feature positions

Our technique starts by finding the centers of the main features in the source texture. The user manually selects a feature from the source texture; based on the CIE L*a*b* color space, the $L^2$ norm is calculated between the selected feature and the sample texture. The technique then finds the positions of similar features as shown in figure 2.

The user is given the chance to assign the main feature to simplify the process of detecting features and to speed
it up. The user doesn’t have to select the feature accurately. In most of the textures we have examined, a simple box, rather than a polygon, is enough for the feature position finder of our technique. Relatively, the effort and time taken from the user is negligible compared to the time to be taken by an automatic program to detect which kind of features are the main features and which are wanted to be considered as back ground. Until now, to my knowledge, there is no technique that can detect the repeated feature from a general texture, some of them may succeed in regular or near regular textures.

To improve the result, the user can interactively add, move or delete feature centers to make the resulting source distribution as accurate as possible, which means, if the cross correlation doesn’t detect all the features, or some of the features near edges, and their existence helps in generating the target distribution, those feature positions can be added. The time taken to modify the result also is negligible and that is because in most of the examined textures, it doesn’t have to be accurate in assigning the positions, but it is enough to put the feature nearly in the center of a Voronoi cell.

### 3.2 Distribution

After detecting the distribution of features in the source texture, a similar distribution is to be generated in the target. The generated point distribution will be used in synthesizing the target texture, where each point will be a position for a feature.

The first step in generating the target distribution is to construct the Voronoi Diagram of the source distribution, and let’s call it the source space. Each point in the source space is a cell generator in the Voronoi diagram.

The first thing which could be learned from the Voronoi diagram is the direct neighbors. For example, the direct neighbors of the point $SP_7$ in figure 3(a) are $\{SP_2, SP_3, SP_5, SP_0, SP_{10}, SP_{12}\}$. The Voronoi diagram also shows which of the points are edge points (edge cells) and which are internal points. For example, $SP_{12}$ is an edge node, $SP_5$ is a corner node, while $SP_7$ is an internal point.

The regularity is calculated from the vectors to the direct neighbors. We calculate the standard deviation of the angles and the standard deviation of the distances to the next neighbors. From those standard deviations, and based on the lattice theory, the regularity of the distribution is calculated. The minimum distance and maximum distance between neighbors are also calculated.

Generating the distribution starts with a node near the top left corner. This first node is randomly selected within the range of distances between minimum cell radius and maximum cell radius from the x-axis and y-axis taking the upper left corner as the origin. That is the blue node in figure 3(b). A random point is taken from the source space ($SP_n$), which is not an edge point, and the neighbors of this point are copied to the target space with reference distances to the target point equal to the reference distances from $SP_n$ in the source space. Those situated points have to be within the target space dimensions.

The point in the target space which is wanted to find it direct neighbors will be called the current point $CP$. A list of all the direct neighbors of the point $SP_n$ in the source is constructed and called SurroundingsList. Each point $TP_n$ from the SurroundingsList is tested to be added to the target space on a relative position to $CP$ as its relative position to $SP_n$. If the tested point satisfies the conditions, it is added to the TargetQueue, otherwise, it is rejected and the technique continues with the next point from the SurroundingsList. The Voronoi diagram is constructed for testing

![Figure 3. Generating a target distribution from the source distribution. The blue point in the target distribution is the initial point, the green points are the located (fixed) points in previous steps, and the yellow points are the suggested points at the current step, which is finding the next neighbors from the red point $CP$ (Current Point). Three of them are discarded (those with the red crosses), because the distances to their fixed neighbors are less than the minimum distance. The other two are accepted $\{TP_5, TP_6\}$. The source distribution is taken from the source distribution of figure 2.](image)
each point to be added to the target space. When the SurroundingsList is empty, a new point from the TargetQueue is popped out to be the new CP. This procedure runs until the TargetQueue is empty, which means all the target space is covered.

Let’s explain the previous steps by taking the example in figure 3. It is needed to generate a target distribution (figure 3(b)) from a given source distribution (figure 3(b)). The source distribution is the distribution in the figure 2(d) resulting from the source texture in figure 2(a).

Suppose that we reached the situation in figure 3(b), where the current point is CP (red spot) and the green points have already been placed successfully in previous steps to the target space and added to the fixed list. We want to generate the next points based on CP. A random point from the source space is picked, and suppose that it is $SP_{10}$ from figure 3(a). It is clear that the surroundings of $SP_{10}$ are $\{SP_{7}, SP_{8}, SP_{11}, SP_{12}, SP_{9}\}$ with relative spacings from this point indicated by blue lines. When placing those 5 surroundings to the target space, they are given the names $TP_{1}$ to $TP_{5}$, which are the yellow spots in figure 3(b). The new $TP_{n}$ points are tested one-by-one to be added to the fixed list. In the testing procedure to add a new point, the Voronoi diagram is created, and the distances between the new added point and its surroundings are tested. If the radius of the cell matches (equals or exceeds) the radius of a random cell taken from the source (it could be different from $SP_{10}$ in our example), the point is added to the target space, otherwise, it is discarded. In the example of figure 3(b), the points $TP_{1}$, $TP_{2}$ and $TP_{3}$ are discarded, while $TP_{3}$ and $TP_{5}$ are kept in the space and added to the TargetQueue. Each of the points in the TargetQueue will be popped out and act as a CP.

In the generated distribution, the user can control the density of the generated distribution. This can be done by introducing a factor $\alpha$ for the ratio between the distances in the source space to the distances in the target space ($dT = \frac{1}{\alpha} d_s$). If this factor is less than 1.0, the effect will be increasing the density for the resulting distribution, and if it is more than 1.0, the target space will be sparser than the source space. Different results are shown in figure 6. The factor must not be less than the value which makes the minimum target Voronoi cell radius equals to $R_{min}$ (the radius of the selected feature) which is shown in figure 2(b). This ensures that complete features fit inside each cell.

### 3.3 Patch selection and stitching

After generating the distribution for the target texture, a Voronoi diagram is constructed. Each cell generator (point) indicates a feature center. Starting with the first cell in the top left corner, a feature position from the source texture is selected randomly. A patch with similar shape (and size) of the target cell is copied from the source and placed in the target texture.

The patch from the source texture to be copied is not only the target cell area, but a region around the selected cell is copied with the patch to overlap the surrounding cells as shown in figure 4. The width of the region can be defined as a parameter, but, in all our examples, we used a relative constant band width which is 0.25 the distance between the point of the Voronoi cell and its neighbor, again to ensure that features aren’t cut.

In order to blend the new patch, the $L^2$ norm of the overlapped region is constructed. A cutting algorithm is used to find the best cut. In our implementation, we used the GraphCut [9] algorithm.

An other parameter can be applied which is the perspective flow to improve the result by putting the copied cells in the same relative position between source and destination. The geese in figure 6 were given a perspective priority to the flow of the features in the source picture. Applying the perspective flow to the source texture gives the result of stretching the texture in a very smooth way resulting in enlarging the texture without loosing the perspective distances.

### 3.4 Texture Editing

In order to provide more flexibility, the technique enables the user to edit textures in a very easy and interactive way. The user can control the distribution of the target texture...
by adding, moving, resizing or deleting features. The technique also enables the user to merge textures (especially if the background is easily matched) as shown in figure 6.

4 Results and Discussion

The main objective of our feature-based texture synthesis and editing technique is to copy complete features from the source to the target textures and to make cuts on the background between features without touching the features themselves. From the results shown in figure 6, we noticed that the aim of the technique is achieved where features are not affected while being copied from source to destination.

From the false color image result shown in figure 5, it is clear that irregular patterns have been copied from random places in the source to the target textures. It is clear that complete features are copied with patches of irregular shapes.

5 Conclusions and Future Work

We discussed a technique which improves the synthesis procedure for an important category of natural textures. The main contribution of the technique is choosing irregular patches such that the interior part of the features are not affected from any discontinuity caused from copying and stitching patches from source to target textures. Our technique generates distributions which are congruous to the source distributions with the ability to change the density. Our technique is fast enough to edit and synthesize textures of high resolutions interactively and in a very flexible, easy and efficient way.

We believe that the technique can be extended to work with the full range of the feature based textures. The technique can be extended to map textures directly on 3D surfaces. The inevitable deformation can be done between the features and the features could be scaled to match the target mesh. The same texture editing can be done over 3D surfaces.

References


Figure 6. Some of the results (on the right) together with their sample textures (on the left.). Different density control factor ($\alpha$) values where used.


