Visual Comput (2006) 22: 936–945
DOI 10.1007/s00371-006-0079-2

ORIGINAL ARTICLE

Jianbing Shen
Xiaogang Jin
Xiaoyang Mao
Jieqing Feng

Completion-based texture design using deformation

Published online: 25 August 2006
© Springer-Verlag 2006

J. Shen · X. Jin (✉) · J. Feng
State Key Lab of CAD & CG, Zhejiang University, Hangzhou, 310027, P.R. China
{shenjianbing, jin, jqfeng}@cad.zju.edu.cn

X. Mao
University of Yamanashi, Japan
mao@yamanashi.ac.jp

Abstract In this paper, we present a novel approach for designing a variety of large textures from a single small sample texture. Firstly, the original small texture is segmented into layers, each of which contains one particular texture element. Secondly, each layer is deformed using a set of chaotic-based transformation operations. Thirdly, all the deformed layers are added together to form a new texture, which is a natural variation of the original sample texture. Since each layer is deformed independently, adding the deformed layers together usually results in a texture with overlapping regions and holes. We employ the graphcut algorithm and an example-based image inpainting technique to seamlessly patch the overlapping regions and to fill the holes. Moreover, an optimized graphcut synthesis algorithm and a new cyclic texture synthesis technique are also developed for efficiently creating large seamless textures. As a result, our approach shows particular strength in generating a large variety of textures from a single sample texture while avoiding highly repetitive patterns. Our experiments demonstrate that the proposed technique can also be used for other texture synthesis applications, such as texture synthesis from multiple samples.

Keywords Texture design · Layer completion · Cyclic texture · Deformation

1 Introduction

Texture has long been a fascinating topic in human perception. It is a ubiquitous and stochastic visual experience, and can describe a wide variety of surface characteristics such as terrain, plants, minerals, fur and skin. Texture is important for many applications in computer graphics, vision, and image processing. However, it is still a challenging task to design textures with a variety of realistic natural patterns.

Procedural textures provide great flexibility and allow for fine tuning of parameters to control the visual pattern. However, unfortunately, they require programming skills that are out of the reach of most users. Spatial textures refer to those textures found on the surface of 3D shapes, such as the patterns on the weathered stone [12] and spots on a leopard [32, 37]. Those textures are usually formed due to natural processes and hence are difficult to achieve with photo editing software or to model as procedural textures. Natural textures, however, are extremely complex, as shown by research in human vision, e.g., [18], and traditional linear analysis and manifold embedding cannot be used to characterize them. Matusi *et al.* [25] developed a system for designing novel textures in the space of textures induced by an input database. However, their morphable texture interpolation is based on a single one-to-one warping between pairs of texture samples, which

might be too restrictive for textures with highly irregular structures, causing discontinuous mappings of the patches to the original image.

In this paper, we present a novel algorithm for designing a wide variety of textures only from a single small input texture. Our algorithm involves four main components. The first component is a segmentation stage for extracting texture layers roughly, which is based on existing color image segmentation techniques [9, 16]. The second component is deformation operations, which are defined by our chaotic-based deformation operations (such as rotation, translation, mirror and scale). The third component is an example-based image completion method together with a graphcut algorithm for inpainting the hole regions induced by deformation. The fourth component is gradient-based Poisson optimization for removing the visual artifacts produced by the third stage.

In summary, the main contributions of this paper fall into three aspects:

- We develop a new framework for designing a large variety of textures by integrating the techniques of 1) image segmentation, 2) texture synthesis, 3) example-based image completion, and 4) gradient-based Poisson optimization.
- A new set of chaotic-based deformation operations is presented for producing a variety of deformed texture patterns.
- A new, easy and efficient cyclic texture designing method is proposed.

In the rest of the paper, we first introduce the related work on image inpainting and texture synthesis in Sect. 2. Then, in Sect. 3, we discuss the details of our completion-based texture design using our newly defined deformation operations. The new cyclic texture designing method is also presented in Sect. 3. After showing the experimental results in Sect. 4, Sect. 5 concludes the paper and shows some directions for future work.

2 Related work

Our work mainly draws from two research areas: image completion and texture synthesis.

Image completion

Filling regions of an image in a seamless manner, known as image completion, has become an important task for editing digital images. Bertalmio *et al.* [4] conducted a PDE-based method to repair damaged images. Unfortunately, their technique does not seem suitable in terms of both synthesis versatility and computational cost for a number of image manipulation tasks when the destination regions are large or the surrounding image is highly textured. Drori *et al.* [13] incorporated pyramid image approximation and adaptive image fragments to achieve impressive results. However, all these approaches are extremely slow due to the high computational complexity. Moreover, the resultant image may have a blurry artifact when a large reconstructed area lacks texture. As for synthesis capabilities, example-based image inpainting techniques [10, 13, 17, 21, 30, 31, 36] appear as an appealing alternative to variational interpolation. The example-based technique is shown to give very good results at low cost for narrow scratch regions, thin structures, or text overlays. Jia and Tang [21] presented a technique for filling image regions through explicitly segmenting the unknown area into different homogeneous texture areas using a tensor voting method. Criminisi *et al.* [10] proposed an example-based image inpainting algorithm with region filling, where the patch filling order is determined by the angle between the isophote direction and the normal direction of the local filling front — so that the structure of the missing region can be filled with priority. Sun *et al.* [31] introduced a novel structure propagation approach to image completion that requires the user to manually specify the important missing structures.

Texture synthesis

A number of algorithms for characterizing textures according to distributions of multi-scale image properties [3] has been proposed. In non-parametric texture synthesis [8, 14, 19], texture is synthesized one pixel (or one patch) at a time by finding pixels (patches) with a similar neighborhood to the already synthesized pixels (patches) in the sample texture. Brooks and Dodgson [6] presented a related technique that uses the similarity between texture pixels to facilitate editing. Wu and Yu [35] improved patch-based texture synthesis using feature matching and patch deformation to reduce artifacts at patch boundaries. The challenge of combining and mixing textures has been tackled by a number of authors. Cohen *et al.* [8], Hertzman *et al.* [19], Efros and Freeman [14], and Kwatra *et al.* [22] synthesized a non-uniform texture composed of homogeneous patches. Zhang *et al.* [37] generated spatially-varying textures from two input textures. Liu *et al.* [23] described a system to analyze and manipulate photographic textures that allows a user to design novel textures. Similar to the work by Liu *et al.* [23], Matusik *et al.* [25] strove to build a comprehensive texture model, then constructed a texture space that spanned the range of textures induced by a database of natural images.

Kwatra *et al.* [22] also discussed the idea of applying transformation to the patches in their patch-based texture synthesis technique using graphcut. They showed the results using rotation, mirror and scaling, but as they mentioned in their paper, the cost for searching matching patches can increase when the extent of deformation increases. By applying transformations to the layers

of different texture elements, our technique can provide local changes to the size, orientation and relative positions of texture elements, and hence can further improve the stochastic property of output textures, resulting in a wider variety of textures.

3 Completion based texture design using deformation

3.1 Algorithm overview

In summary, the overall texture design algorithm consists of the following four steps (as shown in Fig. 1).

1. An image segmentation technique is used to extract the salient texture layers $I_{L_1}, I_{L_2}, \cdots, I_{L_k}$ (Fig. 1(b1), (b2), (b3)) from the input texture image I (Fig. 1(a)). In order to increase the variations during the deformation, each layer can be further divided into child layers.
2. Chaotic-based deformation operations $O(\cdot)$ are firstly applied to transform the segmented texture layers $I_{L_1}, I_{L_2}, \cdots, I_{L_k}$, then the graphcut method is utilized to stitch the transformed texture layers ($O(I_{L_1})$, $O(I_{L_2})$, $\cdots$, $O(I_{L_k})$) together at the overlapping regions. After that, a deformed texture I'_L (Fig. 1(c))is obtained together with a mask I_M corresponding to the hole regions in I'_L (Fig. 1(d)).
3. An image completion technique is employed to fill the hole regions of the deformed texture I'_L to obtain the inpainted texture I_C (Fig. 1(e)).
4. Steps 1 to 3 are repeated to create a set of small deformed and inpainted textures $I_{C_1}, I_{C_2}, \cdots, I_{C_\kappa}$ (Fig. 1(e), (f)) and then stitched them together to obtain a large texture S using graphcut algorithm, followed by an updating process using gradient-based Poisson optimization (Fig. 1(g)). If necessary, the cyclic texture can be generated by further processing the texture S using our new cyclic texture design technique in Sect. 3.5.

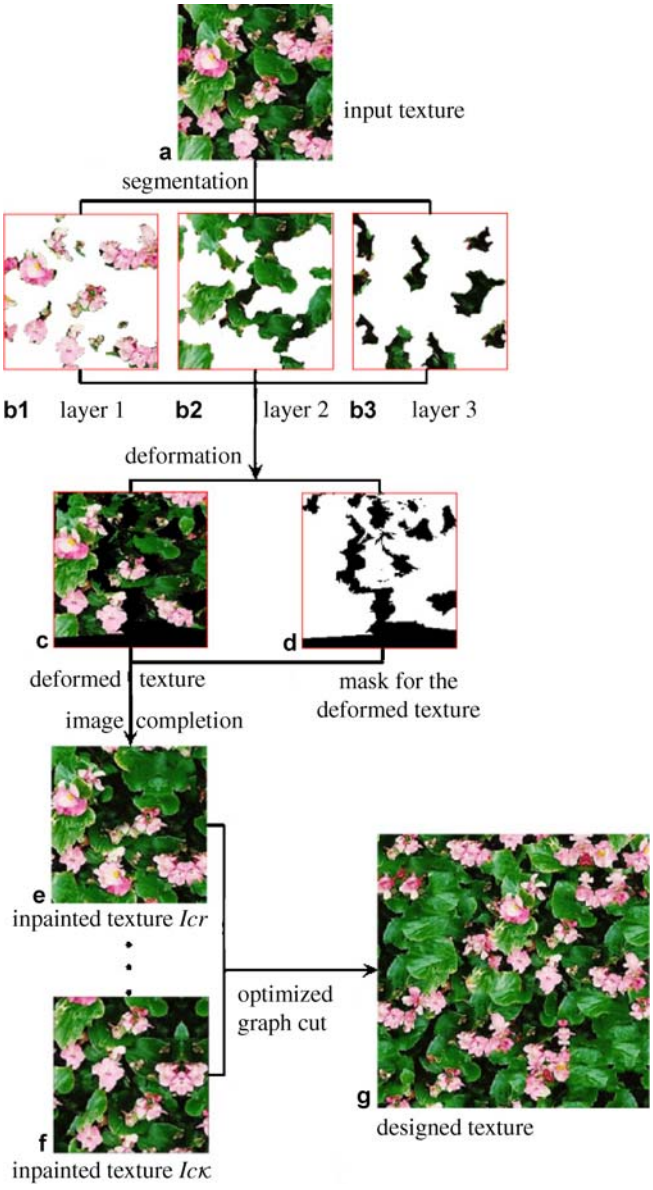


Fig. 1a–g. Proposed texture design algorithm. **a** Small input texture I; **b1**, **b2**, **b3** the segmented layers I_{L1}, I_{L2}, I_{L3}; **c** the deformed texture I'_L; **d** the corresponding mask I_M; **e** inpainted texture I_{C_1}; **f** inpainted texture I_{C_κ}; **g** one of the large designed textures

3.2 Layering the texture

The first step is to segment the input texture I into semantic layers, so that, within each layer, different texture deformation operations can be applied. For example, for the image in Fig. 1(a), we have the following three segmented texture layers: one for the pink flowers (Fig. 1(b1)), one for the green leaves (Fig. 1(b2)), and one for the background. To accomplish this, we use an automatic image segmentation method based on mean-shift [9], combining with efficient graph [16] methods.

Unlike the work by Chuang *et al.* [7], where the layers are extracted manually with the matting method, our whole process for segmenting the texture layers is fully automatic and also very efficient. Readers may argue that automatic segmentation usually fails to produce ideal results. However, a fact that deserves particular mention here is that in our case, the segmentation algorithm actually does not have to be perfect, since the roughly segmented regions will be inpainted by the image completion algorithm described in Sect. 3.4.

3.3 Texture layer deformation

Once we have extracted the texture layers, we apply a set of newly defined texture deformation operations to each

of these layers so as to create various deformed textures. The deformation operations are formulated as follows:

$$O = \{T(d, l),\ R(a, c),\ M(f),\ S(e)\}, \tag{1}$$

where $T(d, l)$ represents the "translation" operation that translates a texture layer along the direction d ($d \in \{\text{up, down, left, right}\}$) with the distance l; $R(a, c)$ is the "rotation" operation that rotates a texture layer with the angles a in a direction given by parameter c, $c \in \{\text{clockwise, anti-clockwise}\}$; $M(f)$ defines the operation "mirror" that mirrors a texture layer in a way specified by parameter f, $f \in \{mirror_{right}^{left}, mirror_{down}^{up}\}$; $S(e)$ is the "scale" operation that changes the size of a texture layer with the scaling extent e, which is set to be $e \in (0.8, 1.2)$ in our experiments.

In order to design a wide variety of natural textures, we need a method to determine which of the above deformation operations should be used. A chaotic-based sequence generator is one possible solution, since it is non-periodic, non-convergent and extremely sensitive to the initial condition [20]. Among the various nonlinear chaotic maps, the most famous and widely used map is the logistic map, which is one of the simplest systems exhibiting order-to-chaos transitions. The basic logistic map is formulated as:

$$x_{k+1} = 1 - \lambda \cdot {x_k}^2, \tag{2}$$

where $\lambda \in [0, 2]$. When $\lambda = 2$, the chaotic maps are called full-maps, which have good statistical properties [20, 24]. x_0 and λ are the initial conditions for producing a chaotic sequence. In our experiments, x_0 and λ are initialized as $x_0 = 0.76234783$ and $\lambda = 2$, respectively.

For the segmented texture layers $I_{L_i}, i = \{1, \cdots, k\}$, the whole process of texture deformation can be summarized as follows.

- Input: $I_{L_1}, I_{L_2}, \cdots, I_{L_k}, x_0, \lambda$.
- Step 1: generate a chaotic sequence $H = \{h(k) = |x(k)|, k = 0, 1, 2, \cdots, \}$ using formula (2), then map the above set H into the range (0, 255).
- Step 2: select $O(m)$ as the operation to be applied to layer k. Here $m = \text{mod}(\text{mod}(h(k), u), v)$, and $O(m)$ is the m-th operation of the operation set O in (1). The parameter of the selected operation, which controls the extent of the deformation, is determined by n ($n = \text{mod}(h(k), u)$). Parameters u, v and ξ are constants for controlling the range of the transformation operation. For example, if $u = 10$, $v = 4$, $\xi = 0.13$, $h(k) = 37$, then $m = 3$, $n = 7$, and hence the fourth operation "scale" $S(e)$ is selected, and the scaling extent $e = \xi \cdot n = 0.91$.
- Output: transformed layers $I_{L_1'}, I_{L_2'}, \cdots, I_{L_k'}$.

All the above transformation operations or their combinations can be applied on the texture layers (or child texture layers) simultaneously, and can also be iterated multiple times to achieve the desired texture design results.

3.4 Layer completion

Since each layer is transformed independently, an overlapping of texture elements and holes can be found among the layers after transformation. Large overlaps will usually produce discontinuity in texture patterns, while large holes will cause texture details to be missing after the layers have been added together. Because of its simplicity and its capacity to handle textured regions, we use the new modified example-based completion algorithm to fill the holes among the layers. This algorithm is based on the work of Criminisi *et al.* [10] .

To solve the overlapping problem, we cut the overlapping regions with a maximum flow or minimum cost graphcut algorithm [22], and stitch the layers along the cut lines. To use a graphcut algorithm, we define each location in the overlapped region as a vertex v. Let $C_i(v)$ and $C_j(v)$ be the color values at the location v in the two overlapped layers L_i and L_j, respectively. Then the weight function $W(v_s, v_t)$ between two vertices v_s and v_t can be defined as follows:

$$\begin{aligned} W(v_s, v_t, L_i, L_j) = &\|C_i(v_s) - C_j(v_s)\| \\ &+ \|C_i(v_t) - C_s(v_t)\|, \end{aligned} \tag{3}$$

where $\|\cdot\|$ denotes the Euclidean distance between color values. After defining the above weight function, the minimal cut can be easily computed by a standard graphcut algorithm [5, 29].

After stitching layers together along the minimum cut lines, we obtain the single deformed texture image I_L' (Fig. 1(c)). A mask image I_M (Fig. 1(d)) is also created according to the hole regions in I_L',

$$I_M(p) = \begin{cases} 1, & \forall p \in \sum_{i=1}^{k} O(I_{L_i}) \\ 0, & \textit{Otherwise} \end{cases} \tag{4}$$

where $O(\cdot)$ denotes the deformation operations defined in Sect. 3.3.

We use a new modified inpainting algorithm to fill the hole regions indicated by I_M. Criminisi *et al.* [10] used the angle between the isophote direction and the normal direction of the local boundary to determine the searching order of the patches, so that the structure of the missing region can be filled before filling in the texture. For the purpose of texture design, however, we need to inpaint more texture information rather than structural information. Therefore, we can simplify the "filling the highest confidence first" approaches proposed in [10, 31, 36] by using the simple

scan-line based patch inpainting method. The completion process can be described as follows.

- Input: I, I'_L, I_M.
- Step 1: a target patch $\Psi_t \in I'_L$ is selected in a scan-line filling order. Searching source image I to determine a source patch $\Psi_s \in I$ shows the highest similarity between Ψ_s and Ψ_t.
- Step 2: copy the whole patch data from Ψ_s to Ψ_t, use graphcut to update the overlapped region, and set $I_M(p) = 1$ for $\forall p \in \Psi_t$;
- Step 3: if there exists a point p so that $I_M(p) = 0$, go back to Step 1.
- Output: the completed texture I_C.

3.5 Cyclic texture design

With existing non-parametric texture synthesis techniques [14, 22, 25], cyclic textures can be generated by imposing a periodic boundary condition in searching for the candidate patches from the input texture. In this paper, we propose a very easy and efficient method to create cyclic textures. The basic idea is that if we cut a tube and open it, we will obtain a sheet whose two sides satisfy the periodic condition. Therefore, given an arbitrary texture, we can make it satisfy periodic condition in horizontal direction simply by first stitching its left and right sides together with a graphcut algorithm to form a tube, and then cut the tube along an arbitrary vertical line. Let w and h be the width and

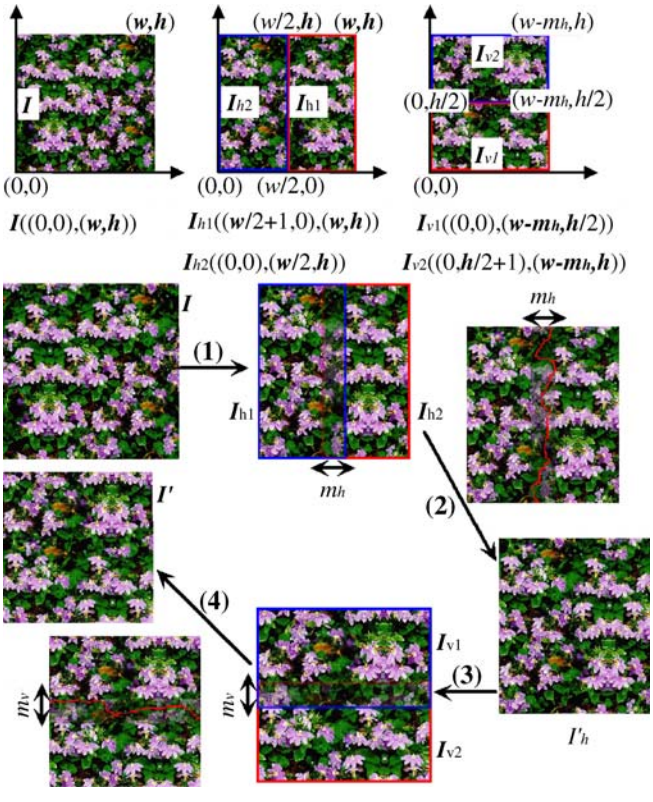


Fig. 2. Illustration of a cyclic texture design algorithm

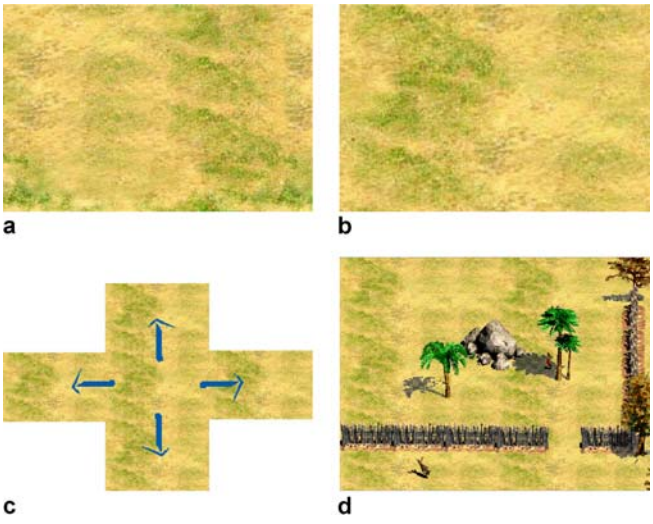


Fig. 3a–d. An example of cyclic texture. **a** the source texture; **b** the synthesized cyclic texture; **c** seamless tiling in each direction (left, right, up, down); **d** a snapshot of applying the cyclic texture for editing terrain texture maps in a video game application

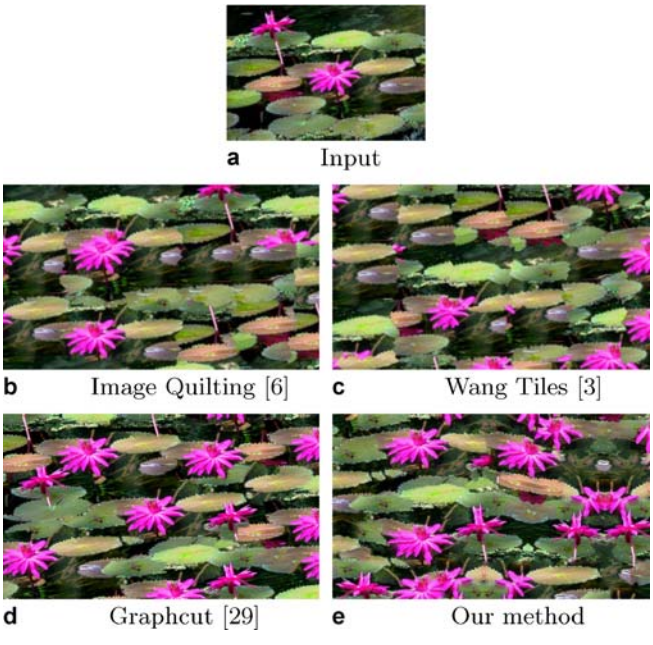


Fig. 4. Comparison of our completion-based algorithm with image quilting [14], Wang tiles [8] and graphcut [22]

height of the large designed texture $I((0,0),(w,h))$. The details of the algorithm are described as follows (Fig. 2).

1. In the horizontal direction, the source texture is rearranged as $I_h = I_{h1} \cup I_{h2}$, where $I_{h1} = I((w/2+1,0),(w,h))$, $I_{h2} = I((0,0),(w/2,h))$.
2. Let the width of the overlapping region be m_h. The graph-cut algorithm is applied to synthesize seamlessly in the horizontal direction. We denote the synthesized texture image as $I'_h = graphcut(I_{h1}, I_{h2})$.
3. In the vertical direction, the image I'_h is rearranged as $I_v = I_{v1} \cup I_{v2}$, where $I_{v1} = I'_h((0,0),(w-m_h,h/2))$, $I_{v2} = I'_h((0,h/2+1),(w-m_h,h))$.
4. Let the height of the overlapping region be m_v. Then run the graph-cut algorithm to synthesize seamlessly in vertical direction. The final synthesized cyclic texture image is represented as $I' = graphcut(I_{v1}, I_{v2})$.

Such a cyclic texture image can be used for seamless tiling in applications such as terrain texture map editing in video games. Figure 3 shows a cyclic texture generated by our new cyclic texture method and its application to seamlessly tiling a terrain.

3.6 Gradient-based Poisson optimization

In the image completion process in Sect. 3.4, for many applications the source patches and the inpainted patches are too dissimilar for a graphcut algorithm alone to result in visually seamless patch updating. Visual artifacts may still exist in the inpainted patches if the graphcut algorithm cannot find the ideal seams.

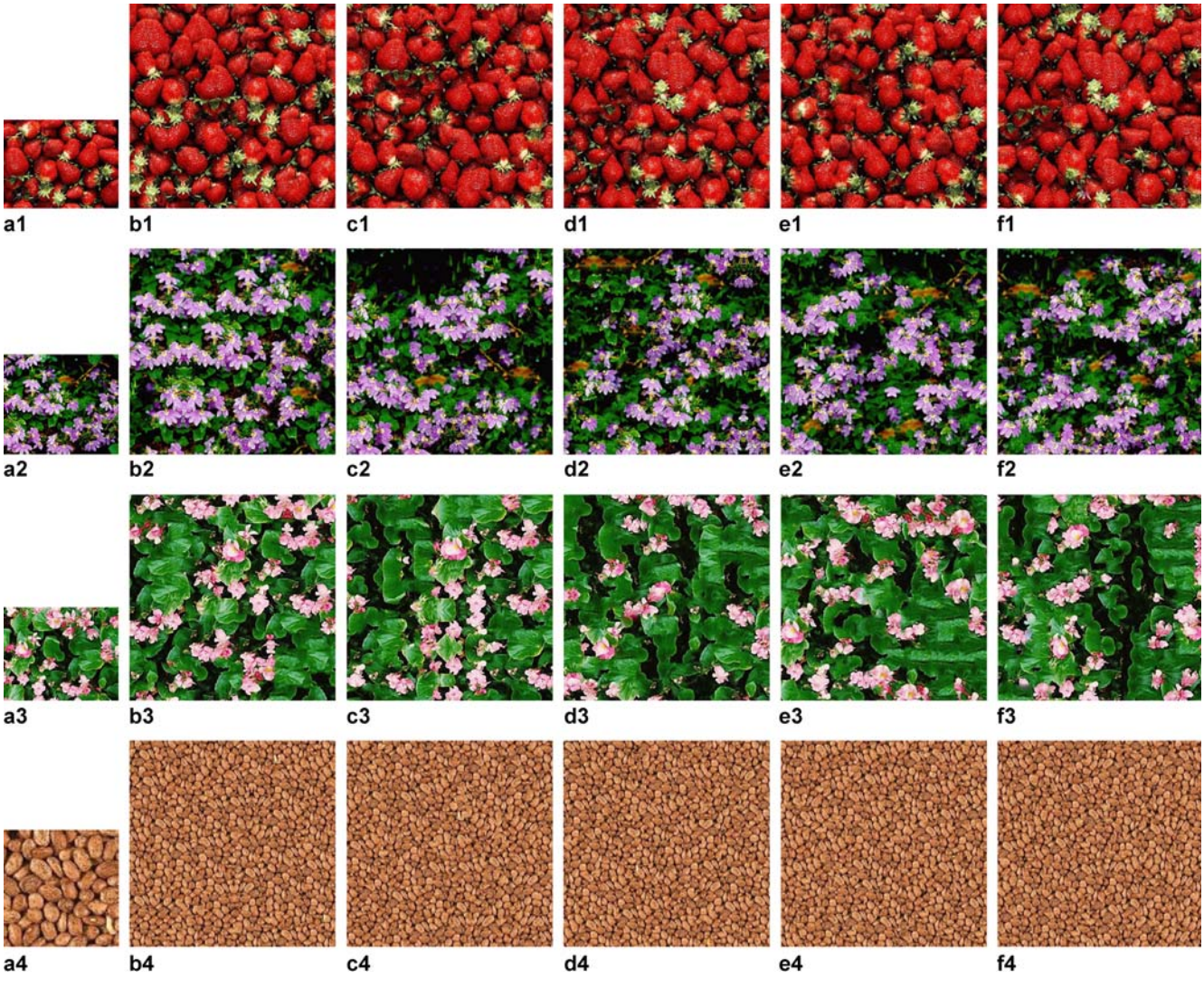


Fig. 5. Examples of spatially varying deformed textures using our completion-based design method. Left columns (**a1**, **a2**, **a3**, **a4**) are the input textures, the others are the deformed textures

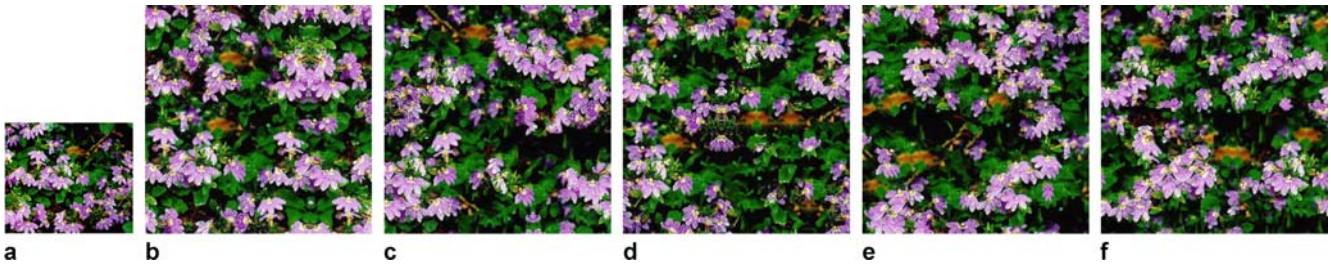


Fig. 6a–f. Examples of cyclic deformed textures obtained by applying the cyclic texture synthesis technique to the textures in Fig. 5b(2), c(2), d(2), e(2), f(2), respectively. Texture size: **a** 268×230; **b**, **c**, **d**, **e**, **f** 310×310

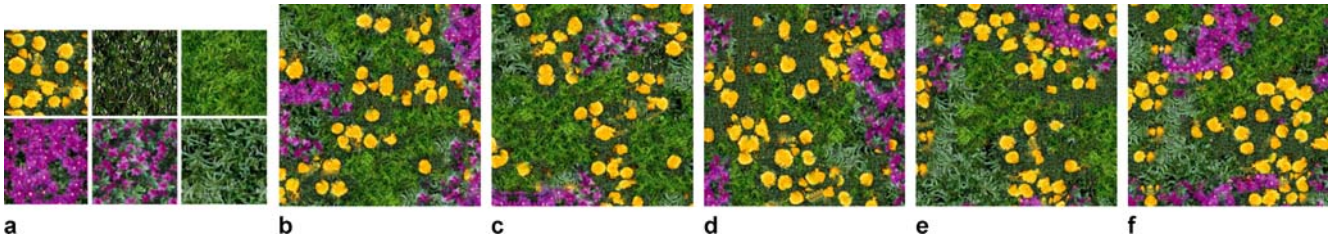


Fig. 7. Blending textures from multi-source textures. Texture size: input: 144×144; output: 360×360

Agarwala *et al.* [1] used a gradient-based Poisson equation to smooth out the color differences between juxtaposed image regions. Similarly, we adopt gradient-based Poisson optimization to smooth out the color differences between the inpainted textures. For a single color channel, the pixel values $I(x, y)$ are re-ordered into a vector $v(x, y)$ [1]; two linear equations are specified by an input gradient $\nabla I(x, y)$ as follows:

$$v_{x+1,y} - v_{x,y} = \nabla I_x(x, y), \tag{5}$$

$$v_{x,y+1} - v_{x,y} = \nabla I_y(x, y). \tag{6}$$

The red, green and blue channels are corrected independently. The Dirichlet boundary condition is the outrider boundary of the completed image. The final optimized texture is obtained by solving a Poisson equation [26, 27]. After this optimization process, the artifacts are reduced and nearly invisible.

4 Experimental results

In our experiments, most of the source texture images were downloaded from a website[1], and for comparison, we tried to use those sample textures that had been used by existing texture synthesis work [8, 14, 22], if possible. All the experiments shown in this section were run on a PC with Pentium IV 1.6GHz CPU + 512MB RAM.

In Fig. 4, we compare our approach with other existing techniques. The result for graphcut was taken from [22], while the other two results were generated by our implementation. The texture size is 268×230 for Fig. 4(a), and 360×360 for Fig. 4((b), (c), (d), (e)). The patch size is selected as 64×64. From the images, we can find that the quality of the texture generated with our approach is superior to that of image quilting [14] and Wang tiles [8], and is comparable to the result produced by graphcut [22].

As demonstrated in Fig. 4, however, a major advantage of our technique over all other existing work is that it can generate a wide variety of large textures with a good stochastic property from a single small sample texture. In the case of a very small sample texture, it is possible that the variations of texture elements contained in the texture are very limited. For example, the sample texture shown in Fig. 4(a) consists of only two different lotus flowers. The techniques that simply use the original patches selected from such a sample texture can result in a large texture consisting of the repetition of those texture elements, such as the one shown in Fig. 4(d), where all the flowers have the sample shape and orientation as either of the two flowers in the sample texture. As shown in Fig. 4(e), however, with the deformation of individual texture layers, our technique can create a texture consisting of flowers of different shapes, sizes and orientations, which looks more like a natural lotus flower field. Figures 5 and 8 give some more examples demonstrating the capability of our technique to create a large variety of textures from a small sample, while maintaining continuity of texture features as well as the shapes of an individual object.

Figure 6 shows the results from further conversion of the textures shown in Fig. 5(b(2), c(2), d(2), e(2), f(2)) into cyclic textures using our new cyclic texture design

[1] *http://www.cc.gatech.edu/cpl/projects/graphcuttextures*.

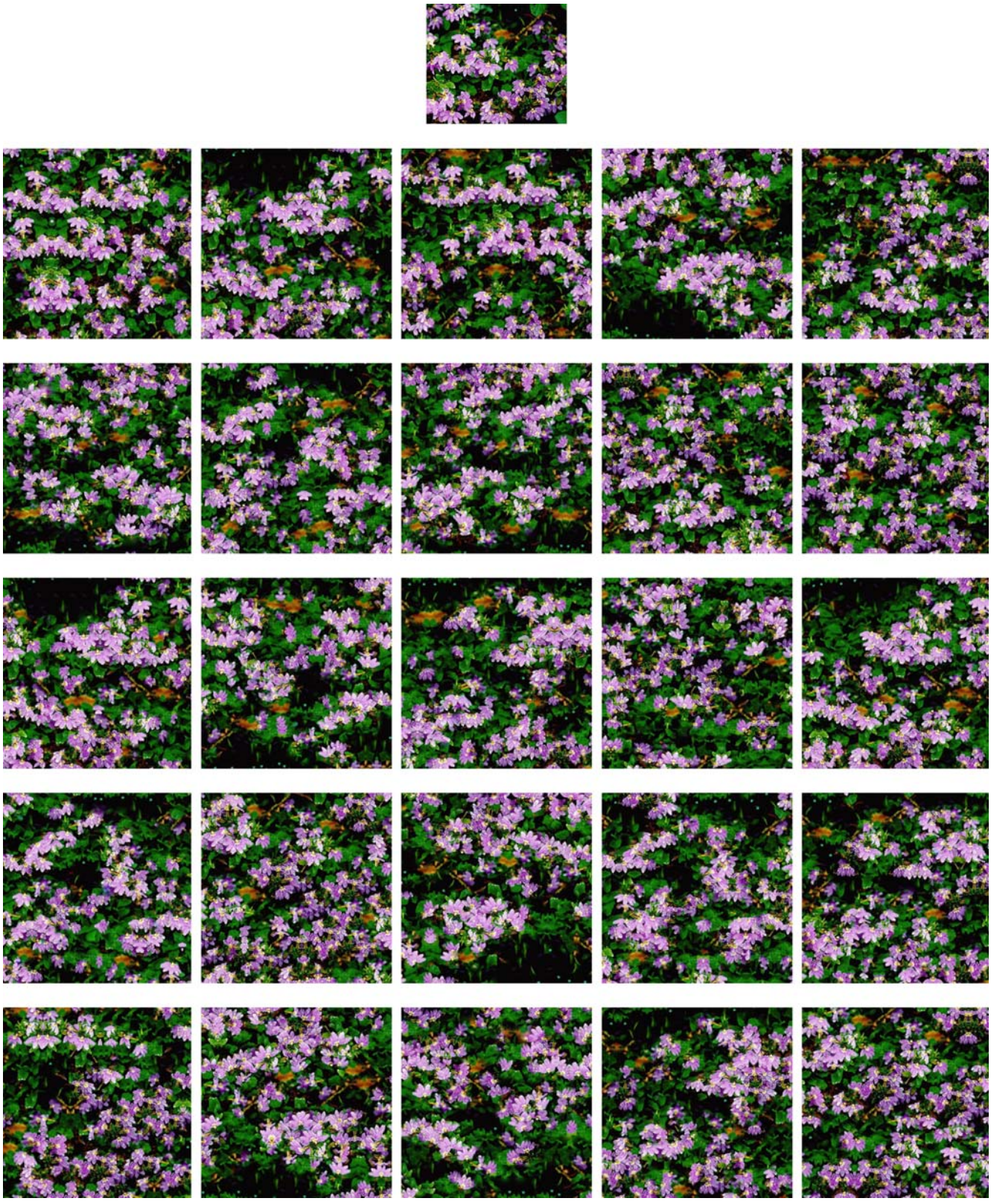


Fig. 8. Deformed textures using our completion based design method. The first row is the small input texture, the others are the deformed textures; Input size: 268×230; O5 mm

technique. Here, the width of overlapping region is set as $m_h = m_v = 50$.

Figure 7 demonstrates another interesting application of our technique, namely texture synthesis from multiple source textures. In Fig. 7, six 144×144 input textures are used to create the output textures of size 360×360 (Fig. 7(b), (c), (d), (e), (f)). Those textures are also the results of our cyclic texture synthesis technique and hence can be used for seamless tiling.

All textures shown in this paper are created full automatically, although the users are allowed to control the extent of the deformation with the parameters discussed in Sect. 3.3. The inpainting patch size is chosen as 32×32.

5 Conclusions and future work

We have presented a novel completion-based approach for designing realistic textures from sample textures. A large advantage of our approach over most existing texture synthesis techniques lies in its capability to create a wide variety of very natural textures from a single small sample texture. The good stochastic property of the resulting textures is achieved through applying chaotically defined deformation operations to the major texture elements extracted from the sample texture, and the local continuity of texture patterns are ensured by employing the newest image completion and gradient-based Poisson optimization techniques. We have also presented a new technique for quickly processing an arbitrary texture to satisfy periodic boundary conditions so that it can be directly used for seamless tiling in some special applications, such as terrain texture map editing in video games and virtual agent environments. However, the broken texture elements will occur when the completion stage cannot inpaint the hole layers ideally or Poisson optimization cannot smooth seamed patches perfectly. Our experimental results also demonstrate that the proposed technique can be applied to other applications such as texture synthesis from multiple sources.

Currently, we are extending our approach from still texture design to video texture [28] design. The difficulties in designing video textures through deformation include maintaining the consistency of the deformed textures between adjacent frames.

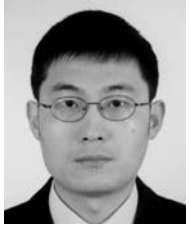
Acknowledgement This work was supported by the 973 program (grant no. 2002CB312101), the National Natural Science Foundation of China (grant no. 60340440422, 60573153, 60533080) and the Program for New Century Excellent Talents in University (grant no. NCET-05-0519).

References

1. Agarwala, A., Dontcheva, M., Agrawala, M., Drucker, S., Colburn, A., Curless, B., Salesin, D., Cohen, M.: Interactive digital photomontage. ACM Trans. Graph. **23**(3), 294–302 (2004)
2. Agrawal, A., Raskar, R., Nayar, S.K., Li, Y.: Removing flash artifacts using gradient analysis. ACM Trans. Graph. **24**(3), 828–835 (2005)
3. Bar-Joseph, Z., El-Yaniv, R., Lischinski, D., Werman, M.: Texture mixing and texture movie synthesis using statistical learning. IEEE Trans. Visual. Comput. Graph. **7**(2), 120–135 (2001)
4. Bertalmio, M., Sapiro, G., Ballester, C., Caselles, V.: Image inpainting. In: Proceedings of SIGGRAPH '00, New Orleans, pp. 417–424. ACM, New York (2000)
5. Boykov, Y., Kolmogorov, V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. IEEE Trans. Patt. Anal. Mach. Intell. **26**(9), 1124–1137 (2004)
6. Brooks, S., Dodgson, N.: Self-similarity based texture editing. ACM Trans. Graph. **21**(3), 653–656 (2002)
7. Chuang, Y.Y., Goldman, D.B., Zheng, K.C., Curless, B., Salesin, D., Szeliski, R.: Animating pictures with stochastic motion textures. ACM Trans. Graph. **24**(3), 853–860 (2005)
8. Cohen, M.F., Shade, J., Hiller, S., Deussen, O.: Wang tiles for image and texture generation. ACM Trans. Graph. **22**(3), 287–294 (2003)
9. Comaniciu, D., Meer, P.: Mean shift: A robust approach towards feature space analysis. IEEE Trans. Patt. Anal. Mach. Intell. **24**(5), 603–619 (2002)
10. Criminisi, A., Pérez, P., Toyama, K.: Region filling and object removal by exemplar-based image inpainting. IEEE Trans. Image Process. **13**(9), 1200–1212 (2004)
11. Dischler, J.-M., Maritaud, K., Lévy, B., Ghazanfarpour, D.: Texture particles. Comput. Graph. Forum **21**(3), 401–410 (2002)
12. Dorsey, J., Edelman, A., Legakis, J., Jensen, H.W., Pedersen, H.K.: Modeling and rendering of weathered stone. In: Proceedings of SIGGRAPH '99, pp. 225–234. ACM, New York (1999)
13. Drori, I., Cohen-Or, D., Yeshurun, H.: Fragment-based image completion. ACM Trans. Graph. **22**(3), 303–312 (2003)
14. Efros, A.A., Freeman, W.T.: Image quilting for texture synthesis and transfer. In: Proceedings of SIGGRAPH '01, pp. 341–346. ACM, New York (2001)
15. Fattal, R., Lischinski, D., Werman, M.: Gradient domain high dynamic range compression. ACM Trans. Graph. **21**(3), 249–256 (2002)
16. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. Int. J. Comput. Vision **59**(2), 167–181 (2004)
17. Harrison, P.: A non-hierarchical procedure for re-synthesis of complex texture. In: Proceedings of International Conference in Central Europe Computer Graphics, Visualization and Computer Vision '01, Czech Republic, pp. 190–197. UNION Agency (2001)
18. Heaps, C., Handel, S.: Similarity and features of natural textures. J. Exper. Psychol. Human **25**(2), 299–320 (1999)
19. Hertzmann, A., Jacobs, C.E., Oliver, N., Curless, B., Salesin, D.H.: Image analogies. In: Proceedings of SIGGRAPH '01, pp. 327–340. ACM, New York (2001)
20. Jakimoski, G., Kocarev, L.: Chaos and cryptography: block encryption ciphers based on chaotic maps. IEEE Trans. Circuits-I **48**(2), 163–169 (2001)
21. Jia, J., Tang, C.K.: Image repairing: robust image synthesis by adaptive tensor voting. In: Proceedings of Conference on Computer Vision and Pattern Regonition '03, Madison, WI, pp. 643–650. IEEE Computer Society (2003)
22. 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)

23. Liu, Y., Lin, W.C., Hays, J.H.: Near regular texture analysis and manipulation. ACM Trans. Graph. **23**(3), 368–376 (2004)
24. Masuda, N., Aihara, K.: Cryptosystems with discretized chaotic maps. IEEE Trans. Circuits-I **49**(1), 28–40 (2002)
25. Matusik, W., Zwicker, M., Durand, F.: Texture design using a simplicial complex of morphable textures. ACM Trans. Graph. **24**(3), 787–794 (2005)
26. Pérez, P., Gangnet, M., Blake, A.: Poisson image editing. ACM Trans. Graph. **22**(3), 313–318 (2003)
27. Press, W., Teukolsky, S., Vetterling, W., Flannery, B.: Numerical Recipes in C: The Art of Scientific Computing. Cambridge University Press, Cambridge (1992)
28. Schödl, A., Szeliski, R., Salesin, D.H., Essa, I.: Video textures. In: Proceedings of SIGGRAPH 00, pp. 489–498. ACM, New York (2000)
29. Sedgewick, R.: Algorithms in C++. Part 5: Graph Algorithms, 3rd edn. Addison Wesley Professional, Reading, MA (2002)
30. Shen, J., Jin, X., Zhou, C., Wang, C.L.: Gradient based image completion by solving the Poisson equation, Comput. Graph. **30**(6), to appear, (2006)
31. Sun, J., Yuan, L., Jia J., Shum, H.Y.: Image completion with structure propagation. ACM Trans. Graph. **24**(3), 861–868 (2005)
32. Walter, M., Fournier, A., Meneveaux, D.: Integrating shape and pattern in mammalian models. In: Proceedings of SIGGRAPH '01, pp. 317–326. ACM, New York (2001)
33. Wexler, Y., Shechtman, E., Irani, M.: Space-time video completion. In: Proceedings of Conference on Computer Vision and Pattern Regonition'04, Washington, DC, pp. 120–127. IEEE Computer Society (2004)
34. Wilczkowiak, M., Brostow, G.J., Tordoff, B., Cipolla, R.: Hole filling through photomontage. In: Proceedings of British Machine Vision Conference '05, Oxford, United Kingdom, pp. 492–501 (2005)
35. Wu, Q., Yu, Y.: Feature matching and deformation for texture synthesis. ACM Trans. Graph. **23**(3), 362–365 (2004)
36. Zhang, Y.J., Xiao, J.J., Shah, M.: Region completion in a single image. EUROGRAPHICS, Grenoble, France, Short Presentations (2004)
37. Zhang, J., Zhou, K., Velho, L., Guo, B., Shum, H.Y.: Synthesis of progressively variant textures on arbitrary surfaces. ACM Trans. Graph. **22**(3), 295–302 (2003)



JIANBING SHEN is a PhD candidate of the State Key Lab of CAD & CG, Zhejiang University, People's Republic of China. He received his BSc and MSc degrees in mechatronic engineering from Zhejiang University of Technology. His research interests include texture synthesis, image completion, and dynamic texture.

XIAOGANG JIN is a professor of the State Key Lab of CAD & CG, Zhejiang University. He received his BSc degree in computer science in 1989, MSc and PhD degrees in applied mathematics in 1992 and 1995, all from Zhejiang University. His research interests include implicit surface modeling, space deformation, computer animation and realistic image synthesis.

XIAOYANG MAO is an associate professor at the University of Yamanashi in Japan. She received her MS and PhD in computer science from Tokyo University. Her research interests include flow visualization, texture synthesis, non-photo-realistic rendering and human computer interactions.

JIEQING FENG is a professor at the State Key Lab of CAD & CG, Zhejiang University, People's Republic of China. He received his BSc degree in applied mathematics from the National University of Defense Technology in 1992 and his PhD in computer graphics from Zhejiang University in 1997. His research interests include space deformation, computer-aided geometric design and computer animation.