Feature-Based Texture Synthesis

Tong-Yee Lee and Chung-Ren Yan

Computer Graphics Group/Visual System Lab(CGVSL),
Department of Computer Science and Information Engineering,
National Cheng-Kung University, Tainan, Taiwan, Republic of China
tonylee@mail.ncku.edu.tw

Abstract. We introduce a new method for texture synthesis on regular and irregular example textures. In this paper, an enhanced patch-based algorithm is proposed to select patches with the best structural similarity and to avoid discontinuity at the boundary of adjacent patches. This new method utilizes a feature-weighted function to measure the structural similarity. A re-synthesis technique is presented to reduce the structural discontinuity. We conduct a comparison study to verify the proposed method. Preliminary experiments show that the proposed method can yield better results than other well-known methods for examples used in this study.

1 Introduction

Texture synthesis is a popular research topic in computer graphics. In the past, many previous methods have been presented. The kernel of these previous works is structural similarity matching. Generally, there are two classes of methods: 1) pixel-based [1, 2, 6, 7, 8] and patch-based [3, 4, 5, 9, 10] methods, respectively. Pixel-based algorithms synthesize only one pixel at a time. Efros et al. [2] synthesize a new pixel by searching the sample image and finding the pixel that has the most similar neighborhood. This algorithm works fine if a larger neighborhood size is chosen to measure similarity. Therefore, it is very slow. Wei et al. [7] utilize a pyramid synthesis and a tree-structured vector quantization method to accelerate [2]. Later, Ashikhmin [1] extends [7] by reducing the searching space of the input image to several candidates. Hertmann [6] presents the other applications, like learning painting styles, based on [7]. Zelinka et al. [8] create a novel jump map, which stores a set of matching input pixels (jumps) for each input pixels. It allows their algorithm to synthesize texture in real time.

In contrast to pixel-based methods, patch-based methods synthesize a patch at a time. Xu et al. [3] synthesize new textures by random patch pasting. The problem with random patch pasting algorithm is that the discontinuity and artifacts at the overlapped region of the adjacent patches. Efros et al. [4] present a minimum-error-boundary-cut within the overlapped regions to reduce the discontinuity and artifacts. Liang et al. [5] apply feathering technique on the overlapped regions and accelerate the search by a quad-tree pyramid data structure. Wu et al. [9] produce a feature map and texture map to guide the patch selection

© Springer-Verlag Berlin Heidelberg 2005

and align overlapped region by measuring structural similarity. In contrast to most patch-based methods, Nealen et al. [10] adaptively splits patches to suitable sizes while satisfying user specified error for the mismatching in the overlapped regions. A pixel-based re-synthesis is then applied to the mismatched pixels in overlapped regions.

2 Methodology

To synthesize texture, most algorithms consist of two steps: 1) searching most similar neighborhood in the input image and 2) avoiding discontinuity in the generated texture. In this paper, the proposed patched-based method uses a feature-weighted function to guide our searching for the most similar neighborhoods and a re-synthesis approach to reduce the artifacts at the overlapped regions.

2.1 Similar Neighborhood Search

Our approach is a patch-based texture synthesis algorithm and it consists of the following steps:

1. Select an initial patch from input texture randomly and paste it to the left top corner of the output texture.
2. Search a new adjacent patch that has the most similar neighborhood in the input texture constrained by a L-shape.
3. Paste the selected patch to the output image in a scan-line order.
4. Re-synthesize the overlapped region.
5. Repeat step 2 to 4 until we complete output image.

In the past, most methods compare the difference of pixel color when they search the most similar L-shape neighborhood. The importance of each pixel in the L-shaped neighborhood is treated equally. However, in these methods, most artifacts happen to the boundary of adjacent patches. To alleviate this problem, we propose to strengthen the importance of pixels closer to the patch boundaries as we measure the similarity of patches. In our algorithm, a Gaussian distribution function is used to modify the importance of pixels. The distance weight equation W_d is defined as follows.

$$W_d(i,j) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{d(i,j)^2}{2\sigma^2}} \tag{1}$$

where $d(i,j)$ is the distance form pixel (i,j) to the closest patch boundary. Figure 1 illustrates the visualization of weight variation for each pixel in L-shape.

To judge whether a synthesis result is good or not depends on if its edge structure is similar to original texture or not. Hence, a structure-weighted function W_s is also considered as the similarity-searching criterion. In eq. 2, W_s is described as follows.

$$W_s(i,j) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{[|M_{color}-P(i,j)|-255]^2}{2\sigma^2}} \tag{2}$$

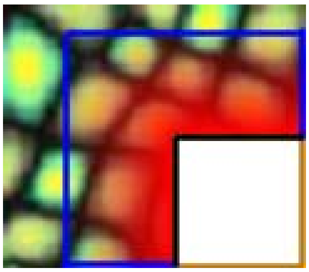

Fig. 1. Visualization of weight variation. Black line represents patch boundary at L-shape. The pixel color in L-shape means the weight of a pixel where red color represents the higher weight

where M_{color} represents the color with the maximum number of pixels among all similar colors in L-shape, $P_{i,j}$ is the color of the pixel at (i, j). In L-shape, most pixels should have colors that are closer to the based color M_{color}, and the structure elements such as edges and lines should have a larger difference to this based color. Therefore, a higher weight is given to pixels that have a larger difference to the based color M_{color}. Finally, both distance and structure weighted function, W_d and W_s, are integrated into our similarity measurement and is called the feature-weighted function below.

$$Sim = \sum_{i,j \in L-shape} P_{diff}(i,j)(W_d + W_s) \qquad (3)$$

where $P_{diff}(i,j)$ is the color difference of pixels in the L-shape between input and output images. Searching similar neighborhood using eq. 3 can help us find a patch with less artifacts at the boundary.

2.2 Repair Artifacts in Overlapping Regions

After the most similar patch is selected and is pasted into the output image, some artifacts may still exist. In this section, we will further reduce artifacts. First, we compute the error for each pixel in the overlapped region. Then, we use Efros et. al's method [4] to find a minimum error path through overlapped region. After the minimum error path is found, we treat this path as the central axis and define a five-pixel wide region, called *repairing area*. Figure 2 shows an example of a repairing area.

In the repairing area, we treat pixels as mismatched pixels if their errors exceed a selected threshold δ. Then we re-synthesize these mismatched pixels in a region-growing manner from the boundary of the repairing region to its central axis. The value of the mismatched pixel is repaired by the median value of the 5×5 matched neighboring pixels. After re-synthesizing all mismatched pixels, we finally apply Gaussian smoothing filter to these newly repaired pixels.

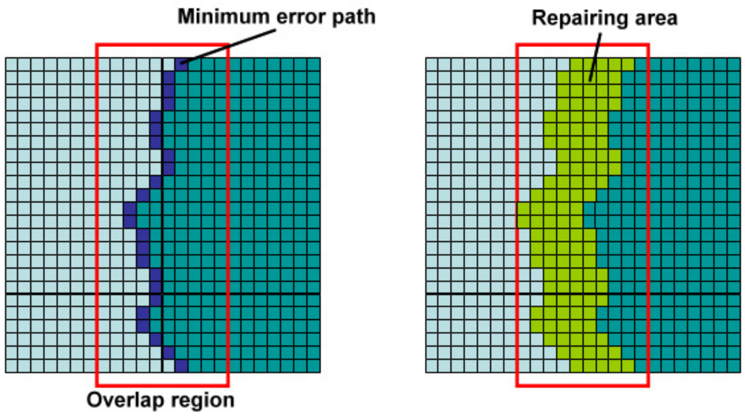

Fig. 2. Left: Computing the error for each pixel in overlapped region and finding a minimum error path. Right: Repairing area

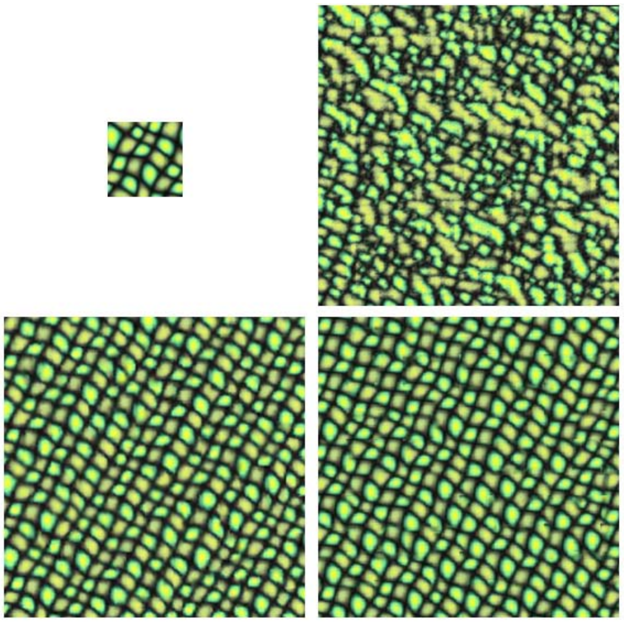

Fig. 3. Experimental Results

3 Experimental Results

In this section, we demonstrate some preliminary results in comparison with two well-know methods: pixel-based [1] and patch-based [5] algorithm. Figure 3 and Figure 4 are near-regular texture. Figure 5 shows an irregular texture example.

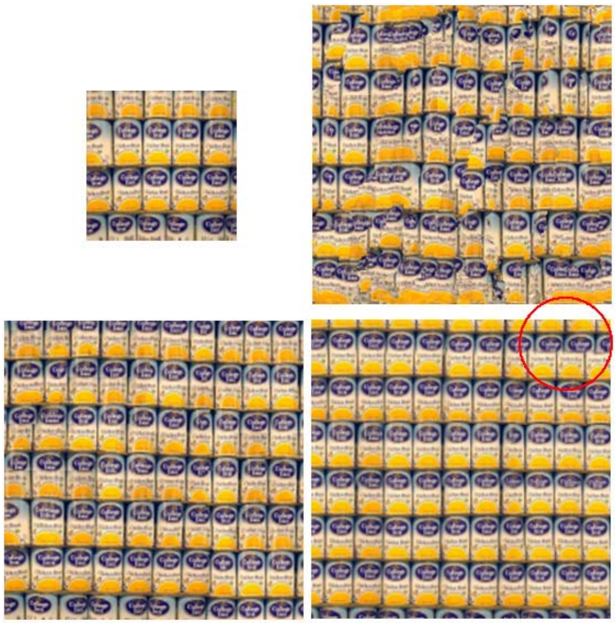

Fig. 4. Experimental Results

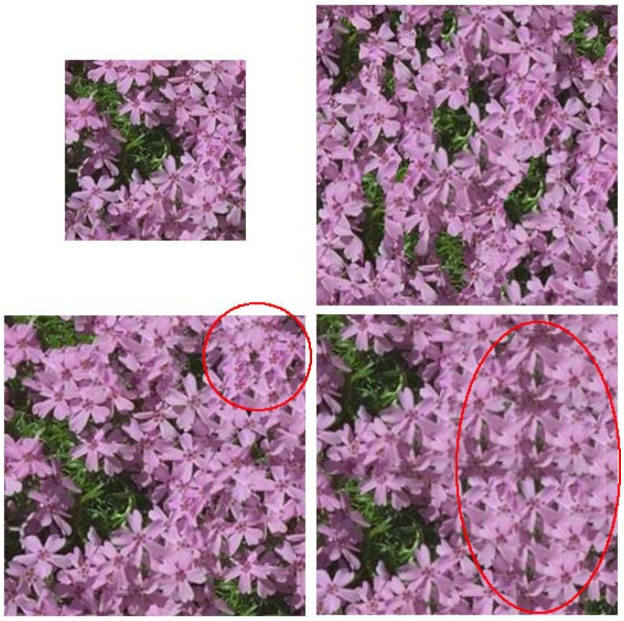

Fig. 5. Experimental Results

In these three examples, the left-top image is original input, the right-top image is synthesized by [1], the left-bottom image is synthesized by our method and the right-bottom image is synthesized by [5]. From our preliminary results on these three examples, the pixel-based method [1] does not work well. The results by our method are comparable to those by [5]. For further comparison, we see some difference between [5] and ours. In Figure 3, the structure distribution, grid size and shape by the proposed method are almost similar to those of the original sample texture. Figure 4 shows that our method can yield better result in preserving complete structural elements and continuity at the patch boundary. Please watch the top-right part of the result by [5]. There is some obvious discontinuity on the shapes of cans. In Figure 5, it is hard to compare difference between ours and [5].

4 Conclusion and Future Work

Pixel-based texture synthesis tends to introduce blurring effect and is more suitable for irregular textures, like clouds, ripples, and other natural images. Patch-based texture synthesis can preserve the global structure of the sample texture, but to reduce artifacts and discontinuity at patch boundary is an important work. In this paper, we proposed a new algorithm on both regular and irregular textures. The proposed method is a patch-based algorithm and it consists of two parts: 1) we first apply feature-weighted function to search the most similar neighborhood patches and 2) then, we re-synthesize mismatched pixels to reduce artifacts and to minimize the structural discontinuity. This new scheme can solve these problems well. In the near future, we plan to extend the current work to the perspective texture synthesis. We also like to apply our method to some textures with special structure, like the wing of butterfly and the tail of peafowl fish. In addition, we would like to apply our technique to synthesize texture for the expression change of head animation [11] or for 3D morphing applications [12, 13].

Acknowledgements. This paper is supported by the National Science Council, Taiwan, Republic of China, under contract No. NSC-93-2213-E-006-026 and NSC-93-2213-E-006-060.

References

1. M. Ashikhmin: Synthesizing natural textures. ACM Symposium on Interactive 3D Graphics, (2001) 217–226.
2. A. Efros and T. Leung: Texture synthesis by non-parametric sampling. Intl. Conf. Computer Vision, (1999) 1033–1038.
3. Y. Xu, B. Guo, And H.-Y Shum: Chaos mosaic: Fast and memory efficient texture synthesis. Microsoft Res. Tech. Rep. MSR-TR-2000-32, April (2000).
4. A. Efros, and T. Freeman: Image Quilting for Texture Synthesis and Transfer. Proceedings of SIGGRAPH, (2001) 27–34.

5. L. Liang, C. Liu, Y. Xu, B. Guo, and H.-Y Shum: Real-time texture synthesis using patch-based sampling. ACM Trans. Graphics, Vol. 20, No. 3, (2001) 127–150.
6. A. Hertmann, C. Jacobs, N. Oliver, B. Curless, and D. Salesin: Image analogies. Proceedings of SIGGRAPH, (2001) 327–340.
7. L.-Y. Wei And M. Levoy: Fast texture synthesis using tree-structured vector quantization. Proceedings of SIGGRAPH, (2000) 479–488.
8. S.Zelinka and M. Garland: Towards real-time texture synthesis with the jump map. Proceedings of the Thirteenth Eurographics Workshop on Rendering Techniques, (2002) 99–104.
9. Q. Wu and Y. Yu: Feature Matching and Deformation for Texture Synthesis. ACM Transactions on Graphics (SIGGRAPH 2004), Vol. 23, No. 3, (2004) 362–365.
10. A. Nealen and M. Alexa: Hybrid Texture Synthesis. Proceedings of Eurographics Symposium on Rendering, (2003) 97–105.
11. Tong-Yee Lee, Ping-Hsien Lin, and Tz-Hsien Yang: Photo-realistic 3D Head Modeling Using Multi-view Images. in Lecture Notes on Computer Science (LNCS) 3044, Springer-Verlag, May (2004) 713–720.
12. Tong-Yee Lee, and Po-Hua Huang: Fast and Intuitive Metamorphosis of 3D Polyhedral Models Using SMCC Mesh Merging Scheme. IEEE Transactions on Visualization and Computer Graphics Vol. 9, No. 1, Jan-March (2003) 85–98.
13. Chao-Hung Lin, and Tong-Yee Lee: Metamorphosis of 3D Polyhedral Models Using Progressive Connectivity Transformations. IEEE Transactions on Visualization and Computer Graphics, Vol. 11, No.1, Jan-Feb. (2005) 2–12.