

# Feature-aware natural texture synthesis

Fuzhang Wu · Weiming Dong · Yan Kong ·  
Xing Mei · Dong-Ming Yan · Xiaopeng Zhang ·  
Jean-Claude Paul

Published online: 4 December 2014  
© Springer-Verlag Berlin Heidelberg 2014

**Abstract** This article presents a framework for natural texture synthesis and processing. This framework is motivated by the observation that given examples captured in natural scene, texture synthesis addresses a critical problem, namely, that synthesis quality can be affected adversely if the texture elements in an example display spatially varied patterns, such as perspective distortion, the composition of different sub-textures, and variations in global color pattern as a result of complex illumination. This issue is common in natural textures and is a fundamental challenge for previously developed methods. Thus, we address it from a feature point of view and propose a feature-aware approach to synthesize natural textures. The synthesis process is guided by a feature map that represents the visual characteristics of the input texture. Moreover, we present a novel adaptive initialization algorithm that can effectively avoid the repeat and verbatim copying artifacts. Our approach improves texture synthesis

in many images that cannot be handled effectively with traditional technologies.

**Keywords** Texture synthesis · Texture feature analysis · Composite texture · Feature-aware synthesis

## 1 Introduction

Image textures can either be artificially created or observed in the natural scenes captured in an image. Natural textures usually include the textures created without mechanical influence or embossed rolls. Realistic textures must be synthesized to decorate objects in virtual scenes in numerous applications.

Texture synthesis is defined in [1] as “a method that starts from a sample image and attempts to produce a texture with a visual appearance similar to that sample”. Most previous example-based techniques [24] focus on handling and generating isotropic textures; however, the real world exhibits many anisotropic examples of natural texture, which contain TEXTure Elements (texels) of varying shapes, color patterns, and materials. These textures are useful in many image editing applications, such as image compositing and texture replacement in natural scenes. One such example is shown in Fig. 1a. The sizes of the flowers and leaves are varied spatially as a result of perspective distortion, and the scene contains complex patterns of distributed flowers. Previous techniques synthesize textures without requiring the preservation of global visual effect. Consequently, clear artifacts are generated, such as content loss and damaged perspectives (e.g., Fig. 1d, e). Therefore, we must recover the information regarding the perspective and texel size of the input texture and use it to guide the synthesis process and generate a good synthesis result. However, this scheme may strongly

---

**Electronic supplementary material** The online version of this article (doi:10.1007/s00371-014-1054-y) contains supplementary material, which is available to authorized users.

---

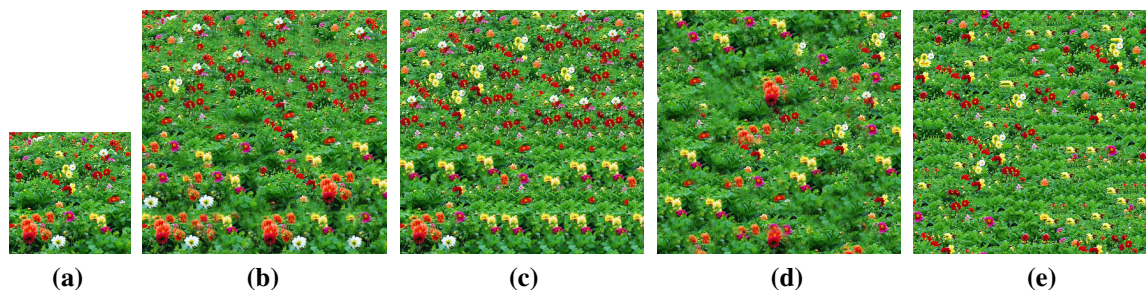
F. Wu · W. Dong (✉) · Y. Kong · X. Mei · D.-M. Yan · X. Zhang  
LIAMA-NLPR, Institute of Automation, Chinese Academy  
of Sciences, Beijing, China  
e-mail: Weiming.Dong@ia.ac.cn

X. Mei  
e-mail: xmei@nlpr.ia.ac.cn

X. Zhang  
e-mail: xpzhang@nlpr.ia.ac.cn

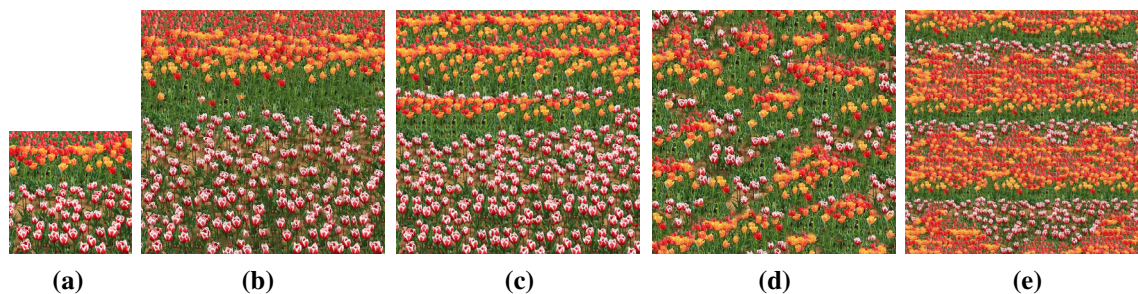
D.-M. Yan  
Visual Computing Center, King Abdullah University of Science  
and Technology, Thuwal, Kingdom of Saudi Arabia  
e-mail: yandongming@gmail.com

J.-C. Paul  
Project CAD, INRIA, Paris, France



**Fig. 1** Synthesis of natural texture, which is not captured by orthographic projection. The results of previous methods display repeat and verbatim copying artifacts [e.g., (c)], or eliminate the perspective visual effect [e.g., (d) and (e)]. Our result reproduces the global visual appear-

ance of the input and is more random visually. **a** Input. **b** Our result. **c** Perspective-aware [3]. **d** Texture optimization [11]. **e** Parallel synthesis [14]



**Fig. 2** Synthesis of natural texture, which is composed of three types of sub-textures. Our method preserves the spatial composition of sub-textures and the perspective characteristic. The image in c depicts only

the perspective characteristic, whereas c, d, and e lose the composition characteristics of the sub-textures. **a** Input. **b** Our result. **c** Perspective aware [3]. **d** Texture optimization [11]. **e** Parallel synthesis [14]

restrict the search space during synthesis and induce obvious artifacts, such as repeat copying a single patch and verbatim copying parts of the sample image (e.g., Fig. 1c). Moreover, the example may involve multiple sub-textures of different materials (referred as composite texture [28], e.g., Fig. 2a). The sub-textures should be synthesized separately to maintain the layout structure. However, the boundary between two sub-textures may be indistinct in some cases; thus, the result is expected to display a transition area.

These examples are not unique and exhibit a common problem, that is, synthesis can fail in general as a result of content complexity when texture images are not captured from orthographic projection (causing perspective distortion), contain small, salient distributed objects (generating complex patterns), involve multiple sub-textures, or display spatially varied color patterns. This problem cannot be avoided given the anisotropic visual characteristics of many natural textures. Therefore, the vivid global visual effects of the sample input are easily lost in the results.

To solve this notorious and universal problem, we propose a novel, feature-aware framework of texture synthesis. This framework analyzes the visual characteristics of an anisotropic natural texture from multiple aspects and then integrates these characteristics to guide the synthesis process in reproducing the global visual characteristics of the sample and generating the final result. This result is visually similar

to the input texture. In addition, we present a new initialization algorithm based on poisson disc sampling [23, 27] to replace the traditional random initialization scheme and to avoid producing the repeat and verbatim copying artifacts that limit the randomness in the output texture.

Our algorithm improves the texture optimization framework [11] by preserving the global visual characteristics of the sample input during synthesis. Our automatic framework is easy to use and can be extended to user-assisted texture design. Our major contributions are as follows:

- An automatic and effective scheme to extract the size variation of texels as a result of perspective distortion, which is common in most natural scenes.
- A novel initialization scheme to solve problems with repeat and verbatim copying. This scheme can maintain the random distribution of texels in the result and effectively enhances the visual quality of the synthesis result.

## 2 Related work

*Texture synthesis* Texture synthesis is a general example-based methodology to synthesize similar phenomena, as explored in [24]. Most common texture synthesis approaches are based on Markov random fields (MRF). However, the

current scheme models a texture as the realization of a local and stationary random process by assuming that pixels with similar neighborhoods should be similarly colored. Per-pixel algorithms develop texture pixel by pixel [1, 2, 6, 25], whereas patch-based algorithms [5, 11, 12, 16, 26] cut and paste selected source regions together to form a new texture. Efficient GPU-based techniques of texture synthesis [13, 14] have also been proposed, which achieve spatial determinism and good controllability. Ma et al. [18] present a data-driven method to synthesize repetitive elements. Kim et al. [10] describe the spatial patterns of real-world textures in symmetry space and have utilized these representations of partial and approximate symmetries to guide texture synthesis and manipulation.

*Anisotropic and inhomogeneous synthesis* The basic MRF-based scheme in most existing methods of texture synthesis cannot adequately address the global visual variation of texels with semantic meanings. Zalesny et al. [28] propose a hierarchical approach to texture synthesis that considers textures to be composites of simple sub-textures. Han et al. [8] use a few low-resolution exemplars as inputs to produce inhomogeneous textures that span large or infinite ranges of scale. However, the two methods above cannot handle the problem of perspective distortion in the sample input. Eisenacher et al. [7] apply distorted textures as inputs and synthesized textures under new distortions. This method is limited in that user interactions are required to define a Jacobian field for use as an input in the synthesis involving distorted space algorithms. Nonetheless, it is effective for structured textures (e.g., bricks and stones), although it cannot address the stochastic textures that are popular in natural scenes (e.g., grass, flowers, and water). Dong et al. [3] establish a perspective-aware texture synthesis algorithm by analyzing the size variation of a texel. This framework can preserve the perspective characteristic of the sample input in the outputted result, but most examples must generate the control maps manually (to indicate the size of the texels). Thus, the results often display repeat and verbatim copying artifacts. Dong et al. [4] present an interactive method to synthesize plant scenes from natural texture samples. Control maps are manually generated to preserve the global appearance of the examples in the results. Image quilting [5] is adapted as the texture synthesis framework. However, this method also suffers from repeat copying and verbatim copying artifacts, as in [3]. Rosenberger et al. [20] propose an example-based method to synthesize control maps suitable for non-stationary textures, such as rusted metal and marble. This method is unfit for natural textures that cannot extract feasible layer maps, such as grass and sand. Ma et al. [17] report on dynamic element textures that aim to produce controllable repetitions by combining constrained optimization and data-driven computation. Park et al. [19] develop a framework for the semi-automatic synthesis of inhomogeneous textures from a theoretically infinite

number of input exemplars. However, this method requires the input examples to be somewhat homogeneous. Moreover, it cannot handle perspective distortions.

### 3 Overview

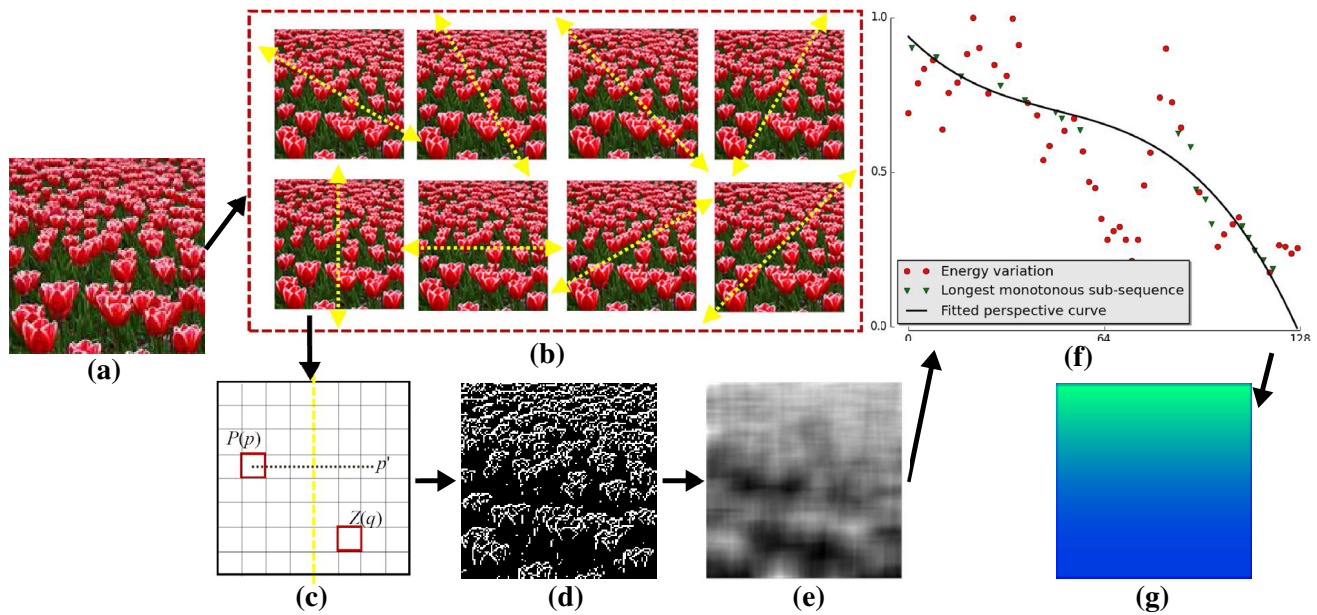
In this study, we investigate feature-aware texture synthesis and manipulation. We mainly specify the visual characteristics of natural textures in feature constraints and use an objective function that has been optimized according to these constraints to guide texture synthesis. To construct the framework, we first analyze the content of the input texture to extract its visual characteristics, including perspective direction (Sect. 4.1), texel sizes (Sect. 4.2), and texel types (a composite texture contains several types of sub-textures, Sect. 4.3). We integrate the characteristics of texel sizes and texel types into a size and a type map separately. Using these characteristics as feature constraints, we develop a new approach to optimize the synthesis process for natural textures (Sect. 5). We also use the size map to control the locations of differently sized texels in the output texture to maintain the perspective effect. We utilize the type map of a composite texture to control which sub-texture appears in which location. Furthermore, we propose a novel initialization algorithm based on poisson disc sampling to enhance visual randomness in the synthesized texture (Sect. 5.1). This operation is crucial in solving the problems of repeat and verbatim copying caused by the limitation in searching space when synthesizing an example with perspective effect. In addition, we propose the new neighborhood metric to preserve the global visual appearance of the input texture during synthesis (Sect. 5.2). Finally, we adapt the expectation–maximization (EM)-like optimization framework to synthesize the output texture (Sect. 5.3).

## 4 Analysis of texture content

### 4.1 Detection of perspective direction

The variation of texel sizes in natural textures is strongly related to perspective direction. Given a sample input texture  $\mathcal{I}$ , we denote the texel size at position  $p$  as  $S(p)$  and assume that  $S(p)$  varies monotonously from large to small in a fixed direction. This assumption is reasonable for many natural scenes according to the common photographic custom. If we segment the image into two parts using a line of the perspective direction in accordance with this assumption, then we can expect their contents to be similar. Thus, we simplify the calculation of perspective direction to determine the best “reflection symmetry axis.” In this study, we refer to this axis





**Fig. 3** Estimation of perspective direction and texel sizes. We apply *blue color* in **g** to illustrate large texels. **a** Input. **b** Perspective directions/lines for testing. **c** Patch distance. **d** Local energy. **e** Rough size map. **f** Fitting texel size curve. **g** Final size map

as the perspective line of a natural texture. In our approach, we check the 8 axes that include 16 perspective directions (i.e.,  $i \cdot 22.5^\circ$ ,  $i = 0, 1, \dots, 15$ ), as shown in Fig. 3b. Each axis represents two opposite directions that form the same line and can be treated by the same operation during texel size estimation (detailed in Sect. 4.2). This scheme usually estimates the texel sizes adequately because we typically do not require a highly accurate size variation map. The varying direction is the most important variable.

We get the perspective line by comparing the content similarities of the two “symmetric regions”. Without loss of generality, we use the test of vertical bisector as an illustration, as depicted in Fig. 3c. First, we divide the two image regions into multiple equally sized patches of equal size ( $32 \times 32$  in all our experiments). We denote  $\mathbb{P}_1$  and  $\mathbb{P}_2$  as the patch sets of these two regions. The metric is defined as:

$$d(\mathbb{P}_1, \mathbb{P}_2) = \min_{\phi} \sum_{i=1}^N d_e(\mathbb{P}_1(\phi(i)), \mathbb{P}_2(i)), \quad (1)$$

where  $\phi$  belongs to the set of permutations of  $\{1, \dots, N\}$  and  $N$  is the number of patches in each region. We adopt Earth Mover’s Distance (EMD) [21] to optimize the mapping cost in Eq. (1). Each patch is assigned the same weight in our implementation. The computation of EMD is then reduced to an assignment problem. Therefore, each patch in the left region is assigned to exactly one patch in the right region. We apply an exponential distance to compute the distance between two patches  $P(p)$  (in the left region) and  $P(q)$  (in the right region):

$$d_e = \exp\left(\frac{\|p' - q\|^2}{\delta_s}\right) \cdot \exp\left(\frac{D(P(p), P(q))}{\delta_c}\right), \quad (2)$$

where  $p$  is the center coordinate of  $P(p)$ ;  $q$  is the center coordinate of  $P(q)$ ;  $p'$  is the symmetric coordinate of  $p$ ,  $\|\cdot\|$  denotes the  $l_2$  norm;  $D$  is the sum of squared distances (SSD) of the pixel colors between  $P(p)$  and  $P(q)$ ; and  $\delta_s$  and  $\delta_c$  are the parameters used to control the contributions of geometric location and color information, respectively ( $\delta_s = 0.6$  and  $\delta_c = 1.0$  in our experiments). Thus, we calculate the “symmetry distance” of each axis. The axis with the minimum distance is the perspective line along which the texel sizes are varied. For example, the vertical axis that has the minimum symmetry distance is the perspective line in Fig. 3b.

#### 4.2 Estimation of texel size

We estimate the variation characteristics of the texel sizes along this line after obtaining the perspective direction (the perspective line). For each pixel in  $\mathcal{I}$  we calculate its gradient in both X and Y directions (normalized to  $[0, 1]$ ) and select the maximum value  $E(p) = \max(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y})$  as the local energy, as displayed in the form in Fig. 3(d). We calculate the number of pixels (denoted as  $N_E(p)$ ) whose energy value is larger than a threshold at each location  $p$  in a  $20 \times 20$  window.  $N_E(p)$  is used to estimate the complexity of local content. We consider the  $N_E(p)$  value at this position to also be large if a local region contains a large number of texels because many texels usually increase the frequency of energy variations. Therefore, texel size  $S(p)$  is inversely proportional



**Fig. 4** Examples of composite natural textures

to  $N_E(p)$ .  $N_E(p)$  is visualized as a noisy texel size estimation, as exhibited in Fig. 3e. The pixels in dark areas have large texels. However, such size maps are unfit to guide the texture synthesis process because of the following two reasons: first, some areas experience errors caused by the unpredictably complex content or by the low quality/resolution of the example; second, the discrete distribution of size values may enhance unexpected seams and texel discontinuity in the result. In fact, the use of a continuous and smooth size map such as Fig. 3g in our experiments generates high-quality results.

To produce a continuously varied size map, we first calculate the sum of  $N_E(p)$  (denoted as  $S_E(x)$ ) along every line, which is perpendicular to the perspective line. For the example in Fig. 3a, we simply calculate  $S_E(x)$  along each row. By setting the line position as the X coordinate and the normalized  $S_E(x)$  value as the Y coordinate, we obtain the red circles in Fig. 3f. We then determine the longest monotonous sub-sequence (the green triangles in Fig. 3f) and apply these values to this sub-curve to fit a cubic polynomial curve (the black curve in Fig. 3f). Finally, we use this polynomial to re-generate the size value of each line and to generate the continuous varied size map in Fig. 3g.

#### 4.3 Sub-textures of composite texture

We display standard composite natural textures in Fig. 4. Each composite texture contains several sub-textures of different materials. To synthesize a composite texture, we first segment the sample input into separate sub-textures. Many texture segmentation techniques [9, 15] can be used in this process. The segmentation may need to be adjusted manually if it is inaccurate. The segmentation results of the four

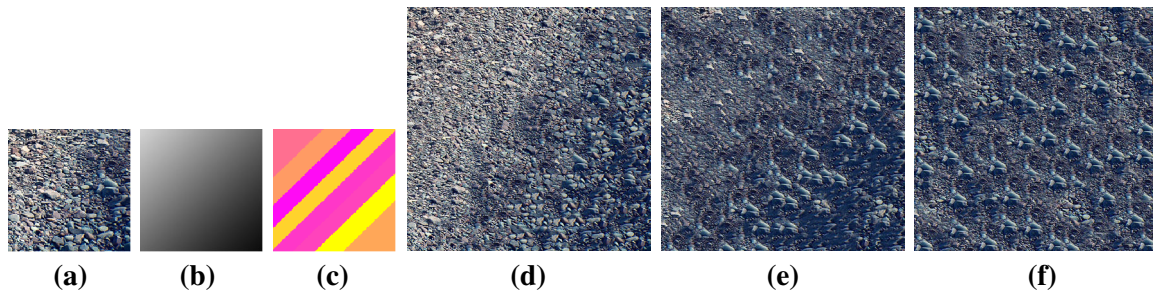
textures in the first row of Fig. 4 are provided in the second row. Different intensities represent various types of sub-textures.

## 5 Synthesis approach

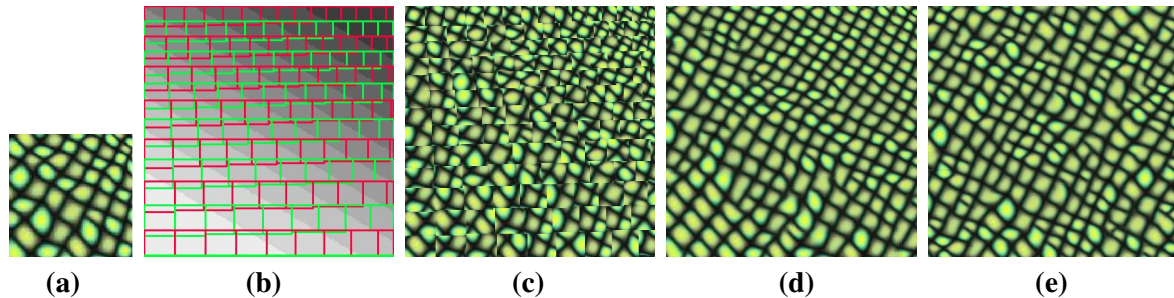
### 5.1 Initialization

We employ patch-based approach in our framework for texture synthesis. However, the normal initialization scheme that typically copies patches from the sample into the output domain randomly is unsuitable for our synthesis requirement. The synthesized results obtained with purely random initialization easily generates repeat copying artifacts and loses the perspective effects in the output images, as indicated in Fig. 5e. To address these problems, we present a new initialization method for natural texture synthesis that considers the visual characteristics of the sample input.

Artifacts may be produced if we use fixed-size patches to initialize the output of a texture under perspective effect, as in previous methods. For instance, areas with small elements are subject to repeat copying if we use large patches, whereas areas with large elements may contain incomplete elements if we use small patches. Therefore, we develop a novel scheme to adaptively determine the size of an initializing patch at each location. We first cluster the pixels in the size map according to the texel sizes through the  $k$ -means method ( $k = 10$  in all our experiments) and calculate the average size value of each clustered set (e.g., Fig. 5c). We then re-sample the clustered input size map  $\mathcal{C}_I$  in relation to the size of the output to generate output control map  $\mathcal{C}_O$  and to denote the texel size component of  $\mathcal{C}_O$  as  $\mathcal{C}_O^S$ . We begin



**Fig. 5** Comparison of texture synthesis results using our initialization and random initialization methods. **a** Input. **b** Control map. **c** Clustered. **d** Our result. **e** Ours with random init. **f** Texture Optimization [11]



**Fig. 6** Initialization of the output image with patches of adaptively variant sizes. **a** Input. **b** Patch grid. **c** Initialized output image. **d** Our synthesis result. **e** Texture optimization [11]

the initialization by placing patches side by side on the output domain in scan-line order. We calculate the patch size of the first patch in the left-top corner of the output using the texel size  $\mathcal{C}_O^S(0, 0)$ . We then calculate the size of another patch in the row according to the  $\mathcal{C}^S$  value of the right-top pixel of the patch on its left. We determine the minimum patch size  $w_{\min}$  of the first row of patches after initialization and apply the next row of patches from  $(0, w_{\min})$ . The size of the first patch in this row is computed according to  $\mathcal{C}_O^S(0, w_{\min})$ . We then position the other patches according to the scheme used on the first row. We continuously place rows of patches with adaptively variant sizes based on the scheme of the second row until the output domain is filled. Fig. 6b shows a sample patch grid wherein patch sizes are adaptively variant.

Two items must be specified for the initialization process described above. The first item is how the patch size is adaptively calculated. Without loss of generality, we take a patch that is located in the middle of a row as an example and denote the size value of its left-top corner as  $\mathcal{C}_O^S(p)$ . We set the patch size  $w$  as:

$$w = 16 + 32 \cdot \mathcal{C}_O^S(p), \quad (3)$$

which is varied in the interval  $[16, 48]$ . The other item involves the selection of a feasible patch  $P(q)$  from the sample input for a patch  $P(p)$ , which is initialized in the output.

Both  $p$  and  $q$  represent the central coordinates of the patch. Given an  $w(p) \times w(p)$  output patch  $P(p)$ , we first locate the cluster to which  $P(p)$  belongs in the clustered input control map. We then randomly choose a patch  $P(q)$  from this cluster. As presented in Fig. 7, we check the following two conditions for each existing patch  $P(p)$  that intersects with a  $4w(p) \times 4w(p)$  neighborhood around  $P(p)$ :

$$\begin{aligned} |p - p'| < w(p) \& |q - q'| \geq |p - p'|, \\ |p - p'| \geq w(p) \& |q - q'| \geq 2w(p), \end{aligned} \quad (4)$$

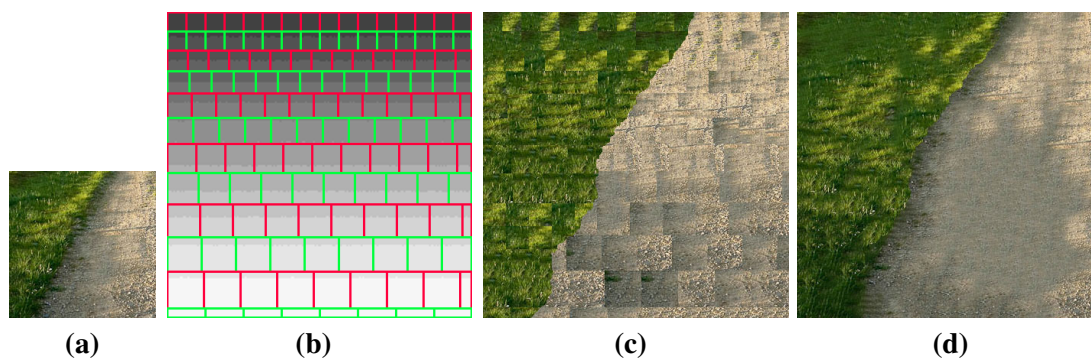
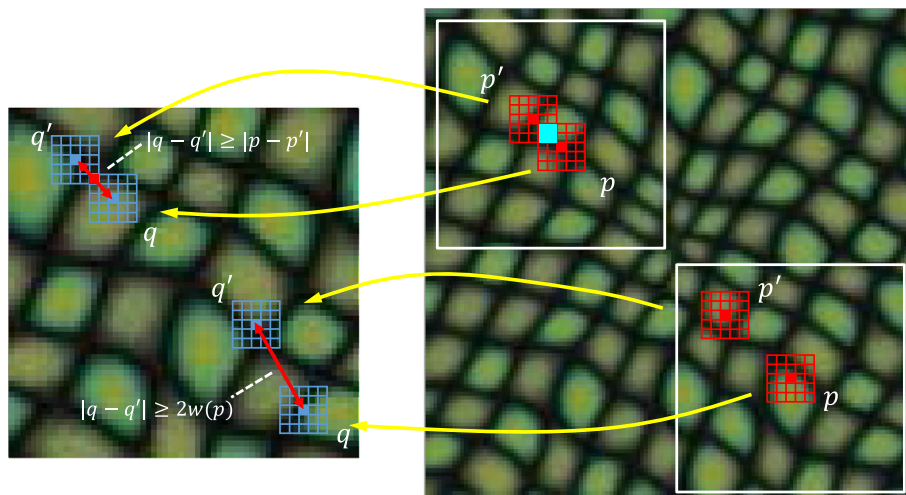
where  $q'$  is the central coordinate of  $P(q')$  in the sample input, which has been copied to  $P(p')$  as an initialization. We copy  $P(q)$  to  $P(p)$  if either of the above conditions is true. Otherwise, we randomly select a new  $P(q)$  if the patch is consistent with either of the two conditions. An initialized output image is depicted in Fig. 6c.

The two heuristic conditions in Eq. (4) can ensure that the same content does not appear very closely and frequently in the output, which can effectively reduce the presence of repeat and verbatim copying artifacts in the result.

We re-sample the input type map to output size in the process of synthesizing a composite texture to obtain the spatial layout of sub-textures in the resultant image. Therefore, we still adaptively calculate the patch sizes according to the size map during initialization, but we select patches from different sub-textures based on the type map. We check the



**Fig. 7** Illustration of the two conditions in Eq. (4)



**Fig. 8** Initialization and synthesis of a composite texture. **a** Input. **b** Patch Grid. **c** Initialization. **d** Result

type value at its central coordinate and initialize the patch by choosing patches from the corresponding sub-texture given a patch that crosses the border of two different clusters. Figure 8 shows the initialization results of a composite texture. Our algorithm maintains the spatial distribution of sub-textures in the initialization results. Moreover, we synthesize each sub-texture separately. This scheme can ensure the generation of a texture that is visually similar to the input example.

### 5.2 Neighborhood metric

The neighborhood similarity metric is the core component of example-based texture synthesis algorithms. We denote  $Z_p$  as the spatial neighborhood around a sample  $p$ , which is constructed by uniting all of the pixels within its spatial extent as defined by a user-specified neighborhood size. We formulate the distance metric between the neighborhoods of two samples  $p$  and  $q$  as:

$$M(Z_q; Z_p) = \sum_{p' \in Z_p} \|\mathbf{c}_{q'} - \mathbf{c}_{p'}\|^2 + \alpha \cdot \|\mathbf{x}_q - \mathbf{x}_p\|^2, \quad (5)$$

where  $p'$  runs through all pixels  $\in Z_p$ ,  $q' \in Z_q$  is the spatially corresponding sample of  $p'$ ;  $\mathbf{c}$  represents the pixel color in the RGB space; and  $\mathbf{x}$  is the feature value of a sample pixel in the feature map. We incorporate the visual characteristic information of the texture content into the neighborhood metric, unlike in traditional texture synthesis wherein the neighborhood metric is usually defined as a simple SSD of the pixel attributes (such as colors and edges). This item can preserve the global appearance without over-smoothing and generating clear partial/broken objects (discussed in detail in Sect. 6).

### 5.3 Optimization

Therefore, we aim to synthesize an output  $\mathcal{O}$  that is visually similar to the original sample texture  $\mathcal{I}$  in appearance. We formulate this problem as an optimization problem [11] using the following energy function:

$$E(\mathcal{I}; \mathcal{O}) = \sum_{p \in \mathcal{O}} M(Z_q; Z_p), \quad (6)$$

where the first term measures the similarity between sample input  $\mathcal{I}$  and  $\mathcal{O}$  using our local neighborhood metric as defined in Eq. (5). Specifically, we determine the corresponding input sample  $q \in \mathcal{I}$  according to the most similar neighborhood [according to Eq. (5)] for each output sample  $p \in \mathcal{O}$ . We then sum up the squared neighborhood differences. Our goal is to obtain an output  $\mathcal{O}$  with a low energy value. For our normal image resizing applications, we assume as null. Furthermore, we follow the EM-like methodology in [11] to optimize Eq. (6) because of its high quality and generality under different boundary conditions.

We synthesize multiple resolutions through an iterative optimization solver. In Eq. (5), we apply a large  $\alpha$  value in low resolutions to enhance the spatial constraint. This scheme helps preserve the global appearance during synthesis. We reduce the  $\alpha$  value in high resolutions to avoid the repeat and verbatim copying artifacts in the local texel. We use a three-level pyramid in all our experiments and fix  $\alpha = 0.65, 0.25, 0.1$  for each level in ascending order. This setting is effective in all our experiments.

Given a composite texture, content discontinuities may be observed in the boundary areas because we synthesize each sub-texture in the output image separately. To reduce this artifact, we develop each boundary by expanding four pixels on both the inward and outward sides. We then detect an overlapping area between two sub-textures in the output image. Subsequently, we re-synthesize these boundary pixels by taking the entire input texture as an example. This scheme helps to effectively maintain content consistency among sub-textures.

## 6 Results and discussions

**Performance** We implemented our method on a PC with an Intel Core(TM) i7 950 CPU, 3.06 GHz, 8 GB RAM, nVidia Geforce GTX 770 GPU, and 2048 MB video memory. The texture analysis and initialization steps in our framework are both performed in real time. Our EM-based synthesis algorithm is fully implemented on the GPU with CUDA and the timing of this step ranges between 16 (input  $128 \times 128$ , output  $256 \times 256$ ) and 80 seconds (input  $192 \times 192$ , output  $400 \times 400$ ) in this study. Table 1 displays the runtime of some examples. The time covers the cost of both texture analysis and synthesis.

**Evaluation and discussion** Figures 1, 2, 5 and 9 exhibit the results of our feature-aware texture synthesis and the comparison with other method. The control maps of all of the samples are generated automatically. Additional results are shown in the supplemental material. The findings with perspective-aware synthesis [3] are all subscribed by the original authors. The results of texture optimization [11] and parallel synthesis [14] are generated through our own imple-

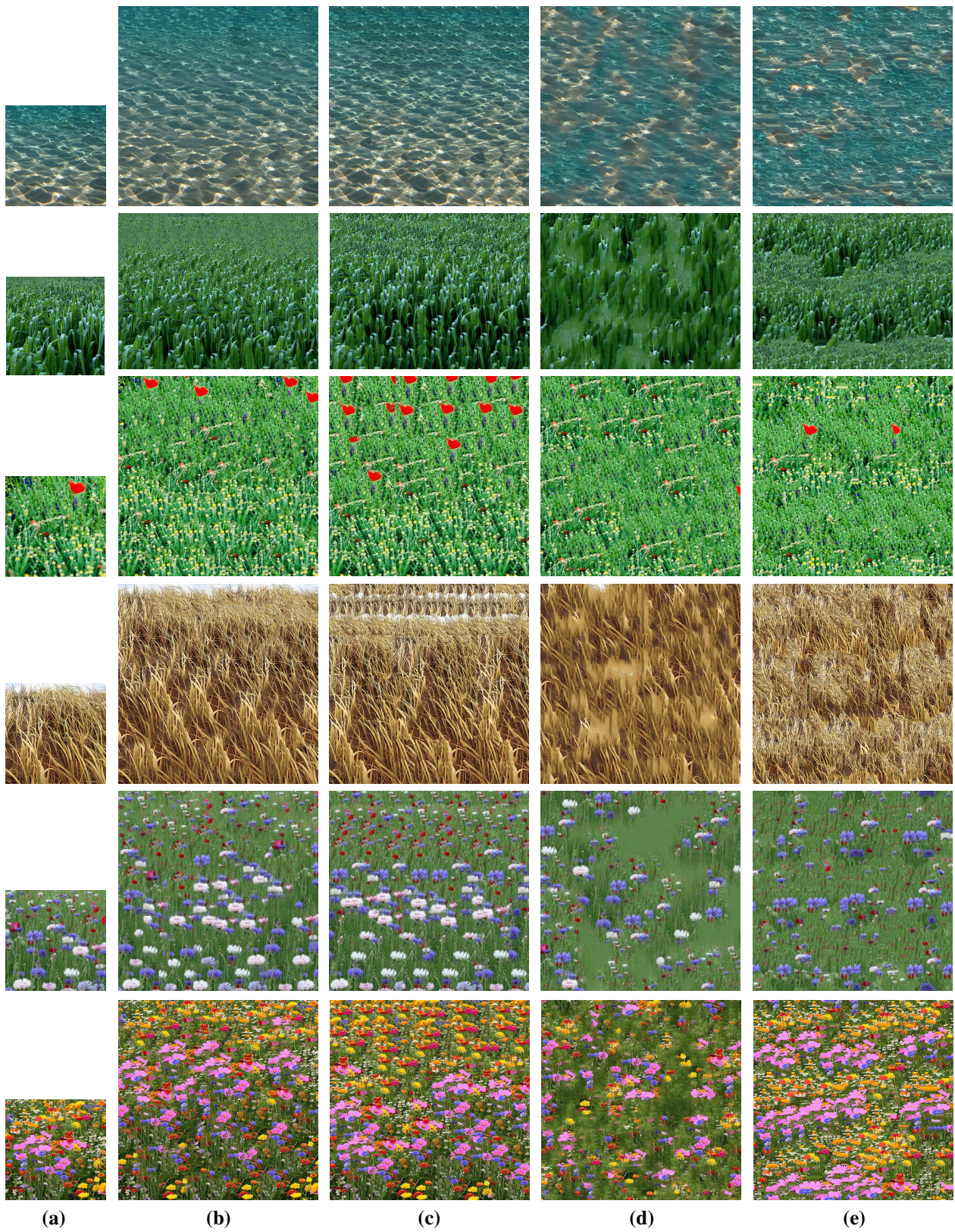
**Table 1** Running time of some examples

Image	Input res.	Output res.	Time (s)
Figure 1	$192 \times 192$	$400 \times 400$	63
Figure 2	$192 \times 192$	$400 \times 400$	71
Figure 5	$160 \times 160$	$400 \times 400$	56
Figure 6	$128 \times 128$	$256 \times 256$	18
Figure 9 (water)	$192 \times 192$	$400 \times 400$	50
Figure 9 (grass 1)	$166 \times 166$	$384 \times 384$	43
Figure 9 (flowers 1)	$128 \times 128$	$400 \times 400$	38
Figure 9 (grass 2)	$192 \times 192$	$400 \times 400$	71
Figure 9 (flowers 2)	$192 \times 192$	$400 \times 400$	80
Figure 9 (flowers 3)	$192 \times 192$	$400 \times 400$	78
Figure 10	$192 \times 192$	$400 \times 400$	69
Figure 11	$128 \times 128$	$256 \times 256$	16

mentation. Our method utilizes the visual features extracted from the content of the sample input in the process of synthesizing a natural texture to guide the synthesis process. Thus, it can easily reproduce the global visual features of the sample in the result. We present a synthesis result obtained with a user-designed control map in Fig. 11.

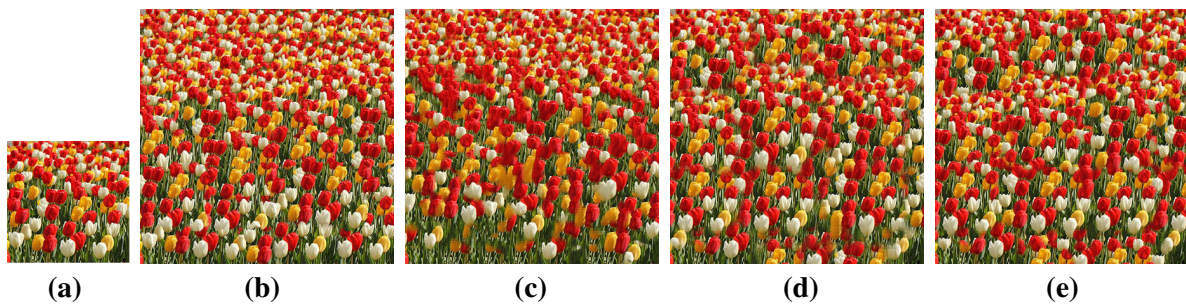
Our method can generate more natural textures than the two state-of-the-art general texture synthesis algorithms in [11] and [14] because our technique preserves global visual effects such as perspective variation, composition of sub-textures, and global color patterns. Our technique is more flexible and effective than those reported in previous works that can generate results with similar appearances, such as those of [3, 4, 12, 14, 29]. Although the method of Dong et al. [3, 4] can also generate textures with perspective effects, the results often display the clear repeat copying artifacts of the same pattern in a small area and the verbatim copying of parts of the sample images. On the contrary, the control maps must typically be generated manually as in [3] and [4], whereas our content analysis method can extract usable control maps (feature maps) from most examples automatically. The input must either be isotropic or treated as isotropic (no scale variations among the texels) in [29] and [14]. Thus, the concept of these schemes differs completely from that presented in our work. The graphcut-based texture synthesis method in [12] can generate a perspective texture by re-sampling the input image (without perspective effect) to different scales and by constraining different portions of the output texture to copy from various scales of the sample. The scope of this method varies from that of our work because we deal directly with perspective examples. However, the graphcut-based method can only expand an input texture with perspective effect in its non-perspective direction, i.e., the output image must be similar to the input in height and width. However, our method can generate arbitrary sizes of output images. As shown in





**Fig. 9** Comparison of synthesis results with different methods. **a** Input. **b** Our results. **c** Perspective aware [3]. **d** Texture optimization [11]. **e** Parallel Synthesis [14]





**Fig. 10** Comparison of our result with those that use fixed-size patches in initialization and synthesis. **a** Input. **b** Our result. **c** Using  $16 \times 16$  patches. **d** Using  $32 \times 32$  patches. **e** Using  $48 \times 48$  patches

**Fig. 11** Extended to user-assisted texture design. We use a user-designed output size map to synthesize the result to get a new global appearance

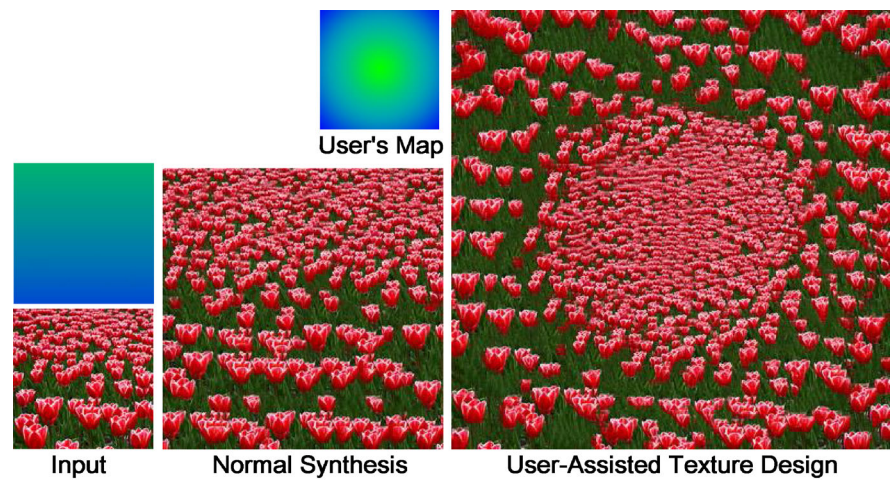
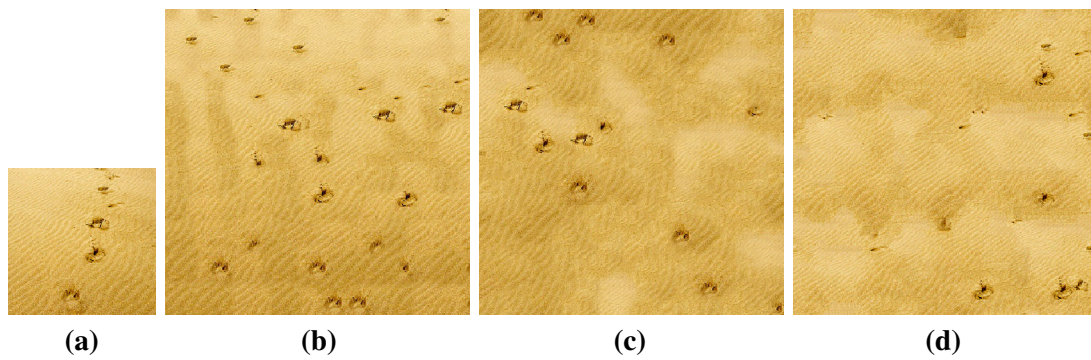


Fig. 9, the output aspect ratio of the green grass texture is different with the sample input, but we can also generate a good result benefiting from the use of feature map in the synthesis framework.

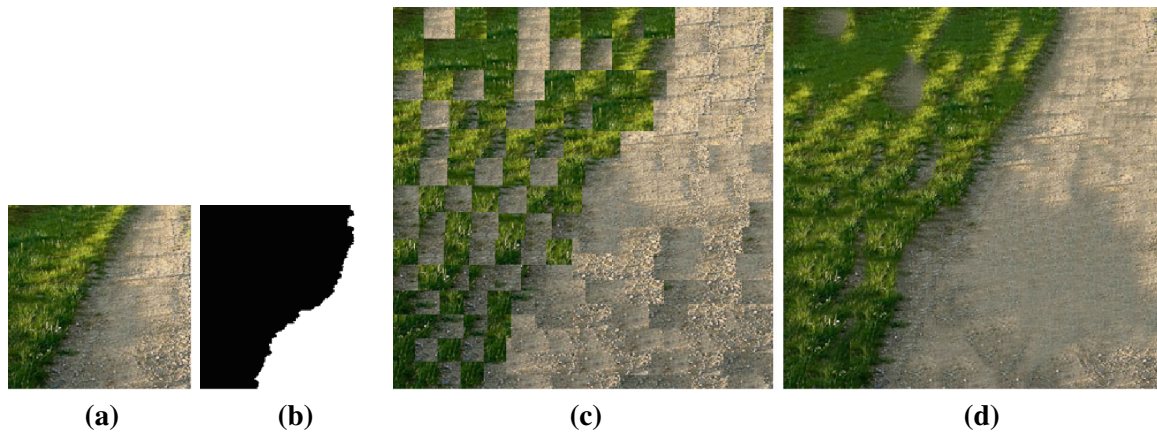
We apply variant sizes of patches during both initialization and synthesis in our algorithm. This scheme is necessary in the preservation of both the global visual effect and in local texel completeness. In Fig. 10, we depict the results of using fixed-size patches in our algorithm to synthesize a natural texture. Texel incompleteness artifacts are generated in the areas wherein texels are large (e.g., Fig. 10c, d) if we use small fixed patches. The perspective effects are limited (by the unexpected mixture of small and large texels) if we apply large fixed patches (e.g., Fig. 10d, e). Therefore, we synthesize the areas of large texels with large patches to preserve the completeness of local texels by assuming that a patch that measures approximately  $48 \times 48$  is large enough to cover the entire or most parts of the largest texels. By contrast, we use small patches to synthesize the areas of small texels to preserve global visual effects by assuming that a patch that measures approximately  $16 \times 16$  is small enough to maintain a continuously global variation. In particular, we aim to avoid the incorrect mixture of large and small texels.

*Limitations* Although our algorithm integrates the features of the visual characteristics of the sample input into the texture synthesis process, some of the high-level semantic appearances may still be damaged in the result. For example, the original spatial layout of the footprints is no longer observed in our result, as shown in Fig. 12. This problem may be solved by integrating object-level information. The feature detection phase may require user interaction in some examples. The perspective direction generated by our evaluation algorithm may also be inaccurate when the input example is evidently asymmetrical along the right direction (e.g., Fig. 8a). This problem can be addressed by generating the size map manually using the method in [3]. Automatic texture segmentation may also fail to produce good type maps for some composite textures. Consequently, the quality of the synthesis results is affected. We manually generate an incorrect type map (Fig. 13) to guide the synthesis process and observed obvious artifacts in the result (Fig. 13d). This finding is mainly attributed to the poor initialization result (Fig. 13c) in which sub-textures are incorrectly composited in some regions. Therefore, we must refine the type map by manually segmenting the input texture to obtain a good synthesis result. Moreover, we still cannot generate a real-time

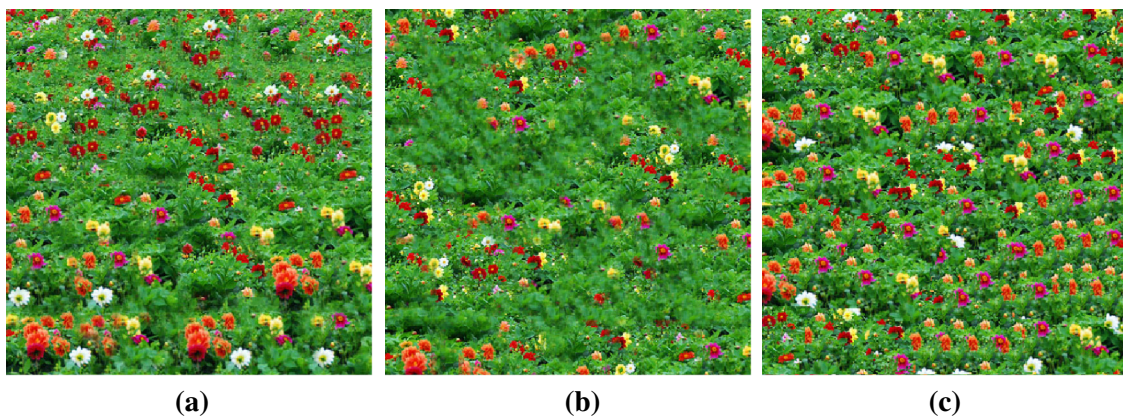




**Fig. 12** Our algorithm may generate an undesired result that damages the original spatial distribution of the semantic objects in the result because we do not possess object-level information regarding the input exemplar. **a** Input. **b** Our result. **c** TexOpt [11]. **d** ParaSyn [14]



**Fig. 13** Result may be unsatisfied with bad type mask. **a** Input. **b** Mask. **c** Initialization. **d** Result



**Fig. 14** We must conduct an accurate search during the neighborhood matching step instead of an approximate nearest neighborhood search ( $K$ -coherence or  $K$ -means) to avoid the blurry blending in **b** and the

local repeat copying artifacts in **c** because of the obviously anisotropic characteristics in natural textures. **a** Accurate search. **b**  $K$ -means. **c**  $K$ -coherence

performance as in the parallel synthesis methods of [13] and [14] although synthesis operation (the most time-consuming step) is fully implemented on GPU as in Table 1. This failure is ascribed to the fact that we must search exhaustively for each  $Z_p$  to obtain the best neighborhood patch  $Z_q$  from

the exemplar in each neighborhood matching iteration. The large resolutions of both the input ( $192 \times 192$ ) and the output textures ( $400 \times 400$ ) in our experiments also increase the runtime of our system (previous works usually synthesize a  $128 \times 128$  or  $256 \times 256$  texture using  $64 \times 64$  or



128 × 128 exemplars). The algorithm can be accelerated using  $K$ -means [11] or  $K$ -coherence [14,22] to determine an approximate nearest neighborhood. However, these neighborhoods may differ significantly from the exact one given the evident anisotropy in natural texture, which tends to induce blurry blending or local repeat copying artifacts (e.g., Fig. 14).

## 7 Conclusion

We have presented a method that facilitates feature-aware texture synthesis for samples with globally variant appearances. Our method is guided by a feature map that roughly masks the visual characteristics in the texture images while the synthesis results produced by our algorithm either preserve the appearances of those characteristics or present user-defined feature constraints. These results are difficult or at least cumbersome to obtain with current software; therefore, we plan to incorporate object-level information into the synthesis process of our future work to address the textures constructed by obvious discrete elements.

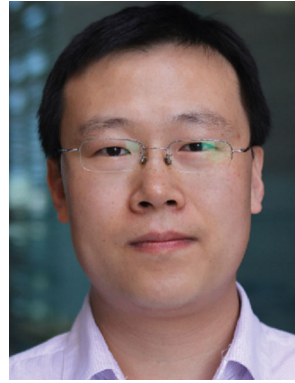
**Acknowledgments** We thank anonymous reviewers for their valuable input. We thank Chongyang Ma for providing some results and valuable comments in the preparation of this paper. This work is supported by National Natural Science Foundation of China under project Nos. 61172104, 61271430, 61201402, 61372184, 61372168, and 61331018.

## References

- Ashikhmin, M.: Synthesizing natural textures. In: *SIGGRAPH '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pp. 217–226. ACM Press, New York, NY, USA (2001)
- Bonet, J.S.D.: Multiresolution sampling procedure for analysis and synthesis of texture images. In: *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pp. 361–368. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA (1997)
- Dong, W., Zhou, N., Paul, J.C.: Perspective-aware texture analysis and synthesis. *Vis. Comput.* **24**(7–9), 515–523 (2008)
- Dong, W., Zhou, N., Paul, J.C.: Interactive example-based natural scene synthesis. In: *Third International Symposium on Plant growth modeling, simulation, visualization and applications (PMA)*, pp. 409–416 (2009)
- Efros, A.A., Freeman, W.T.: Image quilting for texture synthesis and transfer. In: *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 341–346. ACM Press, New York, NY, USA (2001)
- Efros, A.A., Leung, T.K.: Texture synthesis by non-parametric sampling. In: *ICCV '99: Proceedings of the International Conference on Computer vision*, vol. 2, p. 1033. IEEE Computer Society, Washington, DC, USA (1999)
- Eisenacher, C., Lefebvre, S., Stamminger, M.: Texture synthesis from photographs. *Comput. Graph. Forum* **27**(2), 419–428 (2008)
- Han, C., Risser, E., Ramamoorthi, R., Grinspun, E.: Multiscale texture synthesis. *ACM Trans. Graph.* **27**(3), 1–8 (2008)
- Hoang, M.A., Geusebroek, J.M., Smeulders, A.W.M.: Color texture measurement and segmentation. *Signal Process.* **85**(2), 265–275 (2005)
- Kim, V.G., Lipman, Y., Funkhouser, T.: Symmetry-guided texture synthesis and manipulation. *ACM Trans. Graph.* **31**(3), 22:1–22:14 (2012)
- Kwatra, V., Essa, I., Bobick, A., Kwatra, N.: Texture optimization for example-based synthesis. *ACM Trans. Graph.* **24**(3), 795–802 (2005)
- Kwatra, V., Schödl, A., Essa, I., Turk, G., Bobick, A.: Graphcut textures: image and video synthesis using graph cuts. *ACM Trans. Graph.* **22**(3), 277–286 (2003)
- Lefebvre, S., Hoppe, H.: Parallel controllable texture synthesis. *ACM Trans. Graph.* **24**(3), 777–786 (2005)
- Lefebvre, S., Hoppe, H.: Appearance-space texture synthesis. *ACM Trans. Graph.* **25**(3), 541–548 (2006)
- Li, L., Jin, L., Xu, X., Song, E.: Unsupervised color-texture segmentation based on multiscale quaternion gabor filters and splitting strategy. *Signal Process.* **93**(9), 2559–2572 (2013)
- Liu, Y., Lin, W.C., Hays, J.: Near-regular texture analysis and manipulation. *ACM Trans. Graph.* **23**(3), 368–376 (2004)
- Ma, C., Wei, L.Y., Lefebvre, S., Tong, X.: Dynamic element textures. *ACM Trans. Graph.* **32**(4), 90:1–90:10 (2013)
- Ma, C., Wei, L.Y., Tong, X.: Discrete element textures. *ACM Trans. Graph.* **30**(4), 62:1–62:10 (2011)
- Park, H., Byun, H., Kim, C.: Multi-exemplar inhomogeneous texture synthesis. *Comput. Graph.* **37**(1–2), 54–64 (2013)
- Rosenberger, A., Cohen-Or, D., Lischinski, D.: Layered shape synthesis: automatic generation of control maps for non-stationary textures. *ACM Trans. Graph.* **28**(5), 107:1–107:9 (2009)
- Rubner, Y., Tomasi, C., Guibas, L.J.: The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vis.* **40**(2), 99–121 (2000)
- Tong, X., Zhang, J., Liu, L., Wang, X., Guo, B., Shum, H.Y.: Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Trans. Graph.* **21**(3), 665–672 (2002)
- Wei, L.Y.: Multi-class blue noise sampling. *ACM Trans. Graph.* **29**(4), 79:1–79:8 (2010)
- Wei, L.Y., Lefebvre, S., Kwatra, V., Turk, G.: State of the art in example-based texture synthesis. In: *Eurographics 2009, State of the Art Report, EG-STAR*, pp. 93–117. Eurographics Association (2009)
- Wei, L.Y., 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*, pp. 479–488. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA (2000)
- Wu, Q., Yu, Y.: Feature matching and deformation for texture synthesis. *ACM Trans. Graph.* **23**(3), 364–367 (2004)
- Yan D-M., Wonka P.: Gap Processing for Adaptive Maximal Poisson-disk Sampling. *ACM Trans. Graph.* **32**(5), 148:1–148:15 (2013)
- Zalesny, A., Ferrari, V., Caenen, G., Van Gool, L.: Composite texture synthesis. *Int. J. Comput. Vis.* **62**(1–2), 161–176 (2005)
- Zhang, J., Zhou, K., Velho, L., Guo, B., Shum, H.Y.: Synthesis of progressively-variant textures on arbitrary surfaces. In: *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pp. 295–302. ACM, New York, NY, USA (2003)



**Fuzhang Wu** received his B.Sc. degree in computer science from Northeast Normal University, People's Republic of China, in 2010. He is working toward the Ph.D. degree in the Sino-French Laboratory and National Laboratory of Pattern Recognition at the Institute of Automation, Chinese Academy of Sciences. His research interests include image synthesis.



**Dong-Ming Yan** is a research scientist at King Abdullah University of Science and Technology (KAUST), and an Associate Professor at LIAMA-NLPR, CASIA. He received his Ph.D. from Hong Kong University in 2010, and his Master and Bachelor degrees both from Tsinghua University in 2005 and 2002, respectively. His research interests include computer graphics, geometric processing and visualization.



**Weiming Dong** is an associate professor in the Sino-French Laboratory (LIAMA) and National Laboratory of Pattern Recognition (NLPR) at Institute of Automation, Chinese Academy of Sciences. He received his B.Sc. and M.Sc. degrees in computer science in 2001 and 2004, both from Tsinghua University, People's Republic of China. He received his Ph.D. in computer science from the University of Lorraine, France, in 2007. His research

interests include image synthesis and image analysis. Weiming Dong is a member of ACM and IEEE.



**Xiaopeng Zhang** received his M.S. degree in mathematics from Northwest University in 1987, and the Ph.D. degree in computer science from Institute of Software, Chinese Academy of Sciences (CAS), in 1999. He is a professor in the Sino-French Laboratory (LIAMA) and (National Laboratory of Pattern Recognition) at Institute of Automation, CAS. His main research interest is computer graphics and pattern recognition. He received the National Scientific

and Technological Progress Prize (Second Class) in 2004. Xiaopeng Zhang is a member of ACM and IEEE.

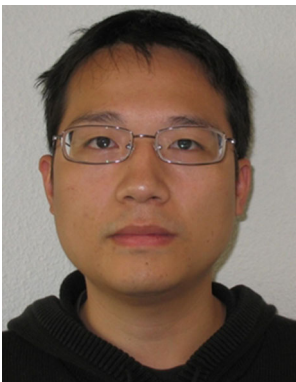


**Yan Kong** is a Ph.D. candidate in the Sino-French Laboratory (LIAMA) and National Laboratory of Pattern Recognition (NLPR) at Institute of Automation, Chinese Academy of Sciences. He received his B.Sc. in computer science in 2011 from Beijing Jiaotong University, People's Republic of China. His research interests include image synthesis and image processing.



**Jean-Claude Paul** is a director of research at INRIA and a professor at Tsinghua University, Beijing, China. He received his Ph.D. degree in mathematics from University of Paris XI and graduated in architecture design from the French National School of Fine Arts (ENSBA) in 1976. In 1995, he obtained the Academie des Sciences Prize and the Academie des Beaux Arts Prize, for both his artistic and scientific work. His research interests include realistic rendering,

geometry processing and curves and surface theory.



**Xing Mei** is an assistant professor in the Sino-French Laboratory (LIAMA) and National Laboratory of Pattern Recognition (NLPR) at Institute of Automation, Chinese Academy of Sciences (CASIA). He received his B.Sc. degree in electronic engineering in 2003 from the University of Science and Technology of China (USTC). He received his Ph.D. degree in 2009 from CASIA. His research interests include image processing, computer vision and computer graph-

ics. Xing Mei is a member of IEEE.