Journal of Advanced Mechanical Design, Systems, and Manufacturing

Vol. 6, No. 3, 2012

Texture Synthesis based on Feature Description*

Yan GUI**, Mingang CHEN**, Zhifeng XIE**, Lizhuang MA** and Zhihua CHEN***

** School of Electronic, Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China
E-mail: guiyan122@sjtu.edu.cn

*** Department of Computer Science & Engineering, East China University of Science & Technology, Shanghai, 200237, China

Abstract

Patch-based texture synthesis techniques from a small sample texture generate a new texture image efficiently, but one significant problem is that these existing texture synthesis algorithms suffer from structure mismatching across patch boundaries due to inaccurate similarity measure. In this paper, we propose a desired distance metric to measure structural similarity when especially handling textures with salient features, which has the ability to find best matches during the patch-based sampling. To do this, we introduce the concept of "layers" to describe the correlation of pixels through computing each pixel weight, and construct the orientation fields to capture the inherent direction and variation of texture features through computing the dominant local orientation angle and the degree of anisotropy of each pixel. Our experimental results demonstrate that the proposed method can significantly alleviate structure discontinuities in synthesized textures.

Key words: Texture Synthesis, Feature Description, Layer Correlation, Orientation Field

1. Introduction

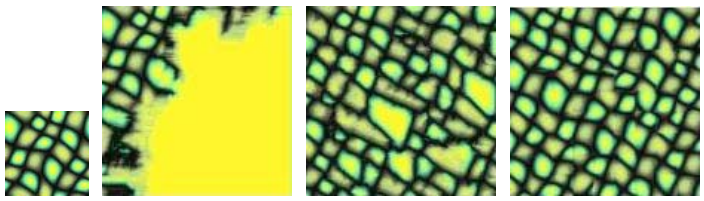
Texture synthesis has been devoted significant work from researchers during the past years. Recently, the example-based texture synthesis[(1)-(17)] has been paid much attention, which can be categorized into two types: pixel-based texture synthesis and patch-based texture synthesis. They always followed the Markov Random Field to analyze the local properties of sample textures, and applied the RGB metric to select the best matches when synthesizing new textures. By using such two types of methods, however, the resulting textures sometimes reveal visual artifacts. For instance, the former methods are easy to cause garbage growing and structure missing (Fig. 1(b) and 1(c)); and structure discontinuity is inevitable by using the latter methods (Fig. 1(d)). The reason is that simple RGB-based metric (summed squared differences of colors) commonly used in similarity measure cannot capture structural information well.

In this paper, our purpose is to design a desirable metric to perform texture synthesis on the general framework of patch-based paradigm. The intrinsic properties of the sample textures, including the inherent position and orientation of texture features, are considered for an accurate similarity measure. We take the spatial correlation of pixels [(17)] into account, and introduce a useful concept called "Layer", namely dividing the boundary region of texture patch into interrelated layers, which describes the degree of the correlation of pixels among layers. The weights of pixels are then computed according to the distance from the

*Received 13 Sep., 2011 (No. 11-0555)
[DOI: 10.1299/jamdsm.6.376]

Copyright © 2012 by JSME

central region of texture patch. When comparing the overlapped region of adjacent patches, the correlation showing a continuous distribution of pixels in position is used to help guide the similarity measure.



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

Fig. 1: The drawbacks of example-based texture synthesis. (a) sample texture; (b) and (c) results from pixel-based texture synthesis; (d) results from patch-based texture synthesis.

On the other hand, our objective is to capture more structural information for a better feature matching. Since textures with salient features are characterized by local selectivity of orientation, the orientation of texture features can be utilized to retrieve the structural layout of textures. The orientation field of textures can be extracted by using an orientated filtering and performing manipulations on the resulting gradient vector field. A. R. Rao et al.[18][19] defined the orientation field of the texture to be comprised of the angle image and coherence image. The angle image captures the dominant local orientation at each pixel of the texture, and the coherence image represents the degree of anisotropy at each pixel of the texture. The similarity between adjacent texture patches is measured based on the constructed the orientation field. Potential benefit of the orientation field applied in our method includes the ability to describe texture features that can vary considerably. Inspired by the above two aspects, we develop a desired distance metric by exploring differences in both layers and the orientation field for selecting the best matches, so as to further improve patch-based texture synthesis.

2. Related work

2.1 Pixel-based texture synthesis

Per-pixel algorithms generate the texture pixel by pixel, which copy each output pixel by using spatial neighborhood comparisons. Efros and Leung[1] first applied a non-parametric sampling process to grow new textures from an initial seed to the whole synthesized region in an inside-out way. However, the method is time-consuming due to the exhaustive pixel searching. Wei and Levoy[2] made further modification by applying a fixed L-shaped neighborhood and synthesizing textures in a scan line order. However, the method has a tendency to cause garbage growing because a pixel might not be sufficient to capture large-scale structures of a texture sample.

On the other hand, many approaches have been proposed to solve the speed and quality issues. Ashikhmin[3] explored spatial coherence to exhibit better time efficiency. Steve Zelinka et al.[4] proposed a real-time texture synthesis with the Jump Map. Paul Harrison[5] presented a non-hierarchical procedure for re-synthesis of complex textures. In addition, GPU-based texture synthesis technique proposed by Lefebvre and Hoppe[6] has been applied to produce textures at very fast rates. However, these methods sometimes caused blurring and discontinuity when handling textures with salient features.

2.2 Patch-based texture synthesis

Patch-based algorithms are an extension from pixel-based approaches because these methods paste large patches instead of pixels to form the final output. Liang et al.[7] developed the patch-based sampling strategy and used feathering technique to blend adjacent patches. Instead of blending, Efros and Freeman[8] used Dynamic programming to stitch overlapped regions of patches together. Kwatra et al.[9] further promoted this idea by

applying Graph-cuts technique. In addition, Cohen et al.[10] presented a Wang Tiles-based method to generate textures. Tiles can be merged together when all shared edges between adjacent tiles have matching colors. Another class of texture synthesis methods proposed by Liu et al.[11][12] was used to deal with near-regular textures. The method well preserves the structural regularity in the synthesized texture.

Recently, many texture features based approaches have been made a significant progress and successfully used in texture modeling. Zhang et al.[13] proposed a texton mask based algorithm for generating texture on surfaces. Wu and Yu[14] developed a texture synthesis method that considers geometric features and colors simultaneously. Lefebvre et al.[15] analyze and transform textures into a low-dimensional and Euclidean appearance space. Dischler et al.[16] proposed a structured texture synthesis with focus on patterns characterized by structural arrangements. However, these techniques have difficulties in extracting features when ambiguous or low contrast texture features present in sample textures, which are strongly limited by the fact that they rely on texture features to keep structural consistency.

3. Feature description based texture synthesis

The major work in this section is attempting to remove structure discontinuity and mismatching. A key observation is that the description of texture features relies on how to determine the position and orientation of each pixel in a given sample texture. For further improving the RGB-based metric, layer-correlation distinguishing and the orientation field based feature matching are adopted in order to better estimate the similarity between texture patches.

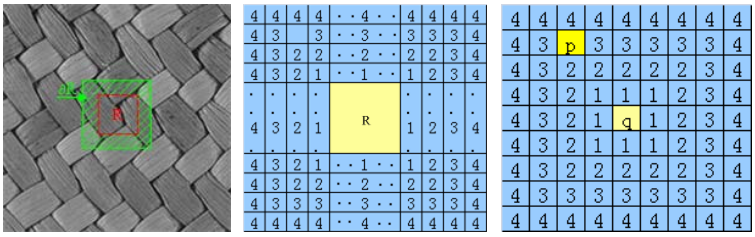
3.1 Layer-correlation based color distinguishing

In most existing texture synthesis algorithms, the L_2 distance is commonly used to measure the similarity of two neighborhoods N_1 and N_2. Such distance metric may get an inaccurate similarity measure, where R, G and B colors are completely different even though the error is minimal between two neighborhoods. The L_2 distance only consider the correlation between two neighborhoods N_1 and N_2, and yet the correlation of pixels in each neighborhood is usually ignored. Note that the neighborhood usually specifies the boundary region of texture patch.

To any pixel p in the sample texture, in general, a pixel q has stronger spatial correlation to pixel p if the distance between them is closer than others. Since the distribution of pixels is discrete at the color space in sample textures, the correlation thus can be used to describe a continuous distribution of pixels. Given that the texture patch is composed of the boundary region ∂R and the central region R (Fig. 2(a)), the boundary region can be partitioned into layers according to the distance from each pixel of the boundary region to the central region. Pixels of the boundary region belong to the same layer when they have equal distance (Fig. 2(b)). Note that the central region of texture patch can be regarded as a pixel because it is surrounded by boundary region. To measure the distance, an 8-neighbor distance function $D_8(p,q) = \max(|i-h|,|j-k|)$ is used in our method, here q is a central pixel, and p is a pixel in neighborhood of pixel q; (i,j) and (h,k) represent the coordinate of these two pixels. Obviously, the shape of the region (Fig. 2(c)) formed by computing 8-neighbor distance is satisfactory to the shape of texture patches, where we specify the texture patch is a square.

Based on aforementioned spatial correlation and layer partition, the correlation of layers[17] in the boundary region of the texture patch can be described as following: ***first***, the degree of the correlation will weaken when one layer gradually deviates from the central region; ***second***, all pixels in one layer have equivalent correlation with respect to the central region. The layer correlation indicates that pixels in the boundary region have different contributions to the central region. The weights of these pixels therefore are computed to

give further characterization for differences in layers. Since the weight $\omega(\cdot)$ of each pixel relies on the pixel to which layer belongs and the total number of pixels in one layer, thus, it is composed of two parts: layer weight $wL(\cdot)$ and intra-layer weight $wIL(\cdot)$.



(a) (b) (c)

Fig. 2: The layer partition for the boundary region of texture patch. (a) shows a texture patch in a sample texture. The green hatched area is the boundary region and the area surrounded by a red rectangle is the central region. (b) shows that the boundary region is partitioned into layers. Layers are marked using numbers. (c) is a region which is formed by 8-neighbor distance. Suppose that the boundary region is divided into four layers.

Given that the boundary region is divided into M layers. Indeed, M is the width of the boundary region. An accumulation $Cost(M)=\sum^{M} L$ crossing each layer L is defined to describe the spatial correlation between the boundary region and the central region. For a pixel p, its layer weight $wL(p_L)$ can be computed as follows:

$$wL(p_L)=\frac{M-L+1}{Cost(M)}, L=1:M \tag{1}$$

$$\sum_{L=1}^{M} wL(p_L)=1 \tag{2}$$

where, the right side of Eq. 1 reflects the degree of the correlation for each layer of boundary region. To computing the intra-layer weight, it is necessary to calculate the contribution of a pixel p with respect to all pixels of one layer L. The number of pixels in the layer are counted as $Num(L)$, and then the intra-layer weight $wIL(p_L^i)$ of a pixel is a inverse of the total number of pixels:

$$wIL(p_L^i)=\frac{1}{Num(L)}, L=1:M, i=1:Num(L) \tag{3}$$

$$\sum_{i=1}^{Num(L)} wIL(p_L^i)=1 \tag{4}$$

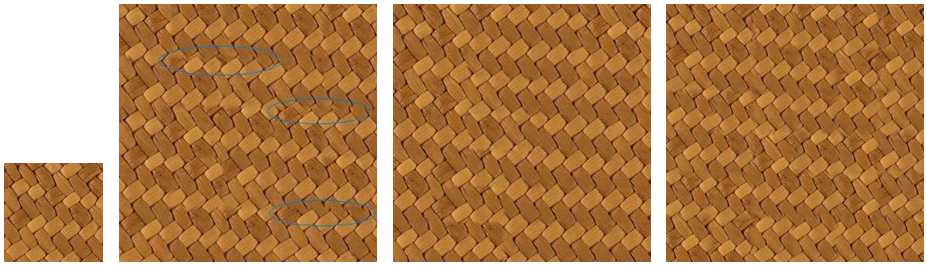
where, p_L^i represents the i th pixel of the L Layer. The value of Eq. 3 depends on the total number of pixels in each layer. Once the layer weight $wL(p_L)$ and the intra-layer weight $wIL(p_L^i)$ are computed, the weight of each pixel p in the boundary region is defined as $\omega(p_L^i)=wL(p_L)\times wIL(p_L^i)$.

Computed by above equations, we obtain the weights of all pixels in the boundary region of texture patches, and then the differences between adjacent texture patches are estimated by using the following defined metric, which considers the layer correlations between neighborhoods B_{In} and B_{Out}:

$$\begin{aligned} D_c(B_{In},B_{Out}) &= \sum_{i=1}^{A}(\omega(p_{In}^i)\cdot p_{In}^i-\omega(p_{Out}^i)\cdot p_{Out}^i)^2 \\ &\approx \sum_{i=1}^{A} sprt(\omega(p^i)\cdot(p_{In}^i-p_{Out}^i)^2) \end{aligned} \tag{5}$$

where, A is the total number of pixels in the boundary region. p_{In}^i and p_{Out}^i represent the values (grayscale or color) of the i th pixel in the respective boundary regions B_{In} and B_{Out}. $\omega(p_{In}^i)$ and $\omega(p_{Out}^i)$ are their weighting coefficients respectively, which are replaced by $\omega(p^i)$ because of equal values. In the second row of Eq. 5, we use $\omega(p^i)$

instead of $\omega^2(p^i)$ because the value of $\omega^2(p^i)$ may be close to zero when the boundary region is wide enough, and we use operator $sprt(\cdot)$ to get the approximate estimation. As the example given in Fig. 3, the color or illumination in the texture elements of the fabric texture (Fig. 3(a)) is too close to distinguish. The synthesized result (Fig. 3(b)) based on RGB measurement shows heavy structure discontinuity, which is marked by blue ellipse. The main reason is that pixels' RGB values only capture how color information affects texture variation. Figure 3(c) illustrates the synthesized result by introducing layer correlation, which can greatly alleviate the related artifacts. Thus, the spatial correlation of pixels used in our implementation is useful to retain the texture features in the synthesized result.



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

Fig. 3: The comparison of the synthesis results. (a) is a sample texture. (b) is the synthesized result by using RGB-based matching. (c) is our synthesized result by using lay correlation distinguishing. (d) is the orientation field based synthesized result. Note that we only use feathering to merge patches in here.

3.2 Orientation field based feature matching

In most cases, the inherent orientation of texture features should be considered in order to search a better feature matching. For example, Wu and Yu[14] focused on the orientations of the tangents of feature pixels in the boundary region of adjacent patches when performing feature matching. Consistent tangent orientation is critical to guarantee geometry continuity, which can regenerate smooth texture features in synthesized results. However, the major limitation is hard to capture ambiguous or low contrast texture features. Unlike the technique depending on feature extraction, the motivation in our method is to construct the orientation fields of sample textures based on an efficient algorithm presented by Rao et al.[18][19]. The most important aspect of the algorithm is that it is provably optimal in estimating the local orientation of textures. Every pixel in the texture is associated with a dominant local orientation and a local measure of the coherence, which form the orientation field of texture features.

The algorithm for estimating the orientation field of a texture presented in paper[19] relies on the gradient of a Gaussian. The gradient of the texture was computed by using finite differences when these textures are grey images. However, for most color sample textures used in this paper, the gradient of each pixel is respectively computed in the red, green, and blue (*rgb*) color channels, and then the gradient combined by the three independent components may cause incorrect results. We thus apply a simple and efficient approach[20] to compute the gradient directly in *rgb* color space, and then obtain the gradient direction θ and the gradient magnitude $F(\theta)$ at each pixel.

For a given sample texture, the whole process for constructing the orientation field can be summarized as follows:

Step 1: smooth the sample texture with Gaussian filtering in order to remove noise, and compute the gradient of the smoothed texture.

Step 2: estimate the dominant local orientation $\hat{\theta}$ for each pixel of the sample texture. In order to compute the orientation at a pixel, one needs to combine the gradient within a

neighborhood of that pixel, in that the gradient directions tend to cancel each other out at intensity texture features. Thus, the dominant orientation $\hat{\theta}_{xy}$ over a $N \times N$ neighborhood, which surrounds the pixel position (x, y) as center, is computed as below:

$$\hat{\theta} = \tan^{-1}\left(\frac{\sum_{i=x-N/2}^{i=x+N/2} \sum_{j=y-N/2}^{j=y+N/2} F^2(\theta_{ij}) \sin 2\theta_{ij}}{\sum_{i=x-N/2}^{i=x+N/2} \sum_{j=y-N/2}^{j=y+N/2} F^2(\theta_{ij}) \cos 2\theta_{ij}} \right) \tag{6}$$

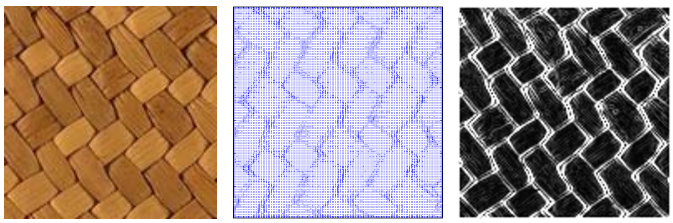
where, i and j are indices of the pixel position in the neighborhood. θ_{ij} and $F(\theta_{ij})$ are the gradient direction and magnitude at (i, j). The estimated orientation angle $\hat{\theta}_{xy}$ at is then $\hat{\theta}_{xy} + \pi/2$ since the gradient vector is perpendicular to the direction of anisotropy.

Step 3: compute a measure of the coherence. To find the coherence of one pixel, we still consider the neighboring pixels. The use of the neighborhood ensures that the pixel has high coherence values at the center of the neighborhood. Given the gradient and the dominant orientation, the measure of coherence is defined as following:

$$\rho = \frac{\sum_{(i,j)\in N} \left\| F(\theta_{ij}) \cos(\theta_{ij} - \hat{\theta}_{xy}) \right\|}{\sum_{(i,j)\in N} F(\theta_{ij})} \tag{7}$$

where, $F(\theta_{ij})\cos(\theta_{ij} - \hat{\theta}_{xy})$ indicates that the gradient magnitude $F(\theta_{ij})$ taken in the direction θ_{ij} is projected onto the unit vector in the dominant orientation $\hat{\theta}_{xy}$ at (x, y).

Thus, the orientation angle and the coherence of all pixels in the sample texture can be obtained when the neighborhood shift over the texture plane in scan-line order. As shown in fig.4, we give the angle image and the coherence image of the given fabric texture respectively. However, different neighborhood sizes have much effect on constructing orientation field. The neighborhood with smaller size is not good at capturing texture features so as that inaccurate orientation field might be obtained. On the other hand, the neighborhood with bigger size will produce many subtle texture features, which may disturb feature matching during synthesis procedure. In addition, it requires heavy computational time when shifting the neighborhood over the texture plane. For balancing the efficiency of constructing the orientation field, the sizes of neighborhood for all sample textures used in this paper are specified as $N = 11$.



(a) (b) (c)

Fig. 4: The constructed orientation field for the given fabric texture (a). (b) shows the estimated orientation field. (c) shows the coherence image, which describe the degree of the anisotropy of each pixel.

Once the sample texture has been expressed as the orientation field, we take it as input for feature matching at the boundary regions of adjacent patches. Since the constructed orientation field is not a vector field, the dominant orientation of the texture at any given pixel identifies the inherent ambiguities in most cases. It is obvious that the measure of the distance taking the subtraction manipulation between two discrete orientation fields is impracticable. Suppose that the orientation field at each pixel is represented by a line segment with a direction, which has length equal to the coherence at that pixel, and the direction is the dominant orientation. The effective measure presented in paper[19] is the area

of the triangle formed by a pair of oriented segments, which are chosen from two orientation fields at a particular position. If the orientations between these two segments are closer to each other, the area tends to zero. In addition, the area becomes zero even if they have totally opposite directions. The area thus can be used to estimate the differences between corresponding orientation fields when performing feature matching. We sum the differences over the boundary regions of adjacent patches:

$$D_o(of_{In}, of_{Out}) = \frac{1}{A}\sum_{i=1}^{A}\left(\frac{1}{2}\rho_{In}^{i}\rho_{Out}^{i} \mid \sin(\hat{\theta}_{In}^{i} - \hat{\theta}_{Out}^{i}) \mid\right) \tag{8}$$

where, of_{In} and of_{Out} are the constructed orientation fields from the boundary regions B_{In} and B_{Out}, respectively. A is the number of pixels in the boundary region. ρ_{In}^{i} and ρ_{Out}^{i} represent the coherence of the i th pixel in the orientation fields of_{In} and of_{Out}. At the same time, $\hat{\theta}_{In}^{i}$ and $\hat{\theta}_{Out}^{i}$ are the dominant orientation.

By considering the orientation filed of the given sample texture, the synthesized result shown in Fig. 2(d) keep better structural regularity than the result by using RGB metric (Fig. 2(b)). The main reason is that the orientation information for features matching can find satisfactory patches which have similar structural layout. However, we still found that there exists texture discontinuity by using the feature matching metric (Eq. 8). On the one hand, although Gaussian filtering is used to remove noise, some primary texture features get blurred. On the other hand, similar texture features are not identical so as that feature misalignments are inevitable.

3.3 Hybrid texture synthesis algorithm

Unlike the RGB-based distance metric used for selecting the best patches, we combine the aforementioned layer correlation and orientation field together to design a new distance metric for obtaining more accurate similarity measure. On the boundary regions of texture patches, the analysis of the correlation among pixels helps to describe the continuity of the distribution of pixels at the color space, and the construction of the orientation field is useful for a symbolic description of texture features. To generate the final synthesized texture, each best patch P selected from the sample texture satisfies the following condition:

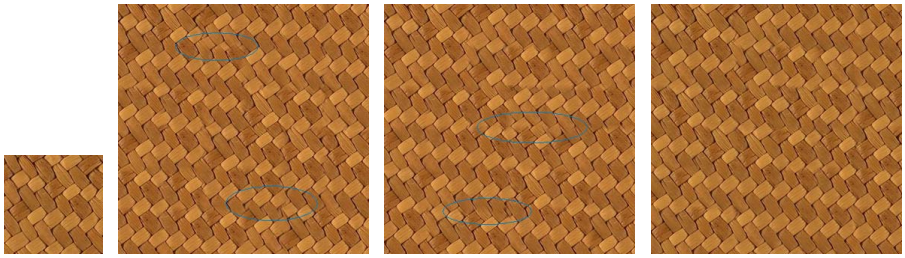
$$P = \min dist(B_P, B_{Out}) \tag{9}$$

$$dist(B_P, B_{Out}) = D_c(B_P, B_{Out}) + \lambda D_o(of_P, of_{Out}) \tag{10}$$

where, B_P and B_{Out} are the boundary regions of the patch of the sample texture and the patch currently being synthesized, respectively. $D_c(B_P, B_{Out})$ represents the differences in colors and $D_o(of_{In}, of_{Out})$ estimates the differences in orientation fields, which are respectively defined by Eq. 5 and Eq. 8. λ is a positive weight designed to adjust the relative importance between the two terms. A balance can be reached when we set $\lambda = 0.13$ in our implementation, and the synthesized textures with high quality can be generated. Figure 5 shows the effects of λ on the final synthesized textures. When $\lambda = 0.6$, although the inherent features of the sample texture can be retained in the synthesized result (Fig. 5(b)), we found that the structural regularity is not well preserved, which is marked by blue ellipses. Compared with the synthesized results (Fig. 5(c) and Fig. 5(d)) when $\lambda = 0.2$ and $\lambda = 0.13$, the former resulting texture reveals obvious structure mismatching because a small feature misalignments may cause structure inconsistency; and the latter shows a better synthesized result, which maintains the most closest visual resemblance with the original sample texture. Figure 3(c) shows the synthesized results when $\lambda = 0$.

For traditional patch-based texture synthesis algorithms, the parameters, including the patch size and the boundary region width, have much impact on synthesis efficiency. If these two parameters are unreasonable, our proposed method can also be influenced more or less especially when handling textures with salient features. Obviously, a patch with a reasonable size can well capture structural information of the sample textures. If the width

of boundary region is suitable, every possible boundary zone can find only one matched patch from the sample texture, which can avoid mismatched features across patch boundaries. In all experiments, we empirically defined that the sizes of patches should be big enough but no more than a half of the minimal size of the sample texture. And the width of the boundary regions is $10-15\%$ of the predefined patch size. For instance, it is typically six pixel wide for stochastic textures and ten pixel wide for textures with high structure when the patch size is 64×64.



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

Fig. 5: The effect of different relative important weight λ. (a) is the sample texture. (b) $\lambda=0.6$. (c) $\lambda=0.2$. (d) $\lambda=0.13$.

During the synthesis procedure, since similar texture features at the boundary region of adjacent patches do not means strictly identical and each best texture patch usually present undesirable intensity discrepancies because of the global variations of illumination, structure discontinuity and color inconsistency are inevitable when using previous techniques[7]-[9] for patches merging, such as feathering, dynamic programming, graph cuts. In order to eliminate such effects, we use image stitching algorithm to merge adjacent patches proposed by Jiaya Jia et al.[21][22], which used structure deformation and propagation to handle structure and intensity misalignment. By using this stitching method, the texture features and the colors transition between texture patches will become smooth. Note that we perform the stitching operation after all texture patches are found.

4. Experimental results and discussions

All of our experiments are performed on AMD Phenom(tm) 8400 2.10 GHz Triple-Core Processor with 2GB of main memory by using unoptimized Matlab code. Our technique works well for a wide range of textures varying from stochastic to regular. The average execution time for a sample texture of size 128×128 and synthesis of 300×300 texture is 5-7 minutes, which spend much time on texture patches sampling. For acceleration, we reduce the dimension of the search space using PCA when comparing the differences in the orientation fields and our algorithm is extended to perform synthesis with a multiresolution scheme. The execution time is reduced to about 1-2 minutes.

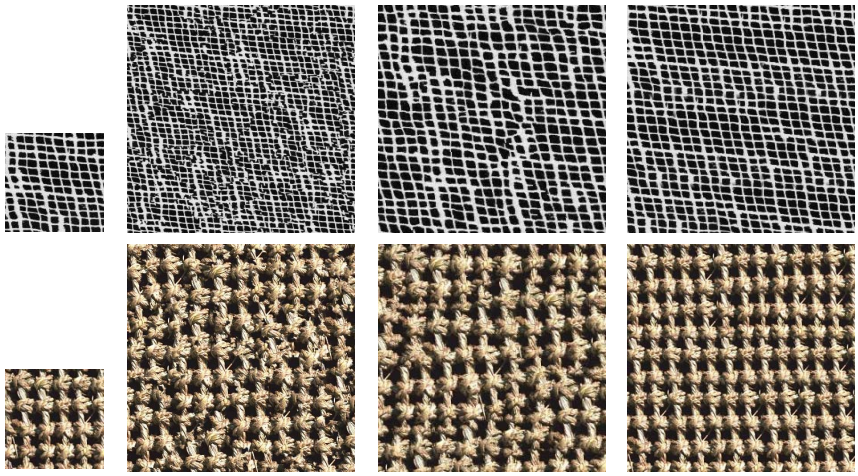
From Fig. 6 and Fig. 7, the given results were generated by some classical pixel-based texture synthesis algorithms. The WL algorithm performs relatively poorly on textures consisting of an arrangement of distinct objects of irregular shapes and sizes, such as talus field and blossoming flower (the given samples in Fig. 6). The size of the L-shaped neighborhood defined in this method critically affects the performance of the algorithm, which often results in severe garbage regions in the synthesized textures. The Ashikhmin's algorithm applied spatial coherence to speed the synthesis procedure. However, local neighborhood searching is easy to produce irregularity and blurring. The Jump Map based texture synthesis algorithm and the parallel controllable texture synthesis algorithm (as shown in Fig. 7) have achieved great improvement in quality and speed, which have ability to capture structure information. There is, however, no guarantee that texture features or structures are matched. Obviously, the results synthesized by our method show better

quality, which maintain a visual resemblance with the original texture samples. Our algorithm in general belongs to the patch-based sample texture synthesis approaches, and the quality and speed can be improved by inheriting the advantages of this category of algorithms, such as their extensibility and high efficiency.



(a) Sample (b) WL algorithm (c) Ashikhmin's based (d) Our algorithm

Fig. 6: Comparison with the Wei and Levoy's algorithm[2] and the Ashikhmin's algorithm[3]. Their synthesized results show in (b) and (c) respectively.

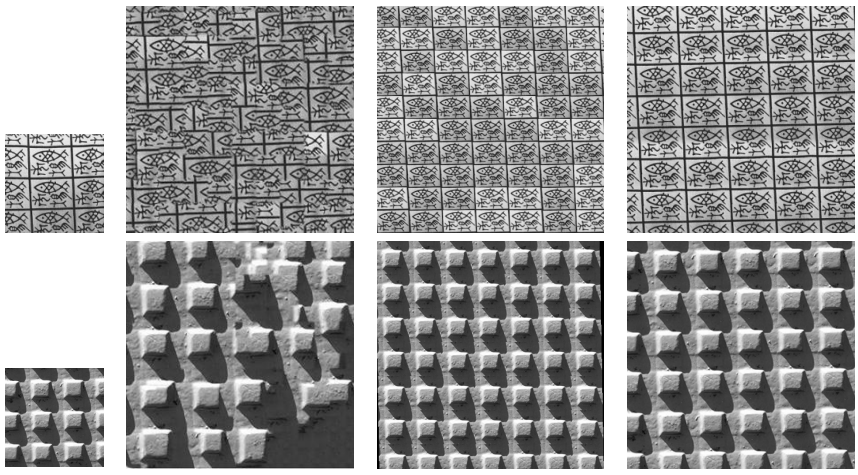


(a) Sample (b) Jump Map based (c) Parallel Controllable (d) Our algorithm

Fig. 7: Comparison with the Jump Map based texture synthesis algorithm[4] and the parallel controllable texture synthesis algorithm[6]. Their synthesized results show in (b) and (c) respectively.

As the results in Fig. 8 shows, synthesizing the given sample textures by using our patch-based method leads to very good results. Traditional patch-based sampling algorithm[7] fails to generate desired results, in that the confusions are easily caused by indistinguishable color information in sample textures. The results shown in the second column reveal visual artifacts, such as severe structural mismatching for the first example and garbage growing for other example. The near-regular texture synthesis algorithm[11][12] is useful to produce results in high quality, which are shown in the third column. However, broken features show in the last example (the shadow of stones) when texture tiles do not exactly pasted together, in that tiles do not maintain the global texture features. In our method, we attempt to explore the correlation of pixels and orientation field when comparing the differences between neighboring patches, which achieves accurate similarity

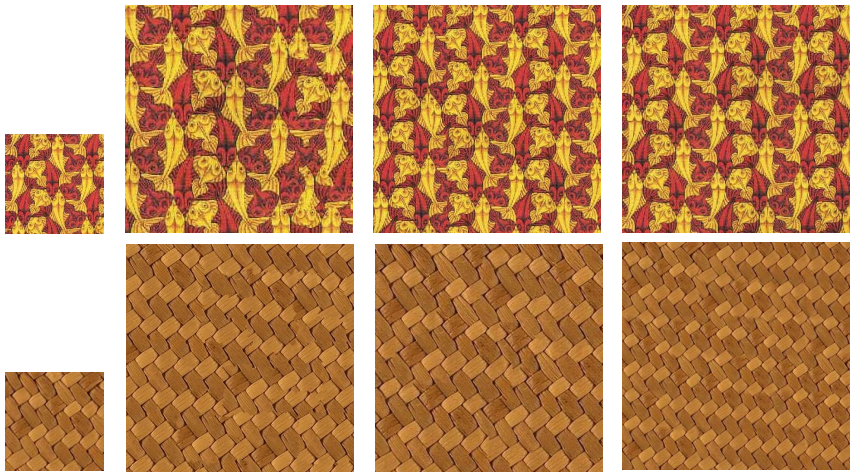
measure. In addition, our method also has ability to preserve the underlying structural regularity.



(a) Sample (b) Patch based (c) Near-regular (d) Our algorithm

Fig. 8: Comparison with patch-based sampling algorithm[7] and near-regular texture synthesis[11][12]. Their synthesized results show in (b) and (c) respectively.

Figure 9 shows a comparison of our method with some state of the art texture synthesis algorithms, including image quilting algorithm[8] and graph-cuts algorithm[9]. Our objective is to compare the synthesized quality when using different techniques to merge texture patches. A common significant problem in these two algorithms is the presence of broken features at the boundary region of adjacent patches. Since the RGB-based similarity measure for finding the best matches cannot satisfy the synthesis requirement, the structure and color misalignment may remain when the graph-cuts and dynamic programming were adopted to merge patches. In our cases, the results for these two examples are better than others when using structure deformation and propagation on the boundary region, which are helpful for performing a smooth transition in texture features.

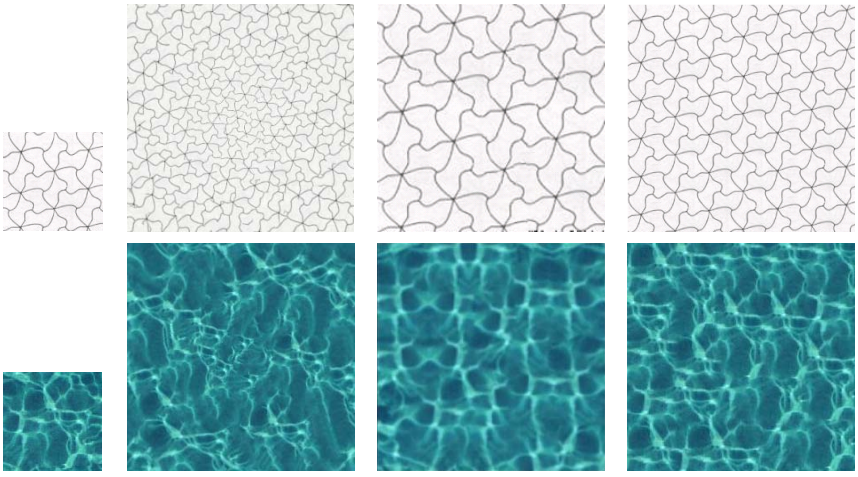


(a) Sample (b) Image Quilting (c) Graph-Cuts (d) Our algorithm

Fig. 9: Comparison with image quilting algorithm[8] and the graph cuts algorithm[9]. Their synthesized results show in (b) and (c) respectively.

Figure 10 illustrates a more comparative study. The left column shows the input, the middle columns from left to right show the synthesized results generated from appearance-space based synthesis technique and feature matching based texture synthesis

approach, the last column shows our results. These two algorithms explore texture features to solve the problem of structure discontinuity. However, the signed feature distance field used in[15] fails to produce a smooth transition because of its ambiguity. In addition, although Wu and Yu[14] introduce local warping to align texture features, the matched features show artificialities in the given results. As demonstrated by this figure, our method provides better results than that of above approaches. The main reason is that layer correlation based color differences succeed to recognize salient structure, and structure continuity is also captured within the orientation field based distance metric. Figure 11 illustrates some more synthesis results. For each example, the left is the sample texture and the right is synthesized result. These sample textures from top row to bottom are classified as textures with salient features. These examples indicate that our method is capable of capturing structural information to generate satisfactory results.



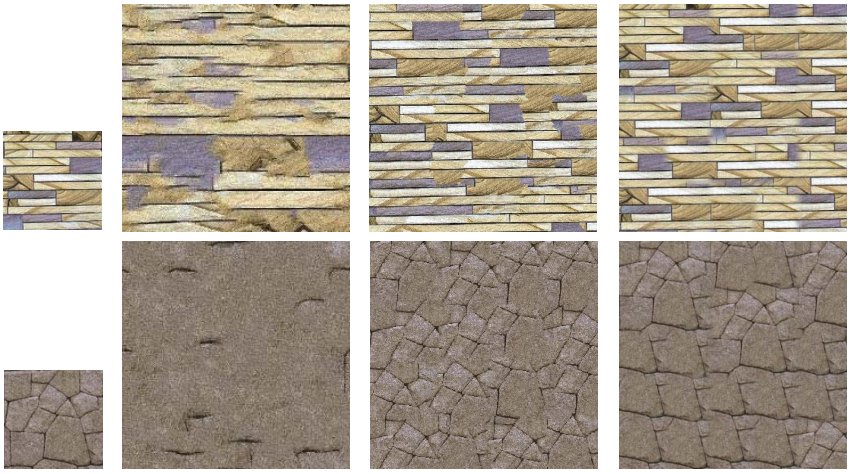
(a) Sample (b) Appearance-space (c) Feature Matching (d) Our algorithm

Fig. 10: Comparison with appearance-space texture synthesis[15] and the feature matching synthesis technique[14]. Their synthesized results show in (b) and (c) respectively.



Fig. 11: More results for texture synthesis. For each pair of images, the left is a sample texture and the right is a synthesized result by using our algorithm.

As shown in Fig. 12, one limitation shown in the first sample is that the synthesized texture becomes less random because we only chosen the best patches with the minimal error. Although it fails to preserve the original structural layout of the sample texture, the texture features or structures are matched. The other limitation shown in the second sample is that our method does not totally eliminate structure discontinuity even though the deformation propagation was used to align the mismatched texture features. When there are only a limited number of feature points detected on the overlapping region due to limited patch boundary, such deformation operation may invalidate to achieve a correct structure matching. In addition, the synthesized result have shown serious verbatim copying , in that there are no methods to estimate a reasonable patch size and boundary region width, which is still an open problem by applying patch-based synthesis framework.



(a) Sample (b) WL algorithm (c) Image Quilting (d) Our algorithm

Figure 12: The failed example, comparison with the Wei and Levoy's algorithm[2] and Image quilting algorithm[8] from left to right. Our results show in the last column.

5. Conclusion

In this paper, we propose an effective feature description based synthesis method for texture generation, which mainly promotes previous texture synthesis techniques using the RGB-based distance metric. The use of layer correlation and orientation field can well capture structural information, enhancing the similarity measure. Moreover, the deformation propagation is used to merge adjacent patches, which succeed to find a smooth transition across patch boundaries. Compared with the state of the art texture synthesis approaches, experiment results have demonstrated that our method not only reduces structure discontinuity, but also works well for textures with salient features, to produce high quality results.

Acknowledgments

This work was supported by the National Basic Research Project of China (No. 2011CB302203), the National Natural Science Foundation of China (No.60873136) and the Innovation Program of the Science and Technology Commission of Shanghai Municipality (No. 10511501200).

References

(1)Efros A. and Leung T., Texture synthesis by non-parametric sampling, *International Conference on Computer Vision*, (1999), pp. 1033-1038.
(2)Wei L. Y. and Levoy M., Fast texture synthesis using tree-structured vector quantization,

In: SIGGRAPH '00: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, (2000), pp. 479-488.
(3)Ashikhmin M., Synthesizing natural textures, *In: ACM Symposium on Interactive 3D Graphics*, (2001), pp. 217-226.
(4)Zelinka S. and Garland M., Towards Real-time Texture Synthesis with the Jump Map, *EGRW '02 Proceedings of the 13th Eurographics workshop on Rendering*, (2002), pp. 99-104.
(5)Harrison P., A non-hierarchical procedure for resynthesis of complex textures, *In WSCG '2001 Conference proceedings*, (2001), pp. 190-197.
(6)Lefebvre S. and Hoppe H., Parallel controllable texture synthesis, *ACM Transaction on Graphics*, Vol. 24, No.3(2005), pp. 777-786.
(7)Liang L., Liu C., Xu Y., Guo B., and Shum H. Y., Real-time texture synthesis by patch-based sampling, *ACM Transaction on Graphics*, Vol. 20, No.3(2001), pp. 127-150.
(8)Efros A., and Freeman W., Image quilting for texture synthesis and transfer, *In: SIGGRAPH'01*, (2001).
(9)Kwatra V., Sch¨odl A., Essa I., Turk G., and Bobick A., Graphcut textures: image and video synthesis using graph cuts, *ACM Transactions on Graphics*, Vol. 22, No.3(2003), pp. 277-286.
(10)Cohen M. F., Shade J., Hiller S., and Deussen O., Wang tiles for image and texture generation, *In SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, (2003), pp. 287-294.
(11)Liu Y. and Tsin Y., The promise and the perils of near-regular texture, *International Journal of Computer Vision*, Vol.62, (2002), pp.1-2.
(12)Liu Y. X., Lin W. C., and Hays J., Near-regular texture analysis and manipulation, *In SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, (2004), pp. 368-376.
(13)Zhang J., Zhou K., Velho L., Guo B., and Shum H. Y., Synthesis of progressively-variant textures on arbitrary surfaces, *In: SIGGRAPH' 03: ACMSIGGRAPH 2003 Papers*, (2003), pp. 295-302.
(14)Wu Q. and Yu Y., Feature matching and deformation for texture synthesis, *ACM Transaction on Graphics*, Vol. 23, No.3(2004), pp. 364-367.
(15)Lefebvre S. and Hoppe H., Appearance-space texture synthesis, *In: SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, (2006), pp. 541-548.
(16)Dischler J. M. and Zara F., Real-time structured texture synthesis and editing using image-mesh analogies, *The Visual Computer*, Vol. 22, No.9(2006), pp.926-935.
(17)Min Chen, Canlin Mo and Shaoyi Guo, Textue synthesis with self-correlation distinguishing and its application, *Journal of Computer Research and Development*, Vol. 45, No. 2(2007), pp. 337-341.
(18)Rao A. R. and Schunck B. G., Computing oriented texture fields, *CVGIP: Graphical Models Image Process.*, Vol. 53, No.2(1991), pp. 157-185.
(19)Rao A. R. and Jain R. C., computed flow field analysis: oriented texture fields, *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol.14, No.7(1992), pp. 693-709.
(20)Dizenzo S., A note on the gradient of a multiimage, *Computer Vision, Graphics, and Image Processing*, Vol. 3, No. 1(1986), pp. 116-125.
(21)Jiaya Jia and Chi-Keung Tang, Eliminating Structure and Intensity Misalignment in Image Stitching, *Tenth IEEE International Conference on Computer Vision (ICCV'05)*, (2005-2), pp.1651-1658.
(22)Jiaya Jia and Chi-keung Tang. Image Stitching Using Structure Deformation, *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol.30, No. 4(2008), pp. 617-631.