Synthesizing Variational Direction and Scale Texture on Planar Region

Yan-Wen Guo[1,2,⋆], Xiao-Dong Xu[1], Xi Chen[1], Jin Wang[1], and Qun-Sheng Peng[1]

[1] State Key Lab of CAD and CG, Zhejiang University, Hangzhou 310027, China
[2] State Key Lab for Novel Software Technology, Nanjing University, Nanjing 210093, China
{ywguo, xuxiaodong, xchen, jwang, peng}@cad.zju.edu.cn

Abstract. Traditional 2D texture synthesis methods mainly focus on seamlessly generating a big size texture, with coherent texture direction and homogeneous texture scale, from an input sample. This paper presents a method of synthesizing texture with variational direction and scale on arbitrary planar region. The user first decomposes the interest region into a set of triangles, on which a vector field is subsequently specified for controlling the direction and scale of the synthesized texture. The variational texture direction and scale are achieved by mapping a suitable texture patch found in the sample via matching a check-mask, which is rotated and zoomed according to the vector field in advance. To account for the texture discontinuity induced by not well matching or different texture directions/scales between adjacent triangles, a feature based boundary optimization technique is further developed. Experimental results show the satisfactory synthesis results.

Keywords: Computer graphics, texture synthesis, texture direction and scale, feature matching.

1 Introduction

Texture synthesis means generating a big size texture from an input texture sample, such that the big size texture appears similar pattern with the sample locally while presents some stochastic attribute globally. As its wide applications in game/film producing, virtual reality, etc., texture synthesis has been widely studied by the researchers of computer vision/graphics for many years [1-11].

Earlier methods emphasize on analyzing the stochastic property of the texture [1-6], and apply obtained texture model to supervise the synthesis process. Recently, some methods deem the synthesis process as a Markov process and texture the given region based on neighbors' matching [7-11]. That is, to synthesize texture on a certain region, a rectangular patch is recursively textured to fill in this region, and the current patch to be textured is only related with its neighbors' texture. Although nearly seamless synthesis results can be obtained with

⋆ Corresponding author.

Y. Zhuang et al. (Eds.): PCM 2006, LNCS 4261, pp. 159–166, 2006.

above methods, most of them can only generate texture with identical texture direction and homogeneous texture scale on a rectangular region. It is difficult for them to conveniently synthesize texture with variational texture direction or scale. However, this class of texture is much useful and universal in our life, such as the tiles of winding road, the pattern of fabric, tree's skin, etc.

In this paper, we present a method of synthesizing texture with variational direction and scale. The method still regards the synthesis process as a Markov process, while applies triangle as the synthesis unit. So we can generate texture on arbitrary 2D region. The variational direction is achieved by rotating the current triangle to be textured and a check-mask, with the angle composed of a given referenced vector and the specified vector adhering to the current triangle, when searching for the suitable texture patch in the sample. Meanwhile, variational scale is generated by zooming in or out the current triangle and its check-mask according to the proportion between the module of the referenced vector and that of the specified vector. To relax texture discontinuity between adjacent triangles, we further introduce a feature matching and warping based boundary optimization technique, which in fact results in resolving the "assignment problem" in linear programming with high efficiency. It is convenient to extend this method to directly synthesize variational texture on surface.

The rest of this paper is organized as follows. Section 2 introduces some related work. We provide the method of synthesizing variational texture detailedly in section 3, and present the feature based matching and warping technique in section 4. Some experimental results are given in section 5. The last section summarizes our work and highlights the future work.

2 Related Work

Earlier methods concentrated on analyzing the texture models and can be classified by the models they applied, such as reaction-diffusion [1], and random field models [2-3], etc. Some also used hybrid models that include a periodic component and a stochastic component [4].

Recently, neighborhood based methods have attracted much attention and achieved significant progress [7-11]. The most important stage of these methods is to search for a suitable texture pixel/patch in the input sample, which is most matched with its neighbors' existent texture. Among these methods, applying pixel as the synthesis unit and texturing a pixel in each step can achieve fine results, but the efficiency is low. Although applying patch as the unit can enhance the performance, texture discontinuity between adjacent patches is a key issue that should be addressed carefully.

To resolve this problem, some methods perform a fusion operation in the common region of the current patch and the existing one, whereas the methods in [8][10] optimize the patch merging stage and find a best texture segmentation by dynamic programming or graph cut technique. With the help of feature map, Wu et al. developed a novel algorithm to perform feature matching and alignment by measuring structural similarity [11], and got remarkable synthesis results.

3 Synthesizing Variational Texture

Our method of texture synthesis adopts triangle as the synthesis unit. As any 2D region with close boundaries can be triangulated by the standard Delaunay algorithm, we can synthesize texture on arbitrary planar region.

In the following, we mainly describe the method of synthesizing texture on the triangulated region, and how to generate the effect of variational direction and scale. First of all, the creation of vector field will be addressed.

3.1 Creation of Vector Field

After triangulating the interest region, we need designate the vector field on this region for controlling the synthesized texture's direction and scale. The user first specifies the vectors of a few seed-triangles, then the inverse distance weighted interpolation method is used to generate the vectors on the rest of the triangles. Besides the vector field, a referenced vector should also be given by the user, which indicates the standard texture direction/scale in sample.

3.2 Texture Synthesis on Triangles

Our method is still neighborhood based. However, it is not feasible to texture the triangulated region with scan line order as most previous methods have done. Instead, we use a priority based traversal method to texture each triangle. The priority of a triangle is defined as the number of its neighbors that have been synthesized. Initially, all the triangles included in the region are pushed into a priority queue with zero priorities. After this, the triangle with the highest priority will be iteratively removed from the queue and be textured, and the priority queue is refreshed simultaneously.

It it obvious that, for the first time, since the triangle removed from the queue has zero priority, this triangle is filled with an arbitrary texture patch selected from the sample. Additionally, the priority of each triangle in fact predicates the constraints to this triangle. That is, the higher priority, the more constraints its check-mask includes. Synthesizing the triangle with the highest priority in each step can actually facilitate searching for the matched texture patch, and reduce efficiently the discontinuity between adjacent triangles.

In the following subsections, we exploit some details of the algorithm.

3.3 Extraction of Check-Mask

Classical neighborhood based methods, like [5, 7], utilize the overlapped region between the current unit and its adjacent synthesized ones as the template to search for the best matched texture patch in the sample, which works well for most kinds of textures.

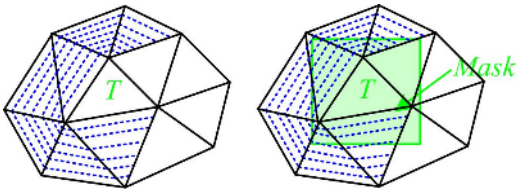

Fig. 1. Check-mask. The left: the current triangle T and its synthesize adjacent neighbors (filled with blue dashed. The right: the check-mask, which is a square surrounding the T.

However, in our method, we can not easily define a regular template as most previous methods have done, for the unknown and irregular shape of the synthesized triangles. Instead, we define here a check-mask, which is in fact a square surrounding the current triangle as shown in Fig. 1. Assume that the triangles filled with the blue dashed have been textured (the left figure). Then their corresponding region lying in the blue square (the right figure) will take effect during the searching process, while the rest area in the mask is useless.

The size of the check-mask is selected in proportion to the size of the bounding box of the current triangle. In fact, the smaller the mask is, the faster the searching speed is; or else the lower the speed is. Nevertheless, if the mask is too small, the continuity or structure of the synthesized texture can not be well preserved. So it is a balance to felicitously select its size. We empirically value the mask 1.2 multiple of the bounding box in our experiments, and restrict its width within 20 pixels as well to ensure the synthesis speed. Experimental results show that this mechanism for choosing the mask works well in most cases.

3.4 Transformation of Check-Mask

To generate the effect of variational texture direction and scale, we need transform the check-mask in the light of the specified vector of the current triangle T. The transformation here in fact can be factorized into a rotation part answering for the variational direction, and a zoom one accounting for the different scale. The check-mask is rotated in term of the angle between the referenced vector specified and the vector of T, and zoomed in or out according to the proportion between the module of the referenced vector and that of T's vector.

Once we have transformed the check-mask, we can search for the suitable texture patch for T in the sample. The searching process is performed by traversing all possible positions of the mask in scan-line order in the sample, and computing the color difference between the color of the mask and that of its corresponding position in the sample. The position with the minimal color difference indicates the texture patch best matched, which are therewith mapped to the position of T in the interest region.

4 A Feature Based Boundary Optimization Technique

For most neighborhood based methods, a key issue existent is the texture discontinuity at the boundary of the adjacent synthesis units. This problem is much more serious in our method incurred by different texture directions and scales between adjacent units. To resolve this problem, we introduce here a feature matching and warping based post processing technique to optimize the boundary.

In fact, for textures with salient features, especially highly structured textures, discontinuity is frequently caused by mismatching of features at the edges of triangles. So if we can rematch the features near the common edge and align them with minor deformation, the synthesis effect will be notably enhanced.

4.1 Feature Matching

Suppose T_P is the current triangle textured, and T_Q is one of its neighbors synthesized. We first detect the feature points, for example the image gradient exceeding a given threshold, on the common edge of T_P and T_Q. Assume that they are: $S_P = \{p_1, p_2, ..., p_n\}$ for T_P and $S_Q = \{q_1, q_2, ..., q_m\}$ for T_Q respectively (see Fig. 2). Our idea is to find a best match between the points of S_P and those in S_Q, relying on which, we align the textures near the common edge via minor deformation.

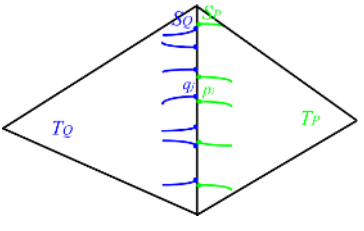

Fig. 2. Rematch of features. The blue/green curves represent the features detected for T_Q/T_P near the common edge, on which the blue/green nodes are the feature points to be matched and aligned.

We first define the matching energy about two points p_i, q_j as:

$$f_{i,j} = \omega_1 f_d(p_i, q_j) + \omega_2 f_c(p_i, q_j) + \omega_3 f_g(p_i, q_j). \quad (1)$$

Where f_d is the normalized Euler distance between two points punishing the large distortion and ensuring small texture deformation. f_c, f_g denote the normalized color and gradient differences respectively. ω_1, ω_2 and ω_3 are the weights balancing the actions of f_d, f_c and f_g, which are valued with 0.4, 0.3, 0.3 in our experiments.

Then the total energy for matching the points between S_P and S_Q is specified as:

$$F(P,Q) = \sum_{k=1}^{min(n,m)} f_{ik,jk}, \quad (1 \leq i_k \leq n, 1 \leq j_k \leq m). \tag{2}$$

For a valid matching of (2), the following terms are required. (a) Any two points in S_P should be matched with dissimilar points in S_Q. For two points in S_Q, the term is the same. (b) The points in one set should be orderly matched with the points in the other set. That is, cross matching is forbidden, since that will cause texture jumping.

According to above conditions, note that, since the points number of S_P may be not equal to S_Q, some redundant points are automatically abandoned after the matching process. The points in S_P or S_Q with the less number will be fully matched.

The objective function for the best matching is thus:

$$\arg\min_{\{ik,jk,(k=1,\ldots,min(n,m),1 \leq ik \leq n,1 \leq jk \leq m)\}} F(P,Q).$$

In fact, solving such a function can be easily converted into solving a "assignment problem" in linear programming, by adding some virtual points in the set with less points. The detailed solution can be found in [12]. However, as the number of feature points lying on the common edge is usually not large, we only need to enumerate all valid matchings (the number of which totals $C_{max(n,m)}^{min(n,m)}$), then the one with the minimal energy is the best matching. In most cases, the solution can be found in nearly real-time.

4.2 Texture Warping

With the matched result, we deform the texture of T_P slightly along the common edge. Without losing generality, assume the point number of S_P is less than S_Q. Thus any point included in S_P is matched with a point in S_Q.

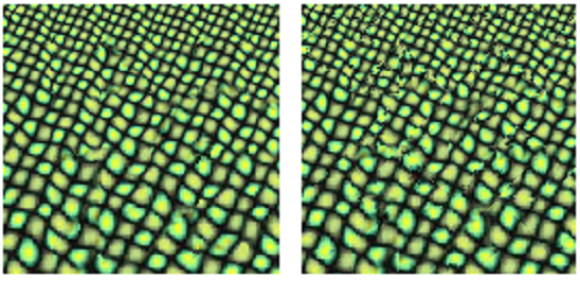

Fig. 3. Comparision between the results with (the left) and without (the right) our algorithm

We first add some virtual points along an edge parallel to the common edge, which have the same number with the points in S_P and preserve consistently relative positions on this edge with the points of S_P on the common edge. The region of T_P near the common edge is then uniformly triangulated with these points. Finally, the texture in each triangle is mapped to its new triangle resulting from the feature matching process. Fig. 3 compares the synthesis results generated with and without the algorithm.

5 Experimental Results

We have performed our experiments on an Intel Pentium IV 1.6GHz PC with 512MB main memory under the Windows XP operating system. Figs. 4, 5 demonstrate our experimental results.

Fig. 4 (a) & (b) give two synthesis results with direction field in vortex shape. The rectangle region consists of 96 triangles, and synthesis timing is 25 seconds. Fig. 4 (c) & (d) demonstrates two results. The synthesized textures become larger progressively from the top-right angle to the bottom-left of the square. We divide the rectangle region into 128 triangles, and fulfill the synthesis using 32 seconds. Fig. 4 (e) show the corresponding direction field for Fig. 4 (c) & (d).

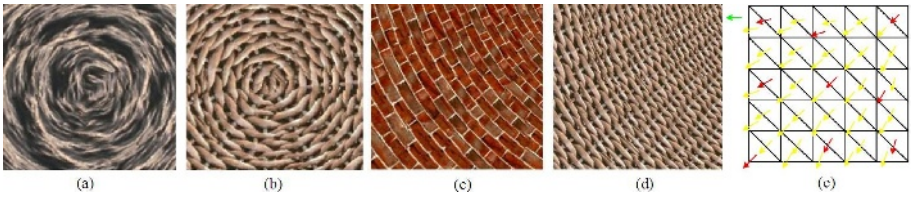

Fig. 4. Synthesis results. The direction field is similar to that shown in Fig. 1. Resolution is (256⋆256).

Fig. 5. Synthesis results. The diameter of each synthesized texture is 256 pixels.

Fig. 5 shows the results of synthesizing circular texture, whose scale around the center is smaller than that near the border. The circle is divided into 400 triangles, and it takes 96 seconds to synthesize one result.

6 Conclusions and Future Work

We have presented a method for synthesizing 2D texture with variational direction and scale, which was not dealt with by previous methods. By searching in the sample a check-mask transformed automatically according to the specified vector, the synthesized texture varies progressively satisfying the user's will. Furthermore, a feature based matching and warping post processing technique is developed, to treat with texture discontinuity between adjacent units. In the future we intend to extend our method to directly synthesize variational texture on 3D surface.

Acknowledgement

This paper is supported by National 973 program (No. 2002CB312101), NSFC grant (No.60403038) and CUHK Direct Research Grant(No.2050349).

References

1. Witkin A, Kass M. Reaction-diffusion textures. In *Porc. SIGGRAPH91*, LasVegas, July 1991, pp.299-308.
2. Fournier A, Fussel D, Carpenter L. Computer rendering of stochastic models. *Communications of the ACM*, 1982, 25(6): 371-384.
3. Bennis C, Gagalowicz A. 2-D macroscopic texture synthesis. *Computer Graphics Forum*, 1989, 8(4): 291-300.
4. Francos, J. M., Meiri, A. Z., Porat, B. A Unified TextureModel Based on a 2DWold-Like Decomposition. *IEEE Transactions on Signal Processing*, 1993, 41: 2665C 2678.
5. Heeger D J, Bergen J R. Pyramid-based texture analysis/synthesis. In *Proc. SIGGRAPH95*, Los Angeles, August 1995, pp.229-238.
6. Portilla J, Simoncelli E P. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 2000, 40(1): 49-70.
7. Wei L Y, Levoy M, Fast Texture Synthesis using Tree-structured Vector Quantization, In *Proc. SIGGRAPH00*, San Antonio, July 2000, pp.479-488.
8. Efros A A, Freeman W T. Image quilting for texture synthesis and transfer. In *Proc. SIGGRAPH01*, Los Angeles, August 2001, pp.341-346.
9. Liang L, Liu C, Xu Y Q, Guo B N, Shum H-Y. Real-time texture synthesis by patch-based sampling. *ACM Trans. Graphics*, 2001, 20(3): 127-150.
10. Kwatra V, Schodl A, Essa I, Turg G, Bobick A. Graphcut textures: image and video synthesis using graph cuts. In *Proc. SIGGRAPH03*, San Diego, July 2003, pp.277-286.
11. Wu Q, Yu Y Zh. Feature matching and deformation for texture synthesis. In *Proc. SIGGRAPH04*, Los Angeles, August 2004. pp.364-367.
12. Hillier F. S, Lieberman G. J. Introduction To Operations Research(7th edition). Prentice Hall, 2002.