Visual Comput (2007) 23: 631–639
DOI 10.1007/s00371-007-0154-3

ORIGINAL ARTICLE

Jianbing Shen
Xiaogang Jin
Xiaoyang Mao
Jieqing Feng

Deformation-based interactive texture design using energy optimization

Published online: 13 June 2007

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 interactive texture design scheme based on deformation and energy optimization. Given a small sample texture, the design process starts with applying a set of deformation operations to the sample texture to obtain a set of deformed textures. Then local changes to those deformed textures are further made by replacing their local regions with the texture elements interactively selected from other textures. Such a deform–select–replace process is iterated many times until the desired deformed textures are obtained. Finally the deformed textures are composed to form a large texture with graph-cut optimization. By combining the graph-cut algorithm with an energy optimization process, interactive selections of local texture elements are done simply through indicating the positions of texture elements very roughly with a brush tool. Our experimental results demonstrate that the proposed technique can be used for designing a large variety of versatile textures from a single small sample texture, increasing or decreasing the density of texture elements, as well as for synthesizing textures from multiple sources.

Keywords Deformation · Interactive · Texture design · Brushes · Energy optimization

1 Introduction

Textures have been a research focus for many years in human perception, computer graphics and computer vision. Recent decades of research activities in this area emphasize on texture synthesis. Given a sample texture, a texture synthesis algorithm generates a new one bearing the same visual characteristics. In spite of the fact that numerous methods have been proposed for texture synthesis, how to design a variety of large textures from a single small sample texture is still a challenging problem.

Recently, Matusi et al. [18] 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 the 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. Shen et al. [22] proposed a completion-based texture design technique for producing a variety of textures by applying deformations to the extracted layers of texture elements. The main limitation of Shen et al.'s method, however, lies in the fact that it has no interaction with the local property of the resulting texture elements.

In this paper, we present a new deformation-based interactive texture design algorithm. The proposed algorithm has the ability to locally change the visual property of texture elements with little user interaction, and hence drastically broadens the variation of textures that can be synthesized with the existing methods. As shown in Fig. 1, from a single small sample texture, our technique can create a variety of versatile textures, regular or irregular, with increased or decreased density of texture elements. The main contributions of our work consist of the following three aspects:

– A novel framework for designing a large variety of textures by integrating the techniques of 1) texture synthesis, 2) interactive image editing, 3) graph-cut based optimization, and 4) gradient-based Poisson optimization.
– An effective graph-cut and energy optimization-based method for automatically extracting texture elements indicated by the designer.
– A new optimization based algorithm for synthesizing textures from multiple sources.

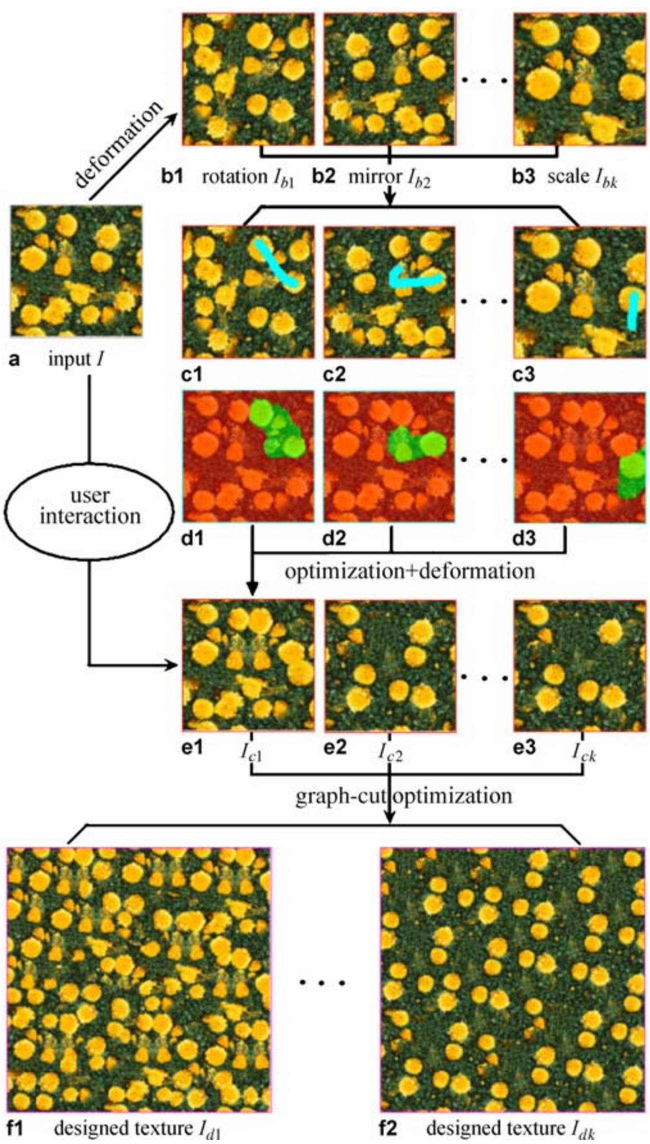

Fig. 1. Our deformation-based interactive texture design algorithm. **a** Small input texture I; **b1**, **b2**, **b3** the initial deformed textures $I_{b1}, I_{b2}, \ldots, I_{bk}$; **c1**, **c2**, **c3** the texture elements regions indicated by the designer's interactive brush; **d1**, **d2**, **d3** the composed result by local deformations using energy optimization; **e1**, **e2**, **e3** the interactive deformed textures $I_{c1}, I_{c2}, \ldots, I_{ck}$; **f1**, **f2** the designed textures $I_{d1}, I_{d2}, \ldots, I_{dk}$

In the rest of the paper, we first introduce the related work on texture synthesis and interactive image manipulation tools in Sect. 2. Then, in Sect. 3, we discuss the details of our deformation and energy optimization-based interactive texture design scheme. The extension of the existing texture deformation algorithm using the completion technique is also described in Sect. 3. The details of the new graph-cut-based energy optimization method are given in Sect. 4, and the method for synthesizing textures from multiple sources using optimization is presented in Sect. 5. After showing the experimental results in Sect. 6, we conclude the paper and show some directions for future work in Sect. 7.

2 Related work

Texture synthesis

There is a long sequence of earlier papers on pixel-based and patch-based texture synthesis, which we can briefly review here. In non-parametric texture synthesis [6, 11], texture is synthesized one pixel (or one patch) at a time by finding pixels (patches) with similar neighborhood to the already synthesized pixels (patches) in the sample texture. The traditional approach is to generate textures sequentially in a scanline order. Improvements include hierarchical synthesis [25], coherent synthesis [9, 19], similarity-based synthesis [4], feature matching and patch deformation synthesis [26], texton revisited synthesis [5], and appearance-space synthesis [15].

A number of authors have tackled the challenge of combining and mixing textures. Efros and Freeman [11], Cohen et al. [6] and Kwatra et al. [14] synthesized a non-uniform texture composed of homogeneous patches. Wei et al. [24] generated mixture of textures from multiple input textures. Liu et al. [17] described a system to analyze and manipulate photographic textures that allows a user to design near regular textures. Similar to the work by Liu et al. [17], Matusik et al. [18] strived to build a comprehensive texture model, then constructed a texture space that spanned the range of textures induced by a database of natural images.

The idea of applying transformations to the patches has also been discussed by Kwatra et al. [14] in their patch-based texture synthesis technique using the graph-cut algorithm. The results are obtained using deformation operations, such as rotation, mirror and scaling. However, as mentioned in their paper, the cost for searching matching patches will increase when the extent of deformation increases. Shen et al. [22] proposed a completion-based texture design algorithm by applying transformations to the extracted texture layers. Their technique can produce a wide variety of textures by making changes to the size, orientation and relative position of texture elements. However, the main limitation of their method lies in its inability to take into consideration the designer's need and creation. Interactions on local texture elements are not allowed for the designers in their method.

Interactive image manipulation tools

Interactive image manipulation and editing packages, such as Adobe Photoshop, are commonly utilized by digital photographers. In their workflow [21], images are manipulated directly and immediate visual feedback is provided.

Recently, many researchers proposed several interactive digital image editing tools by using region-based methods, e.g., the magic wand in Photoshop [21], intelligent paint [2], interactive graph-cut image segmentation [3], lazy snapping [16], and interactive image photomontage [1].

Our work is most closely related to the method of interactive digital montage [1], where users use brushes to indicate which parts of a set of photographs should be combined into a composite result. Similarly, our method also uses the strokes to define constraints for designing a variety of deformed textures. By allowing the user to interact with local texture elements, our technique can provide local changes to the size, orientation and relative position of texture elements according to the texture designer's need and creation. Moreover, our proposed algorithm has the ability to increase or decrease the density of texture elements interactively, which is suitable for designing a variety of versatile textures from a single small sample texture.

3 Our approach

3.1 Algorithm overview

The goal of our algorithm is to enable the texture designer to easily create a deformed texture in a spatially varying manner, along with several common types of deformation operations (rotation, translation, mirror, scale and flip).

Our proposed workflow is summarized as follows:

1. Load a small sample texture image I.
2. Apply deformation operations (rotation, translation, mirror, scale and flip) to produce a set of small deformed textures $I_{b1}, I_{b2}, \dots, I_{bk}$. The range of rotation, translation and scale is interactively controlled by the designer.
3. Make local changes to the deformed textures by copying local texture elements from one to the other.
4. Design large textures from the deformed textures obtained in Step (3) by using the graph-cut optimization algorithm. Apply further deformation operations if necessary.
5. Repeat Steps (2) to (4) until a satisfactory set of textures $I_{d1}, I_{d2}, \dots, I_{dk}$ is obtained, combining the texture deformation algorithm described in Sect. 3.3.

This workflow is illustrated by the sequence of images in Fig. 1. Given an input sample texture (Fig. 1a), a set of deformed textures are produced (Fig. 1b1,b2,b3) after applying deformation operations. Then the user uses brushes to paint some texture elements interactively (Fig. 1c1,c2,c3) and the corresponding regions of those texture elements are automatically calculated (Fig. 1d1,d2,d3). These texture elements are stitched into other textures with the gradient-based Poisson optimization [1, 20, 22], in order to obtain the textures with varying local properties (Fig. 1e1,e2,e3). Finally, by applying the texture deformation algorithm described in Sect. 3.3 to the deformed textures, large textures are designed (Fig. 1f1,f2).

3.2 Interactive local texture deformation

In order to make the above workflow effective, several requirements should be met, such as quickly generated previews of the overall result, a simple, intuitive and easy to use mechanism for performing the local deformation, and an undo function allowing the user to modify previously specified adjustments. Our prototype implementation is based on the interactive digital photomontage technique [1] and supports several types of brushes that can be used to set constraints for the texture's local deformations. Similarly to [1], the designer uses the most frequently used single-texture brushes.

At Step (3), the local deformation of a texture is realized by replacing its local regions with the texture elements from another deformed texture. We call the texture to be locally deformed the *base texture* I_{base} ($I_{\text{base}} \in \{I, I_{b1}, I_{b2}, \dots, I_{bk}\}$) and the texture providing the texture elements the *reference texture* I_{ref} ($I_{\text{ref}} \in \{\{I, I_{b1}, I_{b2}, \dots, I_{bk}\} - I_{\text{base}}\}$). As shown in Fig. 1, the user does not need to precisely specify the region including the texture elements in I_{ref}. Instead, the designer uses the brush to roughly paint the texture elements ("yellow flowers") in I_{ref}. The corresponding region including the texture elements is calculated automatically with the graph-cut-based energy optimization technique. The obtained texture elements are then embedded into to the base texture I_{base} seamlessly by the gradient-based Poisson optimization method [1, 22]. Such local deformations are repeated several times, while at each step the user is allowed to choose new texture elements by painting new strokes according to his creation. The resulting base texture is further refined by the texture deformation algorithm using completion and then is used as the reference texture for another base texture. The descriptions of the texture deformation algorithm using completion and the graph-cut-based energy optimization technique can be found in Sects. 3.3 and 4, respectively.

3.3 Texture deformation using completion

The last step of our texture design workflow employs the texture deformation algorithm using the completion technique [8, 22, 23], which is based on the method proposed in [22]. We refer the readers to [22] for a detailed description of their completion-based texture design method. The texture deformation algorithm using the completion technique is summarized as follows:

– Input: single sample texture I.
– Step 1: Layering, extracting texture layers using existing color image segmentation techniques [7, 12].
– Step 2: Deformation, applying chaotic-based deformation operations (such as rotation, translation, mirror, flip and scale) to the texture layers.
– Step 3: Example-based image completion, inpainting the hole regions induced by deformation with the graph-cut algorithm.
– Step 4: Smoothing, removing the visual artifacts produced by Step 3 through the gradient-based Poisson optimization [1].
– Output: deformed textures $I_1, I_2, \ldots, I_k$.

In order to increase the versatility of the deformed textures, we add a new flip operation to the set of deformation operations provided by [22]. Moreover, we extend it with more robust chaotic maps [13] beyond the basic logistic map. The experimental results demonstrate that our technique can generate a wide variety of large deformed textures with a good stochastic property.

4 Interactive design using energy optimization

Boykov et al. [3] have developed several techniques that use the graph-cut algorithm for optimizing pixel labeling. Some early vision problems, such as image restoration, can be modeled as an image labeling problem which is to find a labeling f that assigns each pixel p a label f_p, so that f is both piecewise smooth and consistent with the target data. Such a labeling f can be obtained as the result of minimizing the following energy:

$$E(f) = E_{\text{smooth}}(f) + E_{\text{data}}(f)\,, \tag{1}$$

where E_{smooth} measures the extent to which f is not piecewise smooth, while E_{data} measures the disagreement between f and the objective data. Boykov et al. [3] proposed an algorithm to find f through an iterative process. At each step, the graph-cut algorithm is used to find out the swapping between two labels α and β (α-β swap) or the assigning of a given label (α-expansion) while decreasing the energy $E(f)$ from that of the previous step. The labeling computation is guaranteed to be within a factor of two of the global minimum when the cost function is a metric.

Agarwala et al. [1] used Boykov's graph-cut optimization algorithm for their interactive digital photomontage application, where a new cost function is used to guide the optimization process resulting a smooth composition of source images.

We further extend Agarwala's work by employing the energy optimization for texture montage. Suppose that we have obtained k deformed textures $I_{b1}, I_{b2}, \ldots, I_{bk}$ after applying the deformation operations in the first step. We want to make local changes to some of those textures by replacing their local regions with the texture elements from remaining textures. As shown in Fig. 2, the user starts with selecting a base texture I_{base} (Fig. 2a, the texture to be locally changed) and the reference texture I_{ref} (Fig. 2b, the texture providing the texture elements). After the user indicates the texture elements in I_{ref} using brushes (Fig. 2c), a subimage I_s ($I_s \subset I_{\text{ref}}$) enclosing the brush stroke is clipped out from I_{ref}. In order to produce the locally deformed texture (Fig. 2f), we use the graph-cut based energy optimization algorithm to compute the label of pixels in the composite texture (Fig. 2d) and find the best path (Fig. 2e) to smoothly stitch I_s with I_{base}. The labeling of the pixels in the composite texture is a mapping of the pixels between the base texture I_{base} and the clipped reference texture I_s. We denote the label for each pixel as $L(p)$, it is certain that a seam (Fig. 2e) exists between two neighboring pixels (p, q) in the output if $L(p) \neq L(q)$.

In [1, 3], the energy function E for the labeling L of an image is defined as follows:

$$E(L) = E_{\text{data}}(L) + \lambda \cdot E_{\text{smooth}}(L), \tag{2}$$

$$E_{\text{data}}(L) = \sum_p E_d(p, L(p)), \tag{3}$$

$$E_{\text{smooth}}(L) = \sum_{p,q} E_s(p, q, L(p), L(q)), \tag{4}$$

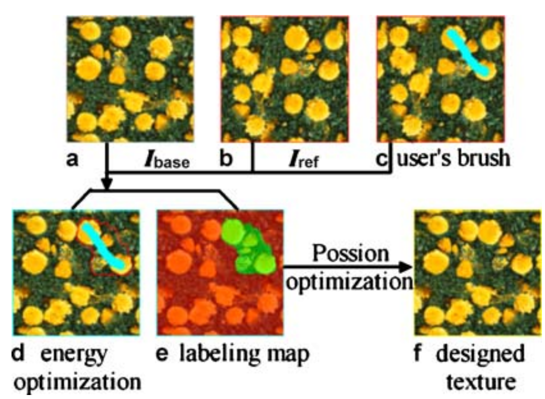

Fig. 2a–f. Illustration of our interactive design using energy optimization. **a** The base texture I_{base}; **b** the reference texture I_{ref}; **c** the position of the user's brush; **d** the cut region (with *red* boundary) including texture elements using our energy optimization method; **e** its corresponding labeling map; **f** the designed texture

where the first term is defined by the distance to the image objective while the second term is defined by the distance to the seam objective. Since we want to replace the local region of the base texture with the specified texture elements in the reference texture, the image objective here is I_s. Therefore $E_{\text{data}}(L)$ is computed as follows:

$$E_{\text{data}}(L) = \begin{cases} 0, & \text{if } L(p) = I_s \\ v, & \text{if } L(p) \neq I_s \end{cases}, \tag{5}$$

where v is a user specified large value.

Since the labeling is a mapping to either I_{base} or I_s , the second term is defined by the distance between the pixels of I_{base} and I_s, that is,

$$E_s(p, q, L(p), L(q)) = 0, \quad \text{if } L(p) = L(q). \tag{6}$$

Otherwise, the energy is computed in the same way as [1]:

$$E_s(p,q, L(p), L(q)) = \begin{cases} M_x, & \text{if "colors"} \\ M_y, & \text{if "gradients"} \\ 0.5(M_x + M_y) & \text{if "colors + gradients"} \end{cases}, \tag{7}$$

where

$$M_x = \|C_{L(p)}(p) - C_{L(q)}(p)\| + \|C_{L(p)}(q) - C_{L(q)}(q)\|,$$
$$M_y = \|\nabla G_{L(p)}(p) - \nabla G_{L(q)}(p)\| + \|\nabla G_{L(p)}(q) - \nabla G_{L(q)}(q)\|,$$

and $\nabla G(p)$ is a six-component color gradient (in R, G, B) at pixel p.

The algorithm terminates when a pass over all labels fails to reduce the cost function. Kwatra et al. [14] and Agarwala et al. [1] have successfully used the 'alpha expansion' with this interaction penalty. In our case, we have also found that it is good enough to produce satisfactory composite textures (Fig. 2f).

5 Texture design from multiple sources using optimization

Our texture design method from multiple sources using optimization is extended from [24], but differs in that we perform patch-based synthesis via optimization while theirs is based on pixel-by-pixel mixture. The goal of multi-source texture design is to synthesize new textures that capture the combined characteristics of several input textures. For example, given four flower and grass textures (Fig. 6a), a set of new textures can be generated with a hybrid appearance (Fig. 6b–l).

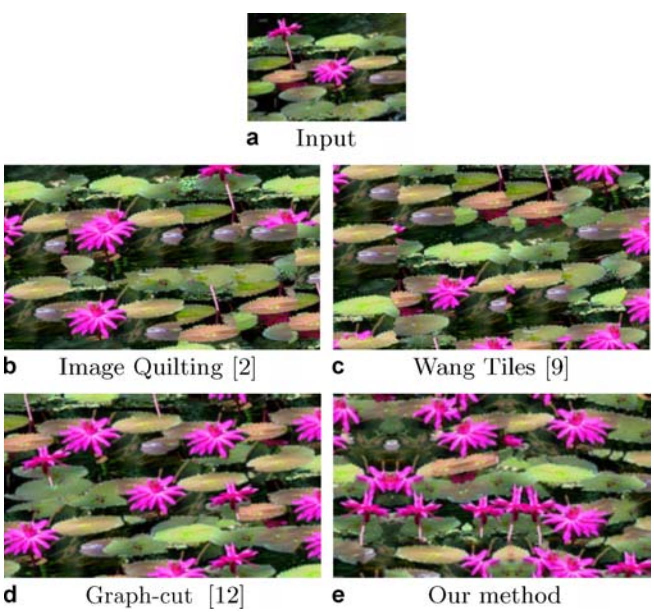

Fig. 3. Comparison of our deformation-based algorithm with image quilting [11], Wang tiles [6] and graph-cut [14]

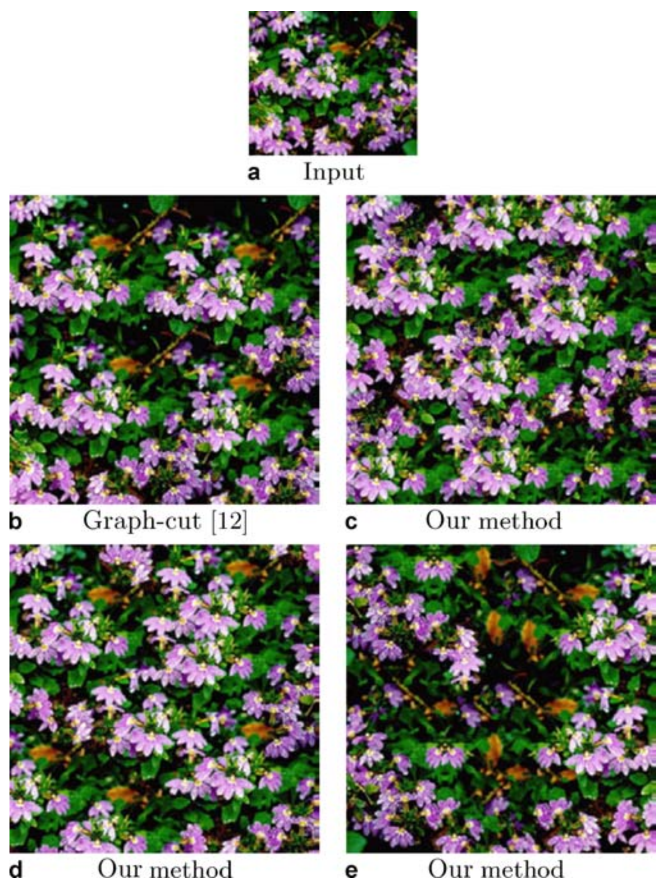

Fig. 4. Comparison of our deformation-based algorithm with graph-cut [14]

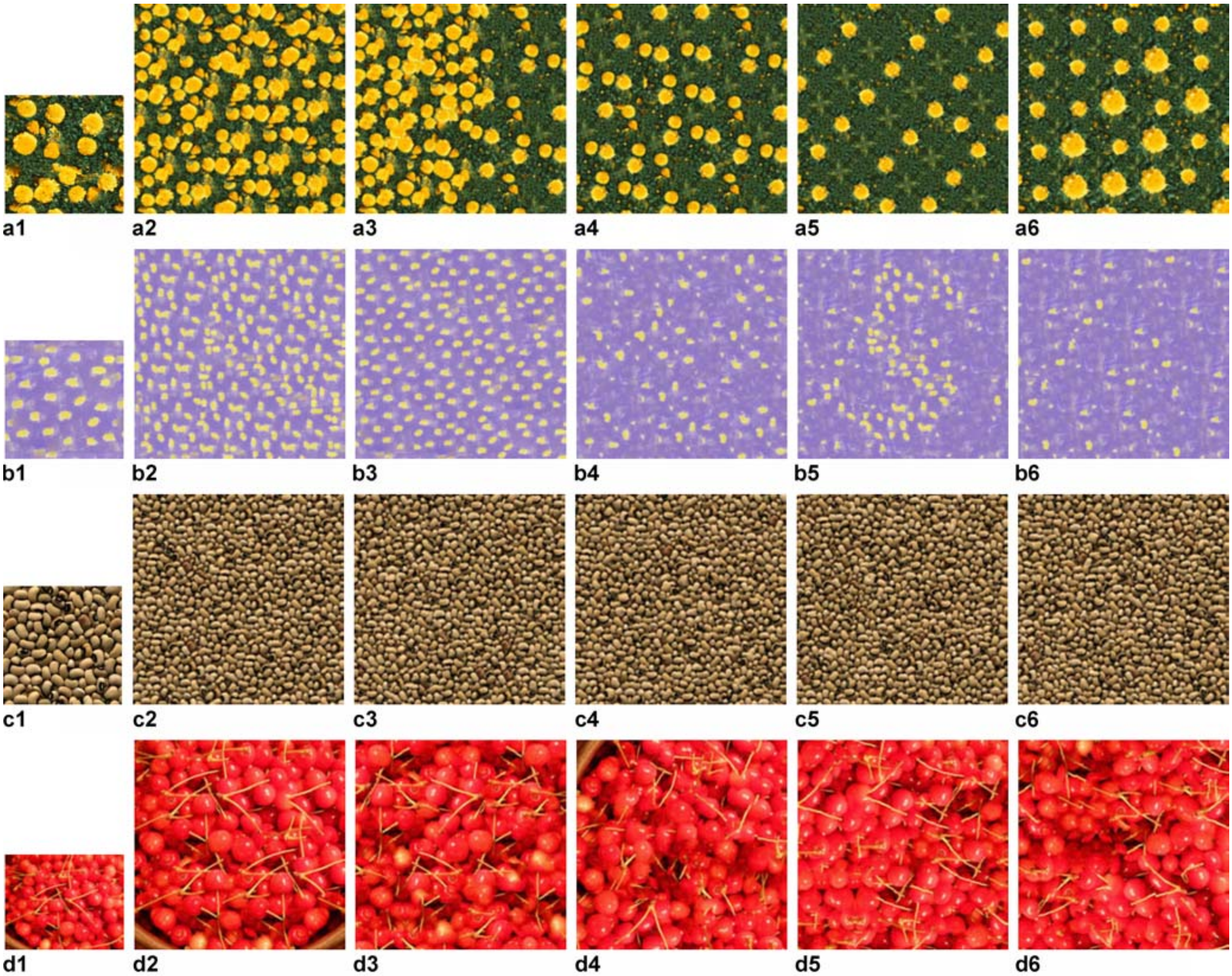

Fig. 5. Examples of spatially varying designed textures using our deformation-based method. *Left columns* (**a1**, **b1**, **c1**, **d1**) are the input textures, the others are the deformed textures. Texture size: input: 268×230; output: 360×360

The details of the proposed texture design algorithm from multiple sources via optimization are described as follows:

- Input: multiple texture sources $\{I_1, I_2, \ldots, I_k\}$.
- Step 1: Each source texture is divided into l patches, and the source textures are represented by the patch sets:

$$I_1 = \left\{P_1^{I_1}, P_2^{I_1}, \ldots, P_l^{I_1}\right\},\ I_2 = \left\{P_1^{I_2}, P_2^{I_2}, \ldots, P_l^{I_2}\right\}, \ldots, I_k = \left\{P_1^{I_k}, P_2^{I_k}, \ldots, P_l^{I_k}\right\}.$$

- Step 2: Randomly select an initial patch $P_l^{I_k}$, paste it to the left top corner of the output texture I_{mk}, then find the best matched neighborhood patch P_i constrained through optimizing the following function:

$$\min \sum_{\left(P_l^{I_k}, P_i\right)} w_i \cdot \left(\left\| P_l^{I_k} - P_i \right\| + \left\| N_i\left(P_l^{I_k}\right) - N_i(P_i) \right\|\right), \tag{8}$$

where index i runs through all the input textures, $N_i(P_l^{I_k})$, $N_i(P_i)$ are the neighborhoods of $P_l^{I_k}$ and P_i, respectively, and w_i are the weights specified by the relative importance of the input sources.
- Step 3: Copy the best matched patch P_i to the output texture I_{mk}. Apply the graph-cut algorithm [3] to obtain a minimum-error-cut seam in the overlapped region between P_i and $P_l^{I_k}$.
- Step 4: Run Steps 2 to 3 iteratively until the whole output texture I_{mk} is synthesized. The texture deformation

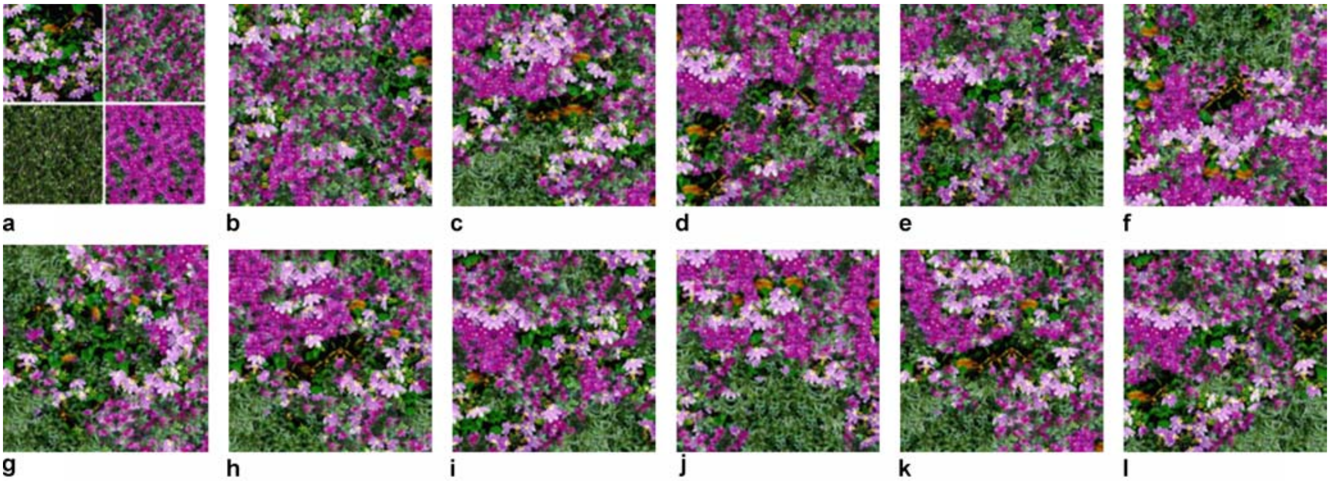

Fig. 6. Designed textures from multi-source textures using our optimization algorithm. Texture size: input: 230×230; output: 360×360

using the completion technique described in Sect. 3.3 is then employed for further designing the deformed textures.

– Output: the final designed textures $\{I_{m1}, I_{m2}, \ldots, I_{mk}\}$.

6 Experimental results and discussions

Our algorithm has been applied to a variety of sample texture images. In our experiments, most of the source texture images are downloaded from the websites[1]. For comparison, we use those sample textures used by existing texture synthesis work [6, 11, 14]. All the experiments shown in this section were run on a PC with Pentium IV 1.6 GHz CPU + 512 MB RAM.

In Fig. 3, we compare our approach with other existing techniques. The result for graphcut was taken from [14], while the other two were generated by our implementation. The texture size is 268×230 for Fig. 3a, and 360×360 for Fig. 3(b,c,d,e). The patch size is selected as 64×64. From the images we find that the quality of the texture generated with our approach is superior to that of image quilting [11] and Wang tiles [6] and is comparable to the result produced by graphcut [14]. The sample texture in Fig. 3a consists of only two different lotus flowers. The techniques that simply use the original patches selected from the sample texture can lead to the repetition of those texture elements in the resulting large texture. As shown in Fig. 3d, all the flowers have the same shape and orientation as either of the two flowers in the sample texture. However, our interactive technique can create the texture consisting of the flowers of different shape, size and orientation, which is demonstrated in Fig. 3e. The density of the lotus flowers in our results can be increased or decreased at the desired position according to the user's need.

Figure 4 is another example demonstrating the effectiveness of our method, compared with the Graph-cut [14] method. As shown in Figs. 4c–e, the density of texture elements (flowers) is increased (Fig. 4c and d) or decreased (Fig. 4e).

Figure 5 gives more examples demonstrating the capability of our technique for creating a large variety of textures from a small sample, while maintaining the continuity of texture features as well as the shapes of individual texture elements. Our method changes the density of texture elements (yellow flowers) interactively according to the designer's need. In Fig. 5a2–a6, the density of the texture elements decrease gradually. We can also interactively make the left part and the right part of the designed texture with different density (Fig. 5a3), create new texture elements (Fig. 5a5), locally enlarge the size of texture elements (Fig. 5a6), design regular (Fig. 5a5,a6) or irregular (Fig. 5a2–a4) texture patterns, and make the shape of designed texture look like a large 'S' shape (Fig. 5b5).

Figure 6 demonstrates another interesting application of our technique. Therein textures are synthesized from multiple source textures using our optimization method. In Fig. 6, four input textures of size 230×230 are used to interactively create a variety of designed textures of size 360×360 (Fig. 6b–l).

7 Conclusions and future work

A novel deformation-based interactive texture design method using energy optimization has been proposed in this paper. Experimental results demonstrate both the feasibility and the effectiveness of our algorithm.

[1] http://www.cc.gatech.edu/cpl/projects/graphcuttextures
http://people.csail.mit.edu/wojciech/TextureDesign/index.html

The main advantage of our algorithm over most existing texture synthesis methods lies in its capability to create a wide variety of very natural textures interactively, from only a single small sample texture, according to the texture designer's need and creation. By applying the extended graph-cut-based energy optimization approach and the completion-based texture deformation method, we have designed textures with good stochastic property. Our experimental results also demonstrate that the proposed technique can be applied to other applications such as texture synthesis from multiple sources.

Although the deformation operations used in our method can produce good results, it would be meaningful to develop more sophisticated and powerful deformation tools in the future. Another potential extension of our method would be its application in dynamic texture design [10] where the consistency of the deformed textures between adjacent frames should be considered.

Acknowledgement This work was supported by the China 973 program (Grant No. 2002CB312101), the China 863 program (Grant No. 2006AA-01Z314), the National Natural Science Foundation of China (Grant No. 60573153), the Natural Science Foundation of Zhejiang Province (Grant No. R105431), the Program for New Century Excellent Talents in University (Grant No. NCET-05-0519), and the Ministry of Education, Science, Sports and Culture, goverment of Japan, Grant-in-Aid for Scientific Research (B) (Grant No.18300030, 2006). The authors would like to thank Vivek Kwatra for making the Graphcut texture synthesis results available.

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. Barrett, W.A., Cheney, A.S.: Object-based image editing. ACM Trans. Graph. **21**(3), 777–784 (2002)
3. Boykov, Y., Kolmogorov, V.: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. IEEE Trans. Patt. Anal. Mach. Intel. **26**(9), 1124–1137 (2004)
4. Brooks, S., Dodgson, N.: Self-similarity based texture editing. ACM Trans. Graph. **21**(3), 653–656 (2002)
5. Charalampidis, D.: Texture synthesis: textons revisited. IEEE Trans. Image Process. **15**(3), 777–787 (2006)
6. Cohen, M.F., Shade, J., Hiller, S., Deussen, O.: Wang tiles for image and texture generation. ACM Trans. Graph. **22**(3), 287–294 (2003)
7. Comaniciu, D., Meer, P.: Mean shift: A robust approach towards feature space analysis. IEEE Trans. Patt. Anal. Mach. Intel. **24**(5), 603–619 (2002)
8. 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)
9. Dischler, J.-M., Maritaud, K., Lévy, B., Ghazanfarpour, D.: Texture particles. Comput. Graph. Forum. **21**(3), 401–410 (2002)
10. Doretto, G., Chiuso, A., Wu, Y., Soatto, S.: Dynamic textures. Int. J. Comput. Vision **51**(2), 91–109 (2003)
11. Efros, A.A., Freeman, W.T.: Image quilting for texture synthesis and transfer. In: Proceedings of SIGGRAPH '01, pp. 341–346. Los Angeles, ACM, New York (2001)
12. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. Int. J. Comput. Vision **59**(2), 167–181 (2004)
13. Jakimoski, G., Kocarev, L.: Chaos and cryptography: block encryption ciphers based on chaotic maps. IEEE Trans. Circ. Syst. – 1: Fund. Theory Appl. **48**(2), 163–169 (2001)
14. 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)
15. Lefebvre, S., Hoppe, H.: Appearance-space texture synthesis. ACM Trans. Graph. **25**(3), 541–548 (2006)
16. Li, Y., Sun, J., Tang, C.K., Shum, H.Y.: Lazy snapping. ACM Trans. Graph. **23**(3), 303–308 (2004)
17. Liu, Y., Lin, W.C., Hays, J.H.: Near regular texture analysis and manipulation. ACM Trans. Graphics. **23**(3), 368–376 (2004)
18. Matusik, W., Zwicker, M., Durand, F.: Texture design using a simplicial complex of morphable textures. ACM Trans. Graph. **24**(3), 787–794 (2005)
19. Nicoll, A., Meseth, J., Müller, G., Klein, R.: Fractional Fourier texture masks: guiding near-regular texture synthesis. Comput. Graph. Forum **24**(3), 569–579 (2005)
20. Pérez, P., Gangnet, M., Blake, A.: Poisson image editing. ACM Trans. Graph. **22**(3), 313–318 (2003)
21. Reichmann, M.: An image processing workflow. http://luminouslandscape.com/tutorials/workflow1.shtml
22. Shen, J.B., Jin, X.G., Mao, X.Y., Feng, J.Q.: Completion based texture design using deformation. Visual Comput. **22**(9), 936–945 (2006)
23. Shen, J.B., Jin, X.G., Zhou, C., Wang, C.C.L.: Gradient based image completion by solving the Poisson equation. Comput. Graph. **31**(1), 119–126 (2007)
24. Wei, L.Y.: Texture synthesis from multiple sources. Proceedings of the SIGGRAPH 2003 Conference on Sketches & Applications. ACM, New York (2003)
25. Wei, L.Y., Levoy, M.: Fast texture synthesis using treestructured vector quantization. In: Proceedings of SIGGRAPH '00, pp. 479–488. New Orleans, ACM, New York (2000)
26. Wu, Q., Yu, Y.: Feature matching and deformation for texture synthesis. ACM Trans. Graph. **23**(3), 362–365 (2004)




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 high dynamic range imaging and processing.

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 current research interests include implicit surface computing, special effects simulation, mesh fusion, texture synthesis, crowd animation, cloth animation and facial animation.

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