Aura Texture

Xuejie Qin Yee-Hong Yang
{xuq, yang}@cs.ualberta.ca
Department of Computing Science
University of Alberta

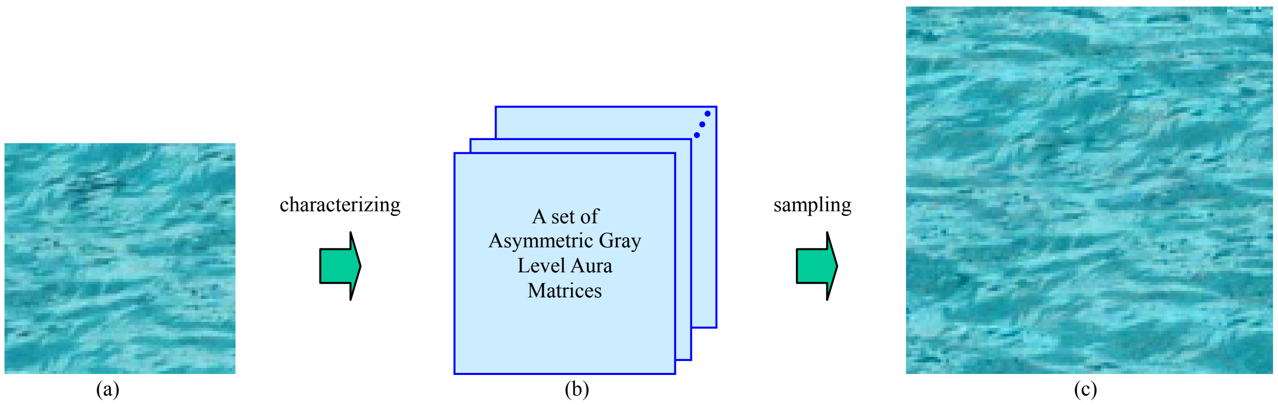

Figure 1: The basic idea of the approach of aura texture synthesis. The input example (a) is first characterized by a set of Asymmetric Gray Level Aura Matrices (AGLAMs) (b), and then the AGLAMs are used to generate an output texture (c).

Abstract

This paper presents a new technique, called aura texture, for generating synthetic textures from input examples. The basic idea of the new approach is to model textures using Asymmetric Gray Level Aura Matrices (AGLAMs), which give (the proof is in a recently submitted paper to the IEEE CVPR 2005 by the authors, and a copy of the paper is included in the supplemental material that accompanies the paper) the necessary and sufficient information of a given texture. For an input texture, the aura texture approach first calculates a set of characteristic AGLAMs to represent the texture. Then, without requiring any further information from the input, it synthesizes an output texture (initialized as white noise) with similar characteristic AGLAMs as those of the input. The experimental results have shown that the new technique can successfully synthesize a wide range of textures and the results are comparable to those of the existing techniques. In addition, based on a metric distance measure, the new technique is able to automatically evaluate the results and determine whether or not the output is a successful synthesis of the input. None of the existing techniques has the ability to evaluate their synthesis results.

1 Introduction

In computer graphics, texture synthesis from examples has been widely recognized as an important tool in generating realistic textures for rendering complex graphics scenes. One major problem of existing example-based texture synthesis techniques is that the output textures are often generated by using some characteristics of input examples, which may not represent the input texture appropriately. For instance, in existing feature-matching approaches [3, 8, 16, 29, 35], a set of filter responses at multiple scales and orientations are used to characterize an example texture. However, as suggested by Zhu et al. in their FRAME (Filters, Random Fields and Maximum Entropy) model [39], it requires an infinite number of filters (each filter is as big as the given texture image) to model a given texture with the necessary and sufficient information. In addition, it is not an easy task to select the filters or to determine the number of filters to model a typical texture [39]. Because of using ambiguous definitions of textures, existing synthesis techniques cannot determine whether or not the synthesis result is acceptable. Visual inspection is the only way to evaluate the synthesis results.

To address the above problems, this paper presents a new technique, called *aura texture*, which synthesizes textures using Asymmetric Gray Level Aura Matrices (AGLAMs). The work is based on a new mathematical framework of AGLAMs [1], which is recently proposed by the authors for texture modeling. We prove that [1] the AGLAMs of a given texture image have the necessary and sufficient information to represent the texture, which, to our best knowledge, has not been addressed by any of the existing texture models.

The main idea of our approach (see Figure 1) is first to characterize a given example texture by a set of characteristic AGLAMs (for the definition, see Section 3), called characteristic AGLAMs. Then, by sampling from the characteristic AGLAMs only, our method generates an output texture similar to the input with similar characteristic AGLAMs. This is done by iteratively modifying the gray level of each pixel in the output image, which is initialized as a random noise image, until the distance between the corresponding characteristic AGLAMs of the output and those of the input is small enough or until the quality of the output texture cannot be improved any further.

A new distance measure (Section 4.2) defined between the characteristic AGLAMs of the input and those of the output texture is used to evaluate the synthesis result. It is proved in another paper by the authors [1] that the new measure is a one-to-one metric in the sense that a zero distance between two images of the same size will guarantee that they are identical. For texture synthesis, however, the property of similarity between two textures is more interesting than the identicalness between them. Based on the one-to-one metric property, we demonstrate that the new measure can be used to evaluate the synthesis results to determine whether or not the output is similar to the input texture. In fact, if the distance value is below a threshold value (for the discussion on the threshold value, see Section 5.2), then the result is considered as a success. Otherwise, it is a failure. Note that this one-to-one metric property is crucial for measuring the similarity between textures. Without this property, which is the case in existing techniques (e.g. [8, 29, 35, 39]), a less similar texture image might be given a higher degree of similarity to the input. Hence, existing techniques only show some synthesized results without evaluating them.

The experimental results have shown that the aura texture can generate acceptable results on a broad range of textures. Compared with existing example-based synthesis techniques, the advantages of the aura texture are: (1) provides accurate representations of example textures, (2) is able to evaluate the results, (3) requires no filters.

The paper is organized as follows. The related work is described in the next section, and then followed by the background knowledge in Section 3. In Section 4, we present the approach of aura texture synthesis. The experimental results and their evaluation are presented in Section 5. Limitations of the aura texture and future work are described in Section 6. Finally, conclusions are given in Section 7.

2 Related Works

Since Julesz's pioneering work in texture analysis [18], various approaches have been proposed for texture analysis and synthesis. One of the most influential approaches is the MRF models [6, 13]. Only a limited range of textures can be modeled with earlier MRF techniques because of the limited size of the cliques and of the low-order statistics used in modeling. To address these problems, Zhu et al. propose the FRAME model, which incorporates filtering theory into the MRF models to synthesize a wider range of textures [39]. The conventional MRF texture models are also generalized by Popat and Picard to the cluster-based probability model [28] and by Paget to the strong MRF model [25] for modeling textures with high order statistics. Different from Zhu et al.'s FRAME model, both approaches are nonparametric. In general, MRF models are slow because of the expensive local probability construction (normally based on exponential functions) at each pixel location during the sampling. To speed up, nonprobabilistic pixel-based sampling techniques [2, 10, 16, 35] are proposed by a number of researchers, which are further improved by the patch-based sampling techniques [11, 20, 21, 34].

Techniques are also developed to synthesize textures by matching features in multiple scales and orientations, pioneered by Heeger and Bergen's work [16] using a global histogram-matching strategy. Later, in the work of Simoncelli and Portilla [29], it is shown that new textures can be synthesized by matching the corresponding joint statistics of complex wavelet coefficients between the input and output image pyramids. Rather than using global joint statistics, DeBonet and Viola use joint occurrence of local features in multiresolutions to model texture images [9]. Their approach has been generalized by Bar-Joseph et al. to texture mixture and video texture using statistical learning [3].

Another influential approach called Gray Level Cooccurrence Matrices (GLCMs) [7, 14] can be used as a powerful tool for texture analysis, segmentation, classification, and synthesis. The disadvantage of the GLCMs is that they contain cooccurrence information between two pixels only, and thus cannot capture the spatial relationship between three or more pixels in the image. This problem can be addressed by using Gray Level Aura Matrices (GLAMs) [12], which incorporate neighborhood systems to model the relationship between the target pixel and its neighboring pixels, and thus can capture the relationship between any number of pixels. However, the neighborhood systems in Elfadel and Picard's aura framework [12] are assumed to be symmetric, and hence cannot model anisotropic textures.

To address the above problem, in [1], the authors propose a mathematical framework based on Asymmetric Gray Level Aura Matrices (AGLAMs), which allows neighborhood systems of arbitrary shapes to model general textures. It is demonstrated that the AGLAMs of a given image have the necessary and sufficient information to represent a given texture. Using AGLAMs, a new distance measure [1] can be defined to measure the similarity between two texture images. It is proved that the AGLAM-based similarity measure is a one-to-one metric. None of the existing proposed measures guarantees this one-to-one condition.

The work in this paper is based on the newly proposed AGLAM-based mathematical framework [1]. We demonstrate that given a texture sample, a synthesized texture can be generated by sampling from a compact set of AGLAMs calculated from the input directly without requiring any filters. Compared with existing pixel-based sampling techniques, the proposed approach is able to synthesize a wider range of textures. Another advantage of the new approach is that it is able to evaluate the results based on the AGLAM-based similarity measure (Section 4.2).

There are also techniques for synthesizing 3D textures, for example, texture mapping [15, 19, 23, 32, 37], procedural texturing [27], and example-based 3D texturing [5, 17, 16, 22, 33, 36, 38]. In this paper, we focus our attention to 2D texture synthesis only.

3 Background Knowledge

The aura texture is based on the aura concepts [12] and the AGALM theory [1], which are briefly described below for ease of reference. For the details, the interested reader is referred to the original papers [1, 12].

Aura: [12] Given an image X defined on a finite rectangular lattice S with a neighborhood system $\boldsymbol{N} = \{N_s, s \in S\}$, where N_s is the neighborhood at site s. Given two subsets $A, B \subseteq S$, the aura of A with respect to B for neighborhood system $\boldsymbol{N}$, denoted as $\vartheta_B(A, \boldsymbol{N})$ (or simply $\vartheta_B(A)$), is given by:

$$\vartheta_B(A) = \vartheta_B(A, \boldsymbol{N}) = \bigcup_{s \in A}(N_s \cap B)\,. \qquad (1)$$

Aura Measure: [12] The aura measure of A with respect to B, denoted as $m(A, B, \boldsymbol{N})$ (or simply $m(A, B)$), is given by:

$$m(A, B) = m(A, B, \boldsymbol{N}) = \sum_{s \in A} |N_s \cap B|\,, \qquad (2)$$

where for a given subset $A \subseteq S$, $|A|$ is the total number of elements in A.

Gray Level Aura Matrix (GLAM): [12] Let $\boldsymbol{N}$ be a neighborhood system over S with an arbitrary shape, and $\{S_i, 0 \le i \le G-1\}$ be the gray level sets of an image over S, then the gray level aura matrix of the image over $\boldsymbol{N}$, denoted by $\boldsymbol{A}(\boldsymbol{N})$ (or simply $\boldsymbol{A}$), is given by:

$$\boldsymbol{A} = \boldsymbol{A}(\boldsymbol{N}) = [a(i,j)] = [m(S_i, S_j)], \quad (3)$$

where G is the total number of gray levels in the image, $S_i = \{s \in S \mid x_s = i\}$ is the gray level set corresponding to the i^{th} level, and $m(S_i, S_j)$ is the aura measure between S_i and S_j given by Eq. 2, and $0 \le i, j \le G-1$.

When the neighborhood system $\boldsymbol{N}$ in a GLAM is symmetric, anisotropic textures cannot be well modeled using GLAMs. For general texture modeling, GLAMs must be used with asymmetric neighborhood systems [1].

AGLAM* & characteristic AGLAM*: [1] An *AGLAM* on S is a *GLAM* computed from an asymmetric neighborhood system $\boldsymbol{N}$. A *characteristic AGLAM* is an AGLAM computed from a single site neighborhood system.

The main theory on AGLAMs is presented in the following theorem. For the proof, the reader is referred to the authors' paper [1].

Theorem Two images of the same size are identical if and only if their corresponding *characteristic* AGLAMs on all possible single site neighborhood systems are identical.

Intuitively, the aura $\vartheta_B(A)$ gives an interpretation of how set B is present in the neighborhood of set A. The aura measure $m(A,B)$ evaluates the amount of mixing between set A and B. A large value of $m(A,B)$ implies that set A and B are mixed together. A small value implies that A and B are separate from each other.

The GLAM, a generalization of the gray level cooccurrence matrix, indicates how much of each gray level is present in the neighborhood of each other gray level. Using asymmetric neighborhood systems, AGLAMs are able to model textures with sufficient and necessary information [1]. For a given texture example, in this paper, we demonstrate that a small set of characteristic AGLAMs can be used to characterize and to synthesize the texture faithfully. In the next section, we describe the approach of aura texture synthesis.

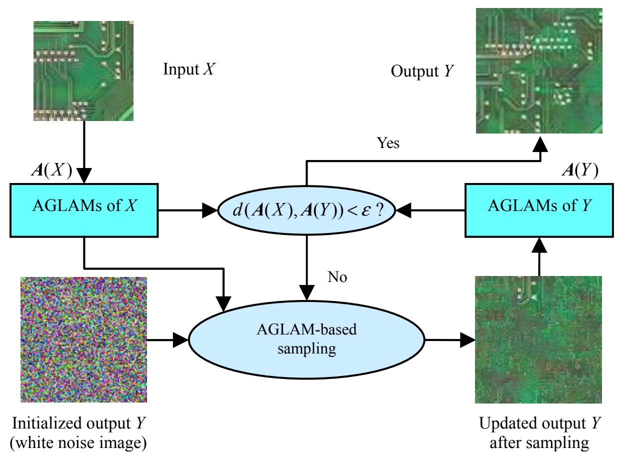
Input X
Output Y
Yes
$A(X)$
$A(Y)$
AGLAMs of X
$d(A(X), A(Y)) < \varepsilon$?
AGLAMs of Y
No
AGLAM-based sampling
Initialized output Y (white noise image)
Updated output Y after sampling

Figure 2: An overview of the approach of aura texture synthesis.

4 Aura Texture Synthesis

Figure 2 gives an overview of the aura texture synthesis approach. Given an input texture X, its characteristic AGLAMs $\boldsymbol{A}(X)$ are computed using an algorithm described later. The output texture Y is initialized as a white noise image, and its characteristic AGLAMs $\boldsymbol{A}(Y)$ are computed. Then, an AGLAM-based sampling procedure is employed to iteratively update the output until the distance between the corresponding characteristic AGLAMs of the output and those of the input is small enough or until there is no further change in pixel's gray level values in the output. During an iteration of the sampling process, the gray level of a pixel in the output Y is modified such that the newly assigned gray level to the pixel will decrease (at least not increase) the distance between the characteristic AGLAMs of the output and of the input.

4.1 Characteristic AGLAMs

For texture synthesis, we only want the output to look similar to (rather than exactly the same as) the input. According to the theorem in Section 3, it is reasonable to have the following assumption: two texture images (not necessarily the same size) are similar if and only if their corresponding characteristic AGLAMs computed from a large enough neighborhood system are close enough.

In this paper, we relax the neighborhood size as a tunable parameter. In general, the larger the texture structures in an image, the bigger the neighborhood size is. For a given texture image, the AGLAMs computed from a given neighborhood system are called the characteristic AGLAMs of the texture. If the neighborhood size is n, then the number of AGLAMs used to characterize the texture is $(n \times n - 1)$.

In our work, a fast algorithm similar to the one in Qin and Yang's work [31] is used to efficiently compute an AGLAM by going through each pixel of the image in one pass. In particular, it works as follows. Initialize each entry of the AGLAM $\boldsymbol{A} = [m(S_i, S_j)]$ to zero, i.e. $m(S_i, S_j) = 0$ for $0 \le i, j \le G-1$. For each site s, let g be its gray level, we check each site r in the neighborhood N_s, and let g' be its gray level. Then we increment the value of $m(S_g, S_{g'})$ by 1. The algorithm stops when all the sites in the image have been processed. Once the characteristic AGLAMs are computed for the input texture, they are stored and used as the only representation of the input to generate the output during synthesis. In other words, the input texture itself will not be needed any more once its characteristic AGLAMs are computed.

4.2 Similarity Measure

During synthesis, it is important to have an accurate measure to determine how close the output texture matches the input. In our work, the similarity between two texture images is measured by the sum of the distances between their corresponding characteristic AGLAMs, where the distance of two matrices is the Manhattan distance of the two matrix vectors. Precisely, given two texture images X and Y defined on S. Let $\boldsymbol{A}(X) = \{\boldsymbol{A}_i \mid 0 \le i \le m\}$ and $\boldsymbol{A}(Y) = \{\boldsymbol{B}_i \mid 0 \le i \le m\}$ be their corresponding characteristic AGLAMs, then the similarity measure between X and Y is given by:

$$d(X,Y) = d(\boldsymbol{A}(X), \boldsymbol{A}(Y)) = \frac{1}{m}\sum_{i=0}^{m} \| \boldsymbol{A}_i - \boldsymbol{B}_i \|, \quad (4)$$

where for a given matrix $\boldsymbol{A} = [a(i,j)]_{0 \le i,j \le n}$, $\| \boldsymbol{A} \| = \sum_{i,j=0}^{n} | a(i,j) |$.

For two images of the same size, it is proved [1] that if the neighborhood system used to calculate the AGLAMs is large enough, then the distance measure defined in Eq. 4 is one-to-one in the sense that a distance measure of zero guarantees that the two images are identical. If two images are of different sizes, Eq. 4 can also be used to measure the distance between them provided that their characteristic AGLAMs are normalized. An AGLAM $\boldsymbol{A} = [a(S_i, S_j)]$ is *normalized* if $\sum_{i,j=0}^{m} a(S_i, S_j) = 1$. In the rest of the paper, we assume that all AGLAMs are normalized. This one-to-one property of the AGLAM-based measure enables our algorithm to evaluate the synthesis results automatically. As far as we know, none of the existing techniques has this feature.

4.3 AGLAM-Based Sampling

The AGLAM-based sampling procedure iteratively modifies the output such that its characteristic AGLAMs match those of the input. In the beginning, the output texture is initialized as a white noise image (see Figure 2). During each iteration of the sampling, each pixel of the output is visited *randomly* once, and its gray level is modified so that the characteristic AGLAMs of the output get closer to those of the input. More precisely, when visiting a pixel, the algorithm first finds the candidate set of all gray levels (different from the current pixel value) that decrease or at least do not increase the AGLAM-based distance (defined in Eq. 4) between the output and the input. Then it randomly chooses a gray level from the candidate set and sets the pixel value to the newly selected gray level. Note that even when a gray level does not decrease the distance, the algorithm also includes it into the candidate set in order to increase the randomness in the output. It is possible that the candidate set is empty at the end of search, which implies that any gray level different from the current pixel value will increase the distance, in which case, the pixel retains its current gray level, and the algorithm goes to the next target pixel. When the AGLAM-based distance between the output and the input is below a threshold or there is no change in gray level values in any pixel of the output, the sampling process returns the output texture as the final result.

The major computation cost of the aura texture synthesis is spent on recalculating the characteristic AGLAMs of the output and the AGLAM-based distance during the AGLAM-based sampling. A brute force method would perform a fresh recalculation each time with a cost of $O(m * (np^2 + G^2))$, where *m* is the size of the neighborhood system, *np* is the number of pixels in the output, and *G* is number of gray levels in the image. A more efficient way is to perform an iterative update based on existing information, which can be done with a computation cost of $O(m * p)$ because when a pixel changes its gray level value, only its neighboring pixels will be affected (for the proof of it, see [1]). To achieve this, however, the algorithm must store the characteristic AGLAMs of the input and of the output as well as the distance between each pair of the corresponding AGLAMs of the input and of the output.

4.4 Color Image

For color input texture images, one cannot simply apply the above basic algorithm to each of the RGB channels separately since the RGB components of a color image are dependent on each other. Before applying the basic aura texture synthesis algorithm, a color-space transformation $\boldsymbol{T}$ based on the singular value decomposition technique (SVD) [30] is used to transform the *R*, *G*, and *B* components of an color image into three independent components R', G', and B' in another color space. After this RGB-color-decorrelation step, the basic synthesis algorithm is applied to each of the independent color components R', G', and B' to generate three output textures in the transformed color space, which are then transformed back (using the inverse transformation of $\boldsymbol{T}$) into the RGB color space to produce the final synthesized color texture image. A detailed algorithm of the RGB color decorrelation can be found in Heeger and Bergen's paper [16].

5 Experiments

5.1 Results

Figure 6 gives some comparison results of texture synthesis, where images in column (a) are the input texture samples, and images in the last four columns (b) – (e) are the synthesized results of: our algorithm, the Heeger and Bergen algorithm [16], the Wei and Levoy algorithm [35], and the Liang et al. algorithm [21]. We implement both Heeger's and Wei's algorithms, in which Heeger's algorithm is based on the steerable pyramid [16] and Wei's algorithm is based on the Gaussian pyramid [35]. The results of our algorithm are generated using 48 characteristic AGLAMs calculated from a square window of size 7x7 around a target pixel. The results of the Heeger's and Wei's algorithms are generated with three levels of image pyramids. The results for the Liang et al.'s algorithm are taken from Paget's website [26].

As shown in Figure 6, Heeger's algorithm is able to capture the overall appearance of a given texture sample, but fails to capture the local structures in the texture because of the global histogram-matching scheme used in the algorithm. Wei's algorithm is able to capture the details of a given texture using a pixel-based sampling scheme, but has a smoothing effect in the output because of the inaccurate SSD measure (sum of squares differences) used to measure the similarity between the output and the input and the Gaussian pyramid used to represent a texture image. Although, Liang's algorithm can generate good results, our algorithm generates better results for the input textures in the 1st, 2nd, and 5th rows. For other input textures in the figure, the results for our algorithm are comparable to those of the Liang's algorithm. More results of the aura texture synthesis can be found in Figure 7 and in the supplemental material that accompanies the paper.

In our approach, the neighborhood size is an important parameter that affects the synthesis results. In general, an image containing large structural textures (see textures in the 1st column in Figure 7) requires a relatively large neighborhood size. For a given input texture, different synthesis results can be generated with different neighborhoods sizes. Figure 3 below gives an example texture and its synthesized textures generated with different neighborhood sizes. It is an interesting future research topic to systematically determine the optimal neighborhood size (e.g. 11x11 for the input texture shown in Figure 3) for a given input texture image to obtain the best run-time performance.

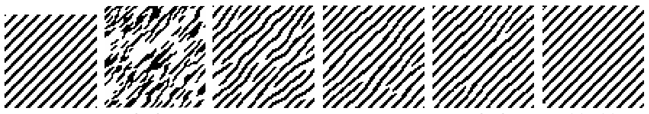

Figure 3: An example of the synthesis results using the neighborhoods of different sizes given under each output.

5.2 Evaluating Synthesis Results

One significant advantage of the aura texture approach over existing approaches is that the AGLAM-based distance measure defined in Eq. 4 can be used to evaluate the synthesis result to determine whether or not the output looks similar to the input. By our experimental results, we found that if two texture images have a distance value greater than 1.0, then they are dissimilar. If the value is below 0.1, then the output is assured similar to the input. However, if the distance value is between 0.1 and 1.0, then the similarity between the two textures is difficult to determine, in this case we consider the output with a distance value below 0.5 a success and a failure otherwise. This observation is made by our extensive experiments. Figure 4 gives an example to demonstrate this point. Note that in Figure 6, each output texture has a number beside it to show its AGLAM-distance to the input.

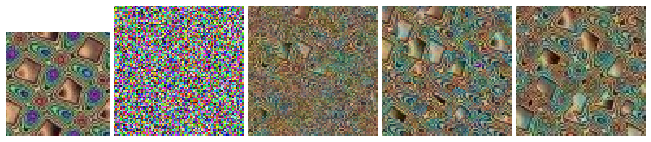

Figure 4: An example using AGLAM-based distance measure to evaluate the synthesized results against the input.

5.3 Acceleration

For acceleration, we extend our algorithm so that it can perform texture synthesis in multiresolutions, similar to the pyramid method used in the Heeger and Bergen's work [16]. However, by our experience, we find that the filtering process only complicates our algorithm. Thus we have used a non-filter-based method, called local decimation [24] to build the multiresolution representation of a given image. For an input color texture sample of size 64x64 with 120 characteristic AGLAMs and an output color texture of size 128x128, the average running time in single resolution is about 2 hours on a 1.4GHz Penntium 4 PC running Windows XP Professional. With a multiresolution scheme of 4 levels and 24 AGLAMs used for each level, the running time is reduced to about 10 minutes. For color images, our algorithm can be extended to synthesize the three independent color channels in parallel after the step of color-space transformation as described in Section 4.4. In this case, the above running time can be further reduced to about 3 minutes.

6 Limitations and Future Work

One limitation of the current implementation of the aura texture synthesis algorithm is the gray level update scheme during the sampling as described in Section 4.3. It is quite possible that after a few iterations, the number of candidates of possible gray levels for a target pixel is less than 3, which may sometimes cause the gray level values for pixels in the output texture to quickly converge to local minima, and thus generate visible seams in the output textures as shown in Figure 5.

In this case, fortunately, the AGLAM-based distance measure between the output and the input cannot decrease any further, and a large distance value (normally above 0.5) is returned to indicate a failure (see Figure 5). However, future research should be carried out to address this problem. One possible solution is to extend the current single-pixel search scheme to a multiple-pixel search scheme during the sampling so that the convergence to the local minima can be avoided as much as possible. Although a direct search in multiple-pixel directions is not practical, Boykov et al. have developed an efficient algorithm to do this based on graph cuts [4]. We are currently considering their method to address this local minima problem in our algorithm.

Another interesting future work is to extend our approach to generate 3D textures. One possible way to do this is to generalize the AGLAM theory [1] to 3D space by defining the concept of 3D neighborhood systems. Given an input texture sample (or multiple samples), one can synthesize a 3D volume texture by coercing a white noise volume to have the same characteristic AGLAMs as those of the inputs by noting the fact that AGLAMs in both 2D and 3D space have the same dimensions.

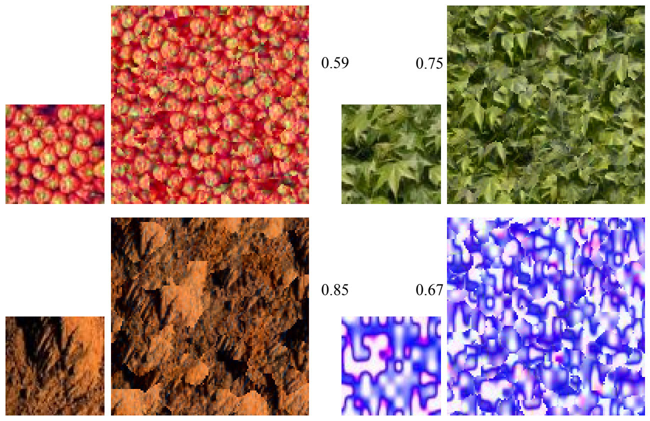

Figure 5: Example of visible seams in the synthesized textures. The number beside each output texture is its distance measure calculated using Eq. 4. Since those values are greater than 0.5, the output textures are considered as failures based on the evaluation criterion described in Section 5.2.

7 Conclusions

In this paper, a new texture synthesis approach, called aura texture, is proposed. Given an input texture, our algorithm first calculates a set of characteristic AGLAMs to represent the texture, and then generates the synthesized texture by sampling only the AGLAMs of the input without requiring any further information. The experimental results show that the new technique can successfully synthesize a wide range of textures and is comparable to the existing techniques. In addition, based on a new distance measure, the new technique is able to automatically evaluate the results and determine whether the output is a successful synthesis of the input. None of the existing techniques has the ability to evaluate their synthesis results.

References

1. Anonymous, *Representing Texture Images using Asymmetric Gray Level Aura Matrices.* Submitted to IEEE CVPR 2005.
2. Ashikhmin, M., *Synthesizing Natural Textures.* The ACM Symposium on Interactive 3D Graphics, 2001: p. 217-226.
3. Bar-Joseph, Z., et al., *Texture Mixing and Texture Movie Synthesis Using Statistical Learning.* IEEE TVCG, 2001. **7**(2): p. 120-135.
4. Boykov, Y., O. Veksler, and R. Zabih, *Fast Approximate Energy Minimization via Graph Cuts.* PAMI (also in ICCV 99), 2001. **23**: p. 1222-1239.
5. Chen, Y., et al., *Shell Texture Functions.* ACM SIGGRAPH, 2004. **23**(3): p. 343-353.
6. Cross, G.C. and A.K. Jain, *Markov Random Field Texture Models.* PAMI, 1983. **5**(2): p. 25-39.
7. Davis, L.S., S.A. Johns, and J.K. Aggarwal, *Texture Analysis Using Generalized Cooccurence Matrices.* PAMI, 1979. **1**(3): p. 251-259.
8. DeBonet, J.S., *Multiresolution Sampling Procedure for Analysis and Synthesis of Texture Images.* Siggraph, 1997: p. 361-368.
9. DeBonet, J.S. and P. Viola, *Texture Recognition Using a Non-parametric Multi-Scale Statistical Model.* IEEE CVPR, 1998: p. 641-647.
10. Efros, A. and T. Leung, *Texture Synthesis by Non-Parametric Sampling.* ICCV, 1999: p. 1033-1038.
11. Efros, A.A. and W.T. Freeman, *Image Quilting for Texture Synthesis and Transfer.* Siggraph, 2001: p. 341-346.
12. Elfadel, I.M. and R.W. Picard, *Gibbs Random Fields, Cooccurrences, and Texture Modeling.* PAMI, 1994: p. 24-37.
13. Geman, S. and D. Geman, *Stochastic relaxation, Gibbs distributions, and the Bayesian Restoration of Images.* PAMI, 1984. **6**: p. 721-741.
14. Haralick, R.M., K. Shanmugan, and I.H. Dinstein, *Textural Features for Image Classification.* IEEE Trans. Syst. Man Cybern., 1973: p. 610–621.
15. Heckbert, P.S., *Fundamentals of Texture Mapping and Image Warping. Master's Thesis*, in *Dept. of Elec. Eng. and Compt. Sci.* 1989, Univ. of California: Berkeley.
16. Heeger, D.J. and J.R. Bergen, *Pyramid-Based Texture Analysis/Synthesis.* Siggraph, 1995: p. 229-238.
17. Jagnow, R., J. Dorsey, and H. Rushmeier, *Stereological Techniques for Solid Textures.* ACM SIGGRAPH, 2004. **23**(3): p. 329-335.
18. Julesz, B., *Visual Pattern Discrimination.* IEEE Transactions on Information Theory, 1962: p. 84-92.
19. Kraevoy, V., A. Sheffer, and C. Gotsman, *Matchmaker: Constructing Constrained Texture Maps.* Siggraph, 2003. **22**(3): p. 326-333.
20. Kwatra, V., et al., *Graphcut Textures: Image and Video Synthesis using Graph Cuts.* Siggraph, 2003. **22**(3): p. 227-286.
21. Liang, L., et al., *Real-Time Texture Synthesis by Patch-Based Sampling.* TOG, 2001: p. 127-150.
22. Liu, X., et al., *Synthesis and Rendering of Bidirectional Texture Functions on Arbitrary Surfaces.* IEEE Transactions on Visualization and Computer Graphics, 2004. **10**(3): p. 278-289.
23. Oliveira, M.M., G. Bishop, and D. McAllister, *Relief texture mapping.* Siggraph, 2000: p. 356-368.
24. Paget, R. and I.D. Longstaff, *Texture Synthesis via a Noncausal Nonparametric Multiscale Markov Random Field.* IEEE TIP, 1998. **7**(6): p. 925-931.
25. Paget, R., *Strong Markov Random Field Model.* PAMI, 2004. **26**(3): p. 408-413.
26. Paget, R., *Image Source for Liang et al.'s Texture Synthesis Results.* http://www.vision.ee.ethz.ch/~rpaget, 2004.
27. Perlin, K., *An Image Synthesizer.* Siggraph, 1985. **19**(3): p. 287-296.
28. Popat, K. and R.W. Picard, *Cluster-Based Probability Model and its Application to Image and Texture Processing.* IEEE TIP, 1997. **6**(2): p. 268-284.
29. Portilla, J. and E.P. Simoncelli, *A Parametric Texture Model Based on Joint Statistics of Complex Wavelet Coefficients.* IJCV, 2000. **40**(1): p. 49-71.
30. Press, W.H., et al., *Numerical Recipes in C++: The Art of Scientific Computing (2nd ed).* Cambridge University Press, 2002.
31. Qin, X. and Y.H. Yang, *Similarity Measure and Learning with Gray Level Aura Matrices (GLAM) for Texture Image Retrieval.* IEEE CVPR, 2004: p. 326-333.
32. Soler, C., M. Cani, and A. Angelidis, *Hierarchical Pattern Mapping.* Siggraph, 2002. **21**(3): p. 673-680.
33. Turk, G., *Texture Synthesis on Surfaces.* Siggraph, 2001: p. 347-354.
34. Wang, B., et al., *Efficient Example-Based Painting and Synthesis of 2D Directional Texture.* IEEE Trans. Vis. Comput. Graph., 2004. **10**(3): p. 266-277.
35. Wei, L. and M. Levoy, *Fast Texture Synthesis Using Tree-Structured Vector Quantization.* Siggraph, 2000: p. 479-488.
36. Wei, L.Y. and M. Levoy, *Texture Synthesis over Arbitrary Manifold Surfaces.* Siggraph, 2001: p. 355-360.
37. Zelinka, S. and M. Garland, *Interactive Texture Synthesis on Surfaces Using Jump Maps.* Eurographics Symposium on Rendering, 2003: p. 90-96.
38. Zhang, J., et al., *Synthesis of Progressively-Variant Textures on Arbitrary Surfaces.* Siggraph, 2003. **22**(3): p. 295-302.
39. Zhu, S.C., Y. Wu, and D. Mumford, *Filters, Random Fields and Maximum Entropy - Towards a Unified Theory for Texture Modeling.* IJCV, 1998: p. 1-20.

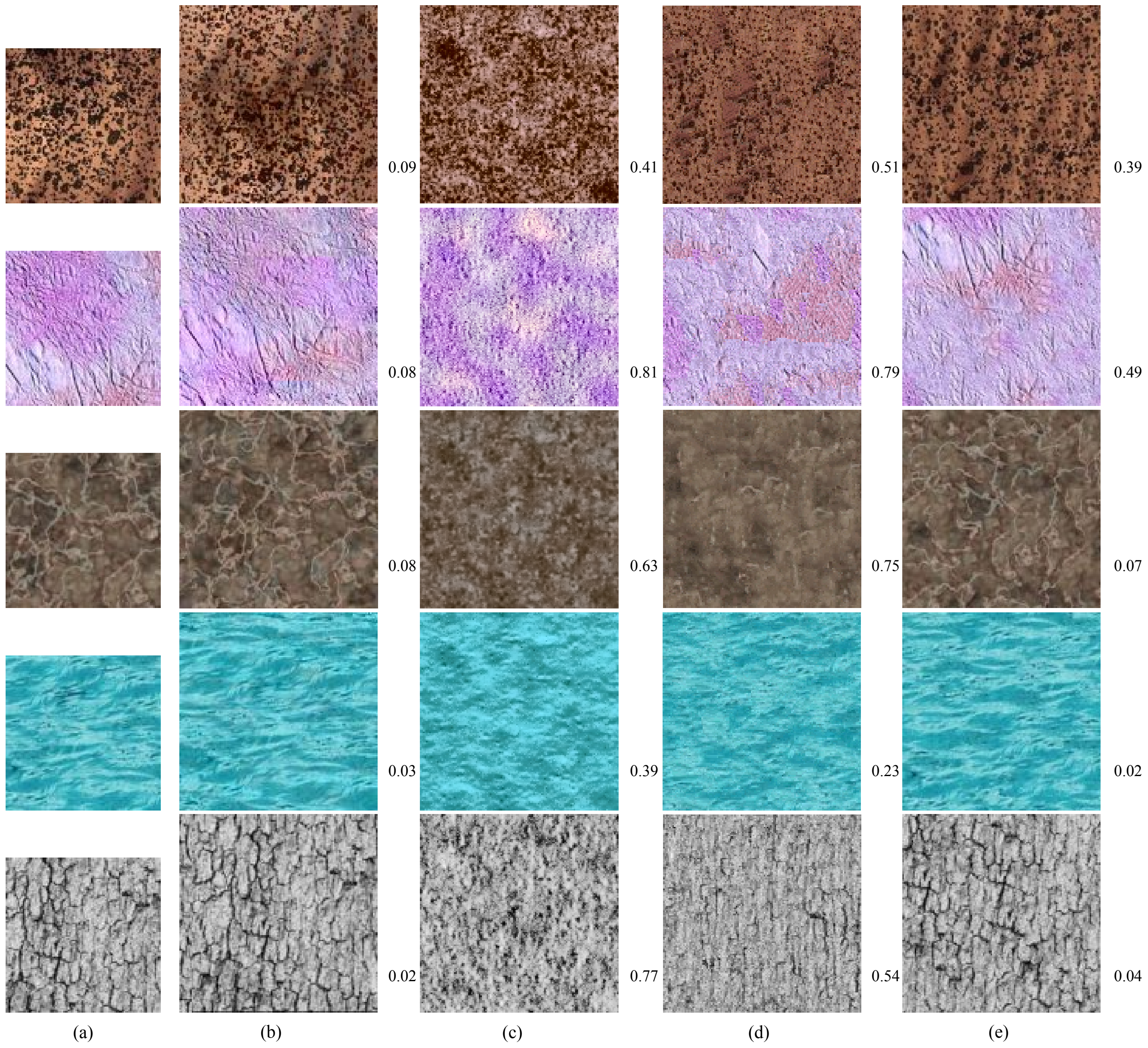

Figure 6: The comparison of results of our approach (column (b)) with Heeger and Bergen's algorithm (column (c)), Wei and Levoy's algorithm (column (d)), and Liang et al.'s algorithm (column (e)), where the input textures are in column (a). For our algorithm, 48 characteristic AGLAMs calculated over a square window of size 7x7 around a target pixel are used to represent the input and to synthesize the output. The results for Heeger's algorithm are generated using steerable pyramids with 3 levels and 4 orientations (i.e. 0, 45 90, and 135 degrees). For Wei's algorithm, a Gaussian pyramid of 3 levels is used to synthesize from a given input texture. The neighborhood sizes used for a Gaussian pyramid are {3x3,1}, {5x5,2}, {7x7,2} from the lowest resolution level to the highest resolution level, where {7x7,2} means a multiresolution neighborhood of 2 levels (with size 7x7 at the higher resolution level and 3x3 at the lower resolution level) is used to generate the highest resolution level. As shown in the figure, Heeger's approach is able to capture the overall appearance of a given texture sample, but fails to capture the local structures in the texture. Wei's algorithm is able to capture the details of a given texture, but has a smoothing effect in the output. Liang's algorithm generates good results for all input textures in the figure. However, our algorithm generates better results for the input textures in the 1st, 2nd, and 5th rows. The number beside each output texture is the value of the AGLAM-based distance measure (see Eq. 4) of the output compared to the input, which shows that the synthesis results from our approach are better than those from Heeger's and the Wei's algorithm, and very comparable to those of the Liang's algorithm.

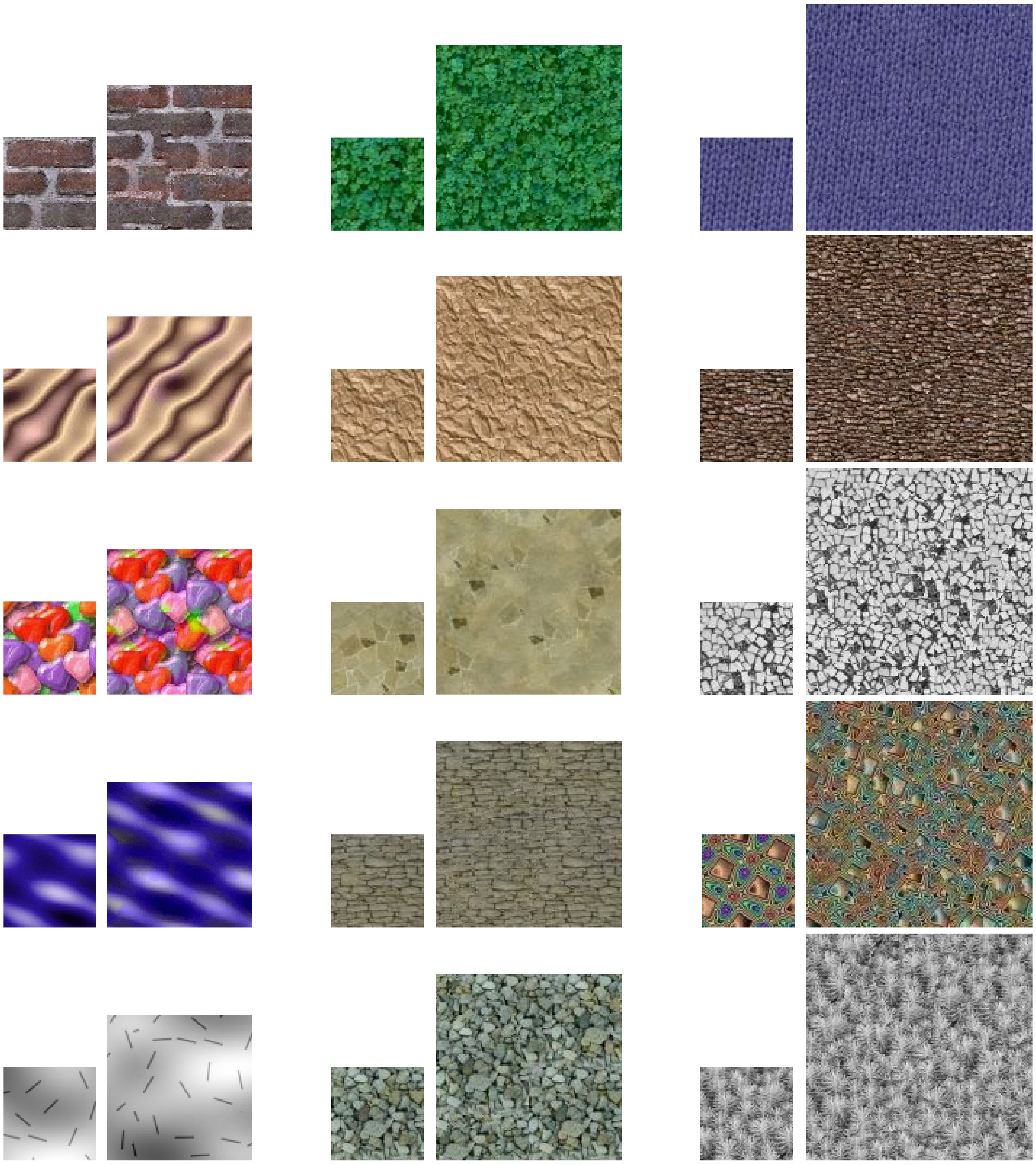

Figure 7: Examples of aura texture synthesis. The smaller image in each pair is the input texture (size 64x64), and the larger image is the synthesized texture. The sizes of output texture in column 2, 4, and 6 are 100x100, 128x128, and 156x156, respectively. Since the textures in the first column contain large structures, 120 characteristic AGLAMs (calculated over a neighborhood system of size 11) are used to generate the output in the second column. The output textures in the 4th and 6th column are generated using 48 characteristic AGLAMs.