Pattern Recognition 43 (2010) 457–469

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

A unified framework for designing textures using energy optimization

Jianbing Shen[a,c,*], Hanqiu Sun[b], Jiaya Jia[b], Hanli Zhao[c], Xiaogang Jin[c], Shiaofen Fang[d]

[a]School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China
[b]Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
[c]State Key Lab of CAD & CG, Zhejiang University, Hangzhou 310027, China
[d]Department of Computer and Information Science, Indiana University-Purdue University Indianapolis, Indianapolis, IN 46202, USA

ARTICLE INFO

Article history:
Received 31 May 2008
Received in revised form 6 January 2009
Accepted 2 March 2009

Keywords:
Interactive texture design
Daubechies wavelet
Energy optimization

ABSTRACT

A unified framework is proposed for designing textures using energy optimization and deformation. Our interactive scheme has the ability to globally change the visual properties of texture elements, and locally change texture elements with little user interaction. Given a small sample texture, the design process starts with applying a set of global deformation operations (rotation, translation, mirror, scale and flip) to the sample texture to obtain a set of deformed textures automatically. Then we further make the local deformation to the deformed textures interactively by replacing the local-texture elements regions from other textures. By utilizing the energy optimization method, interactive selections and deformations of local-texture elements are accomplished simply through indicating the positions of texture elements very roughly with a brush tool. Finally the deformed textures are further utilized to create large textures with the fast layer-based texture deformation algorithm, and the wavelet-based energy optimization. Our experimental results demonstrate that the proposed approach can help design a large variety of textures from a small example, change the locations of texture elements, increase or decrease the density of texture elements, and design cyclic marbling textures.

1. Introduction

Texture analysis and synthesis has a long history in human perception, psychology and computer vision, and has also been widely recognized as an important research topic in computer graphics. In the past decade, many texture synthesis approaches have been proposed. Neighborhood-based texture synthesis methods [3,6,8,14,15,20,33,36], especially patch-based techniques [3,6,14,15], have achieved significant progresses. Nevertheless, how to design a variety of deformed textures from similar texture patterns is still a challenging problem, though some attempts have been made [24].

The idea of applying transformations to patches has been discussed by Kwatra et al. [15] in their patch-based texture synthesis technique using the Graph-cut algorithm. The results are generated using deformation operations, such as rotation, mirror and scaling. But as mentioned in their paper, the cost for searching matching patches will increase when the extent of deformation increases. Matusi et al. [24] has developed a system for designing novel textures from an input database. However, their morphable texture interpolation is based on a single one-to-one warping between the pairs of texture samples, which is too restrictive for textures with highly irregular structures and may cause discontinuous mappings of the patches to the original texture.

In this paper, we present a unified framework for designing textures using energy optimization. The proposed scheme considers both the global and local deformations of the texture elements, and consists of the following stages: (1) global deformation (rotation, translation, mirror, scale and flip); (2) interactive local deformation using energy optimization; (3) texture design using wavelet-based optimization; (4) fast layer-based texture deformation algorithm. Our interactive texture design scheme has the ability to globally change the visual property of texture elements, and locally change texture elements with easy user interaction. Hence, our method drastically broadens the variation of deformed textures from an input example. As shown in Fig. 1, with only a single small exemplar texture, our technique can create a variety of textures results. The density of texture elements ("purple flowers") could be decreased (Fig. 1(e)) or increased (Fig. 1(h)–(j)), while the existing Graph-cut texture synthesis [15] result (Fig. 1(b)) does not have such ability.

Ourtexture design scheme has the following new features:

- An interactive local-texture design approach using energy optimization.
- A fast layer-based texture design algorithm using energy optimization and deformation.

* Corresponding author at: School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China. Tel./fax: +86 1068468051.
E-mail address: shenjianbing@bit.edu.cn (J. Shen).

0031-3203/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2009.03.003

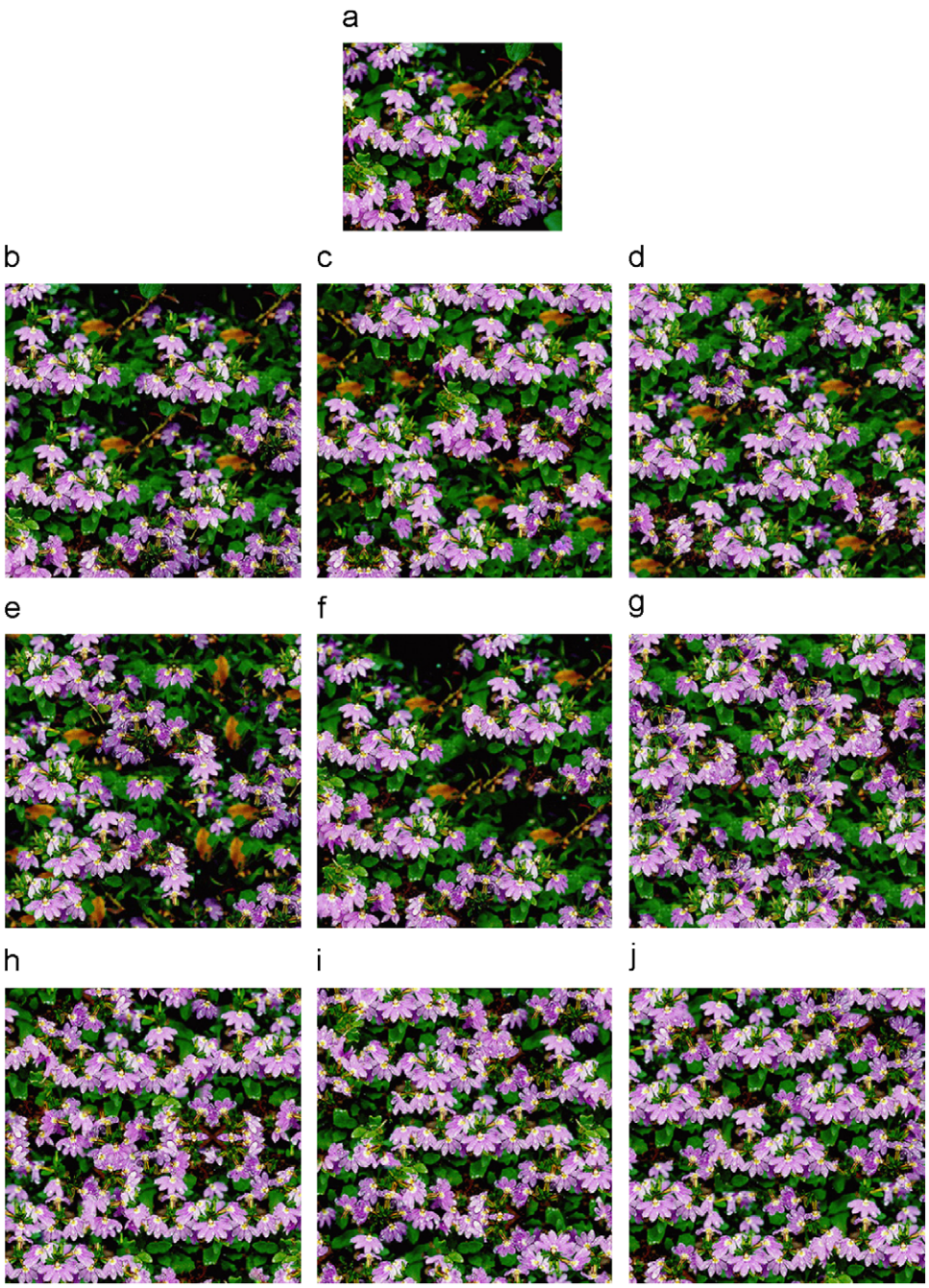

Fig. 1. Illustration of our interactive texture design scheme using energy optimization. (a) The input sample texture; (b) the result by Graph-cut [15]; and (c)–(j) the designed texture by our method.

- A new energy optimization method for designing textures using Daubechies wavelet.
- A unified framework for designing textures by integrating the techniques of (1) interactive image editing, (2) energy optimization, and (3) Daubechies wavelet-based energy optimization.

2. Related work

2.1. Texture synthesis

Numerous methods have been proposed for texture synthesis. Most of the recent methods generate new textures possessing the same statistical and visual characteristics as the input one. These approaches can be either pixel-based or patch-based. They include hierarchical synthesis [3], coherent synthesis [8], spatially variant texture synthesis [16], real-time texture synthesis [14], feature matching and patch deformation synthesis [20], texton revisited synthesis [32], appearance-space synthesis [33], and constrained texture synthesis [36].

The pixel-based texture synthesis approach generates a synthesized image pixel by pixel, while the patch-based one creates a new texture by copying patches from sample textures and pasting them together in a consistent way. The advantage of patch-based one is that the texture structure inside the patches is maintained. Hertzman et al. [5], Efros and Freeman [6], Cohen et al. [13] and Kwatra et al. [15] synthesized a non-uniform texture composed of

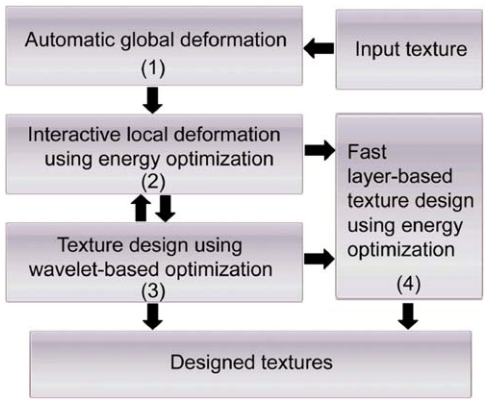

Fig. 2. The unified framework for designing textures using energy optimization. Input a sample texture I; (1) the global deformed textures $\{I_{g1}, I_{g2}, \ldots, I_{gk}\}$; (2) the local deformed textures $\{I_{l1}, I_{l2}, \ldots, I_{lk}\}$ using energy optimization, by the designer's interactive brush to indicate texture elements regions; (3) the designed large textures $S_1 = \{I_{w1}, I_{w2}, \ldots, I_{wk}\}$ by wavelet-based energy optimization; and (4) the designed large texture set $S_2 = \{I_{d1}, I_{d2}, \ldots, I_{dk}\}$ by the fast layer-based energy optimization method.

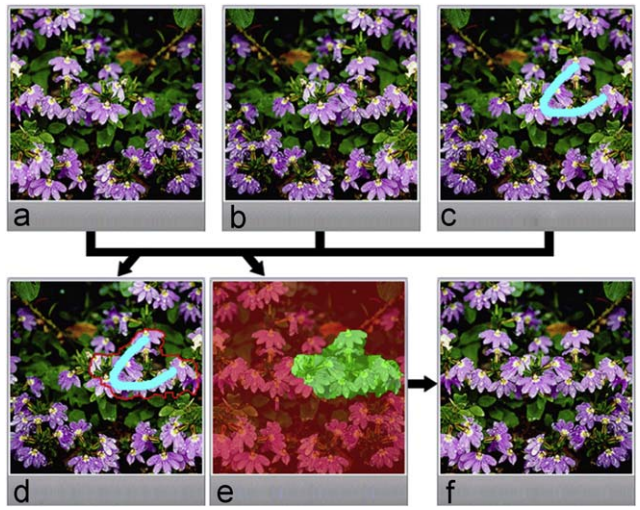

Fig. 3. Illustration of our interactive texture design using energy optimization. (a) The base texture I_{base}; (b) the reference texture I_{ref}; (c) the position of the user's brush; (d) the cut region (with red boundary) including texture elements using our energy optimization method; (e) its corresponding labeling map; and (f) the designed texture. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

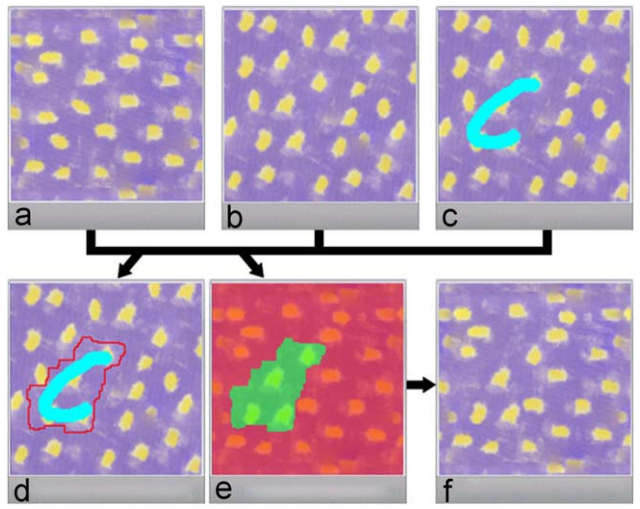

Fig. 4. Another illustration of our interactive texture design scheme using energy optimization. Note that our approach creates the fabric textons with different orientations in the designed texture.

homogeneous patches. Brooks and Dodgson [9] presented an interactive texture editing method that utilizes the self-similarity to replicate intended operations globally over an image. Liu et al. [22] described a system to analyze and manipulate photographic textures which allowed a user to design novel textures. Matusik et al. [24] strived to build a comprehensive texture model. They constructed a texture space that spanned the range of textures induced by a database of natural images. Shen et al. [29] proposed a completion-based texture design technique for producing a variety of textures. The main limitation of Shen et al.'s method, lies in its lack of interaction over the local property of the resulting texture elements.

2.1.1. Image editing using energy optimization

The use for image editing through energy optimization is very prevalent. Agarwala et al. [19] presented an interactive digital photomontage system that finds perfect seams by energy optimization, to combine parts of a set of photographs into a composite image. Kwatra et al. [27] developed an energy optimization method for example-based texture synthesis. Their method formulates the synthesis problem as minimization of an energy function, which is optimized using Expectation Maximization (EM)-like algorithm. Rother et al. [30] developed the AutoCollage technique that automatically creates a collage image from a collection of images. Optimizing the energy function has been done in a variety of ways, including dynamic programming (DP) [6,13], Graph-cuts [21,19], EM-like algorithm [27], or drag-and-drop pasting technique [31].

3. Our approach

3.1. Algorithm overview

Our interactive scheme allows the texture designer to easily create a large variety of versatile textures using energy optimization only from a small input sample texture. The design stages include both the global texture deformation operations (rotation, translation, mirror, scale and flip), and the interactive local-texture elements deformation indicated by the designer's brush.

Our proposed design scheme (Fig. 2) is summarized as below:

- Input: a small sample texture image I.
- Stage (1): apply the global deformation operations (rotation, translation, mirror, scale and flip) to produce a set of small deformed textures $\{I_{g1}, I_{g2}, \ldots, I_{gk}\}$. The ranges of rotation, translation, scale and flip are initialized and controlled interactively by the designer.
- Stage (2): obtain the local deformed textures $\{I_{l1}, I_{l2}, \ldots, I_{lk}\}$ by the designer's interactive brush, indicating texture elements regions using energy optimization.
- Stage (3): design the large texture sets $S_1 = \{I_{w1}, I_{w2}, \ldots, I_{wk}\}$ from the deformed textures obtained in stage (2), by using the wavelet-based energy optimization algorithm. Local deformation operations in stage (2) are further applied if it is necessary.
- Stage (4): employ the fast layer-based texture deformation algorithm to design the deformed texture set $S_2 = \{I_{d1}, I_{d2}, \ldots, I_{dk}\}$.
- Output: the final designed texture set, $S = \{S_1 \cup S_2\}$, consists of the deformed texture sets in stages (3) and (4); the stages (2)–(4) are repeated several times until satisfactory textures are created.

The texture design scheme is illustrated by the sequence of description in Fig. 2. Given an input sample texture, a set of global

deformed textures are automatically obtained after applying deformation operations (rotation, translation, mirror, scale and flip) in stage (1). In stage (2), the designer is required to paint some texture elements interactively by brushes, and the local deformed textures are produced by employing the interactive texture deformation algorithm described in Section 3.2. Then in stage (3), the large designed textures are synthesized from the deformed textures obtained in stage (2), by using the wavelet-based EM optimization algorithm described in Section 3.3. In order to design more deformed textures with varying global and local properties, in stage (4), the fast layer-based texture design method using energy optimization is employed. Finally, the designed texture set is achieved by repeating the stages (2)–(4) several times, until a satisfactory texture set is created.

3.2. Interactive local-texture deformation using energy optimization

Many early vision problems, such as stereo matching or image restoration, are modeled as an image labeling problem which is to find a labeling that assigns each pixel a label. The labels usually represent some local quantity such as disparity. Naturally, such pixel-labeling problems can be represented in terms of energy minimization. There are several powerful energy optimization approaches, which use the Graph-cut algorithm for optimizing pixel labeling, have been developed in [21]. Agarwala et al. [19] used the Graph-cut optimization algorithm [21] for their interactive digital photomontage application, where a new cost energy function is used to guide the optimization process resulting a smooth composition of source images. In [19,21], the energy function E for assigning the labeling L of an image is defined as follows:

$$E(L) = E_d(L(p)) + \lambda E_s(L(p)) = \sum_p E_d(p, L(p)) + \lambda \sum_{p,q} E_s(p, q, L(p), L(q)) \quad (1)$$

where the first term is the data energy, which is defined by the distance to the image objective. While the second term is smoothness energy, which is defined by the distance to the seam objective.

In stage (2), the local deformation of a texture is realized by replacing its local regions with the texture elements from another deformed texture using energy optimization. Our prototype implementation is based on the interactive digital photomontage technique [19], and we further extend Agarwala's and Kwatra's [15] work by employing the energy optimization for texture montage. After we have obtained the deformed textures $I_{g1}, I_{g2}, \ldots, I_{gk}$ by applying the global deformation operations (rotation, translation, mirror, scale and flip) in the stage (1). We call the texture to be locally deformed is the *base texture* $I_{base}(I_{base} \in \{I, I_{g1}, I_{g2}, \ldots, I_{gk}\})$, and the texture providing the texture elements the *reference texture* $I_{ref}(I_{ref} \in \{\{I, I_{g1}, I_{g2}, \ldots, I_{gk}\} - I_{base}\})$. The task of local-texture deformation is to make local changes to some of those textures by replacing their local regions with the texture elements from remaining textures. In this process, the user is not required to precisely indicate the texture elements regions. Instead, the designer uses the brush to roughly paint the texture elements in I_{ref}. The corresponding region including the texture elements is calculated automatically with the energy optimization technique. The obtained texture elements by the energy optimization method are then embedded into the base texture I_{base} seamlessly by the gradient-based Poisson optimization method [19].

As shown in Fig. 3, the user starts with selecting a base texture I_{base} (Fig. 3(a)), the texture to be locally changed) and the reference texture I_{ref} (Fig. 3(b)), the texture providing the texture elements). After the user indicates the texture elements ("purple flowers") in I_{ref} using brushes (Fig. 3(c)), a sub-image I_s ($I_s \subset I_{ref}$) enclosing the brush stroke is clipped out from I_{ref}. In order to produce the locally deformed texture (Fig. 3(f)), we employ the energy optimization algorithm to compute the label of pixels in the composite texture (Fig. 3(d)) and find the best seam (Fig. 3(e)) 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. 3(e)) exists between two neighboring pixels (p, q) in the output if $L(p) \neq L(q)$. Fig. 4 gives another illustration that our interactive scheme have the ability to create the fabric textons with different orientations (Fig. 4(d)). Such local deformations are repeated several times, while at each step the designer 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 and then is used as the reference texture for another base texture. This process will continue until a set of satisfactory textures $I_{l1}, I_{l2}, \ldots, I_{lk}$ with local deformation is produced in this stage.

Table 1
Pseudocode: wavelet-based texture design using energy optimization.

```
Input: the sample texture image Z.
Output: the designed texture image X.
        z_p^0 ← Initial neighborhood in Z, ∀p ∈ X†
        for iteration n = 0 : N do
                x^{n+1} ← arg min_x E_t(x; z_p^n) // E-step
                z_p^{n+1} ← arg min_z(‖x_p − z‖^2 + ‖c_{x_p} − c_z‖^2) // M-step
                // {z_p} is a neighborhoods set in Z
                if z_p^{n+1} == z_p^n, ∀p ∈ X†, then
                        x ← x^{n+1}
                        break
                end if
        end for
```

Since we want to replace the local region of the base texture with the specified texture elements in the reference texture, the labeling is a mapping to either I_{base} or I_s , and the image objective here is I_s. Therefore $E_d(p, L(p))$ is computed as follows:

$$E_d(p, L(p)) = \begin{cases} 0 & \text{if } L(p) = I_s \\ v & \text{if } L(p) \neq I_s \end{cases} \quad (2)$$

where v is a user specified large value.

The second term is defined by the distance between the pixels of I_{base} and I_s in a similar way as [19], that is

$$E_s(p, q, L(p), L(q)) = \begin{cases} 0 & \text{if } L(p) = L(q) \\ k_1 S_x + k_2 S_y & \text{otherwise} \end{cases} \quad (3)$$

where $S_x = \|C_{L(p)}(p) - C_{L(q)}(p)\| + \|C_{L(p)}(q) - C_{L(q)}(q)\|$, $S_y = \|\nabla G_{L(p)}(p) - \nabla G_{L(q)}(p)\| + \|\nabla G_{L(p)}(q) - \nabla G_{L(q)}(q)\|$, $\nabla G(p)$ is a 6-component color gradient (in R, G, B) at pixel p, and $k_1 = k_2 = 0.5$ is used throughout this paper.

3.3. Wavelet-based texture design using energy optimization

Kwatra et al. [27] presented a novel technique to synthesize textures using energy optimization. Their method treats the set of output pixel colors $\mathbf{x}$ as a high-dimensional variable, unlike most previous greedy heuristic-based algorithms [3,6,15], its value is determined by energy minimization. The texture energy $E(\mathbf{x})$ measures the perceptual similarity between the input sample and the output synthesized texture in terms of a simple local neighborhoods [27]. A limitation of their technique is that the misalignment of texture elements happens, because spatially distant neighborhoods communicate with each other only through intermediate overlapping neighborhoods. In order to alleviate the misalignment of texture elements, we extend their method [27] by defining the texture energy using wavelets.

The discrete wavelet transform (DWT) is a mathematical tool that can be used to describe 1D or 2D signals (images) in multiple

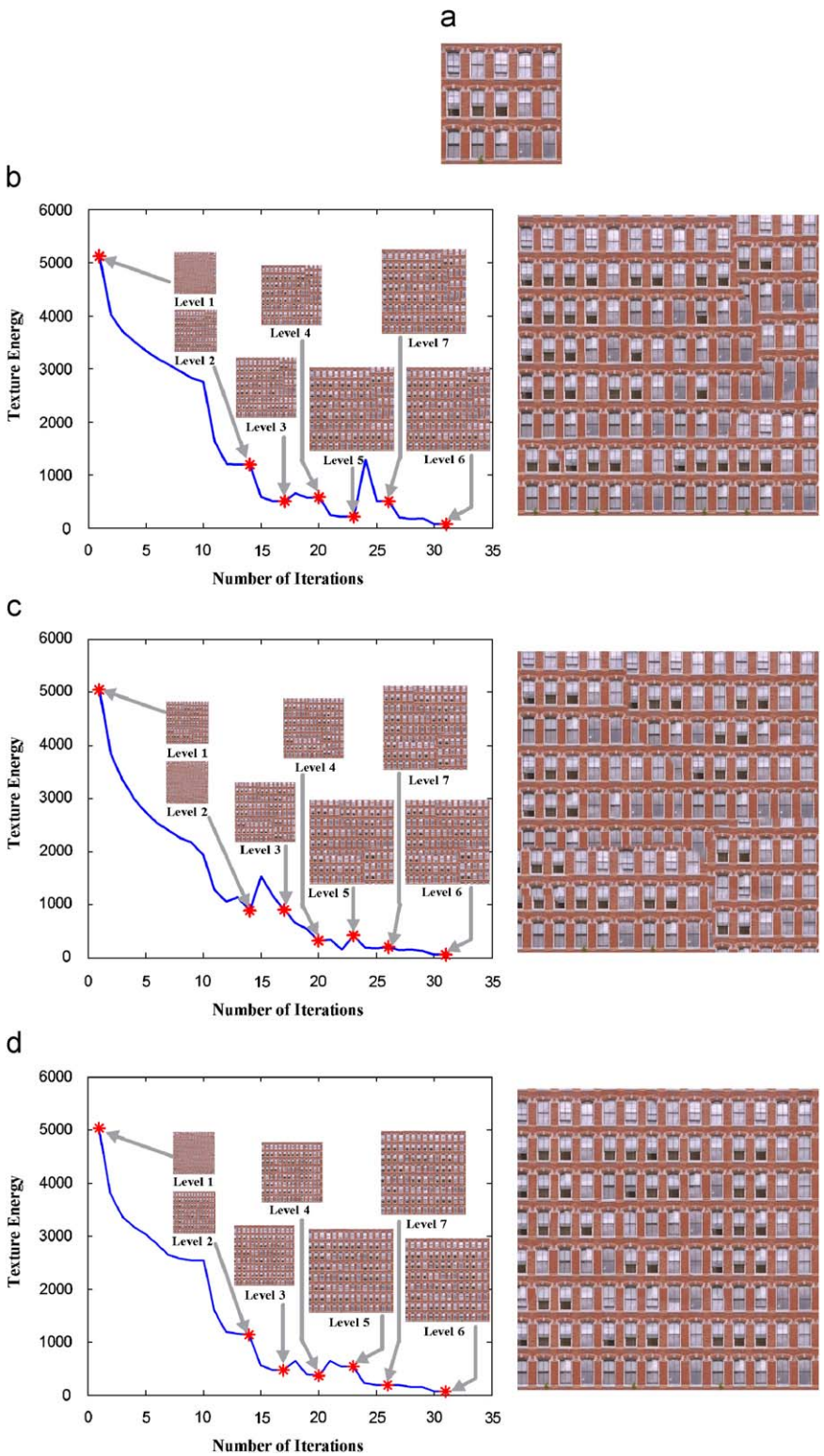

Fig. 5. Texture energy plotted as a function of number of iterations: (a) input; (b) result by texture-optimization [27]; (c) result by our Haar wavelet-based algorithm; and (d) result by our Daubechies wavelet-based algorithm. Level 1 shows the random initialization. Level 2 shows synthesis at ($\frac{1}{4}$ resolution, 8×8 neighborhood). Level 3: ($\frac{1}{2}$, 16×16). Level 4: ($\frac{1}{2}$, 8×8). Level 5: (1, 32×32). Level 6: (1, 16×16). Level 7: (1, 8×8).

resolutions. We believe that it is not the best approach for defining the energy function just by computing the raw pixel-to-pixel differences, the important image structures that are relevant in the human visual system may be disregarded [6,20]. Wavelet coefficients encode both information on the original pixels and multi-scale edge information [1,4]. One of the main contribution in this paper is to employ a wavelet-based multi-resolution method to compute the distance when defining the texture energy function. In other words,

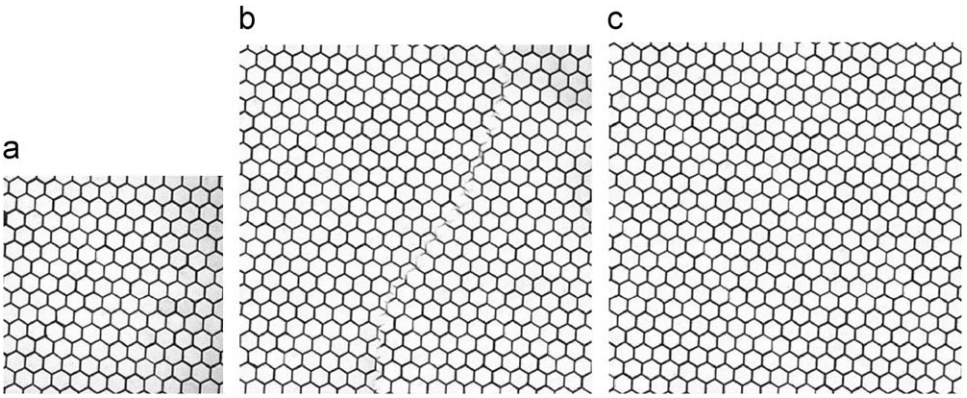

Fig. 6. Comparisons of our Daubechies wavelets-based algorithm with texture optimization [27]. (a) Input; (b) texture optimization [27] and (c) our method.

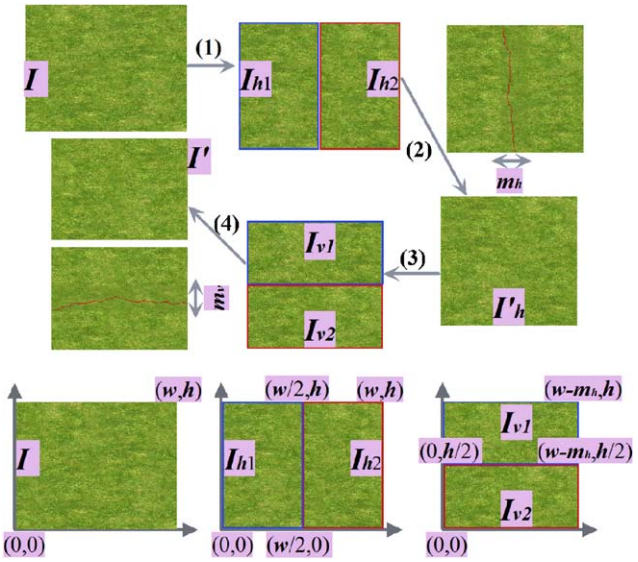

Fig. 7. Illustration of cyclic texture design algorithm using energy optimization algorithm (m_h=120, m_v=120, $I((0,0),(w,h))$, $I_{h1}((w/2+1,0),(w,h))$, $I_{h2}((0,0),(w/2,h))$, $I_{v1}((0,0),(w-m_h,h/2))$, $I_{v2}((0,\ h/2+1),(w-m_h,h))$.

we define the wavelet-based texture energy $E_t(\mathbf{x};\mathbf{z}_p)$ as follows:

$$E_t(\mathbf{x};\mathbf{z}_p)=\sum_{p\in X^{\dagger}}\left\{\frac{1}{N}\sum_{\Omega}\sum_{i=1}^{N}(\|\mathbf{x}_p-\mathbf{z}_p\|_{\Omega}^2+\|c^i_{\mathbf{x}_p}-c^i_{\mathbf{z}_p}\|_{\Omega}^2)\right\} \tag{4}$$

where X denotes the texture over which we want to compute the texture energy, Z is the input sample texture, N is the number of pixels in the neighborhood, $c^i_{\mathbf{z}_p}$ represents the values of the i-th wavelet coefficient in the neighborhood of $\mathbf{z}_p$, $\mathbf{x}$ is the vectorized version [27] of X at the color channel Ω, and $\mathbf{z}_p$ is the vectorized pixel neighborhood in Z whose appearance is the most similar to $\mathbf{x}_p$ using the wavelet-based similarity measurement. This energy function is optimized by running the EM-like steps multiple times until convergence, or a maximum number of iterations is reached. Refer Table 1 for details of these two steps (EM-like steps). For clarity of exposition, we summarize our wavelet-based texture design algorithm in pseudocode in Table 1.

In Fig. 5, the energy of the designed textures are plotted as a function of number of iterations by the Texture-Optimization technique in [27], our Haar wavelet [2]-based energy optimization and Daubechies wavelet [2]-based energy optimization algorithm, respectively. When the texture energy generally decreases as the number of iterations increase, the quality of the designed textures improves accordingly. Note that the final designed texture at the highest resolution and scale level by our Daubechies wavelet-based energy optimization algorithm achieves the best quality than the other ones (Fig. 5(b)–(d): right column).

In our work, we adopt the Daubechies wavelets [2] for minimizing the energy function because of its above robust characteristic and computational efficiency. In Fig. 6, we show more comparison results of the Texture-Optimization technique in [27] with our Daubechies wavelet-based energy optimization algorithm. The authors in [27] provide the texture synthesis results of the Texture-Optimization. Our wavelet-based energy optimization approach alleviates the misalignment of the texture elements, better than the method in [27].

3.4. Fast layer-based texture design using energy optimization

In order to design more deformed textures with varying patterns after stage (3), we introduce the fast layer-based texture design using energy optimization and deformation, which is inspired by the work in [29], further by incorporating both the global deformation. The SSD-based algorithm used is computationally expensive if the search is carried out exhaustively like [29]. Instead of performing a full exhaustive search in the example-based image completion stage, we employ the quadtree pyramid (QTP) [14] as our accelerating technique for finding the optimal texture patches. The experimental results demonstrate that our accelerating technique is capable of reproducing Shen's result [29] while is about five times faster than the completion-based texture design method in [29]. Moreover, we can extend it with more robust chaotic maps [7] beyond the basic logistic map.

4. Designing cyclic textures using energy optimization

Based on the existing non-parametric texture synthesis techniques [6,15,24], 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 using energy optimization.

The basic idea is that if we cut a tube and open it, we will get a sheet with its two sides satisfying the periodic condition. Therefore given an arbitrary texture, we can make it satisfy periodic condition in horizontal direction simply by first stitching its left side and right side together with Graph-cut-based energy optimization algorithm to form a tube, and then cut the tube along an arbitrary vertical line. Let w and h be the width and height of the large designed texture $I((0,0),(w,h))$, the details of the algorithm are described

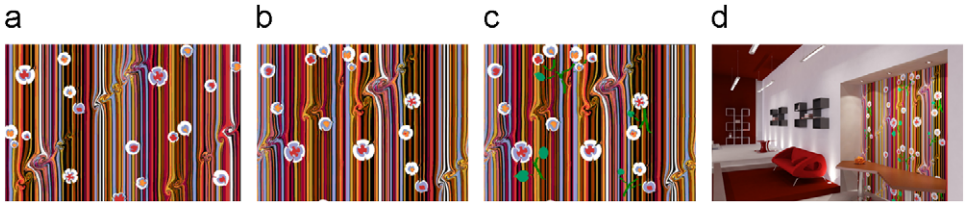

Fig. 8. Illustration of the cyclic marbling texture design. (a) An existing noncyclic marbling texture. (b) The cyclic texture created from (a) using our tool. (c) A new cyclic texture obtained by applying some marbling operations to (b). (d) A rendered image with the seamless tiling of (c). See the original article for more details [35].

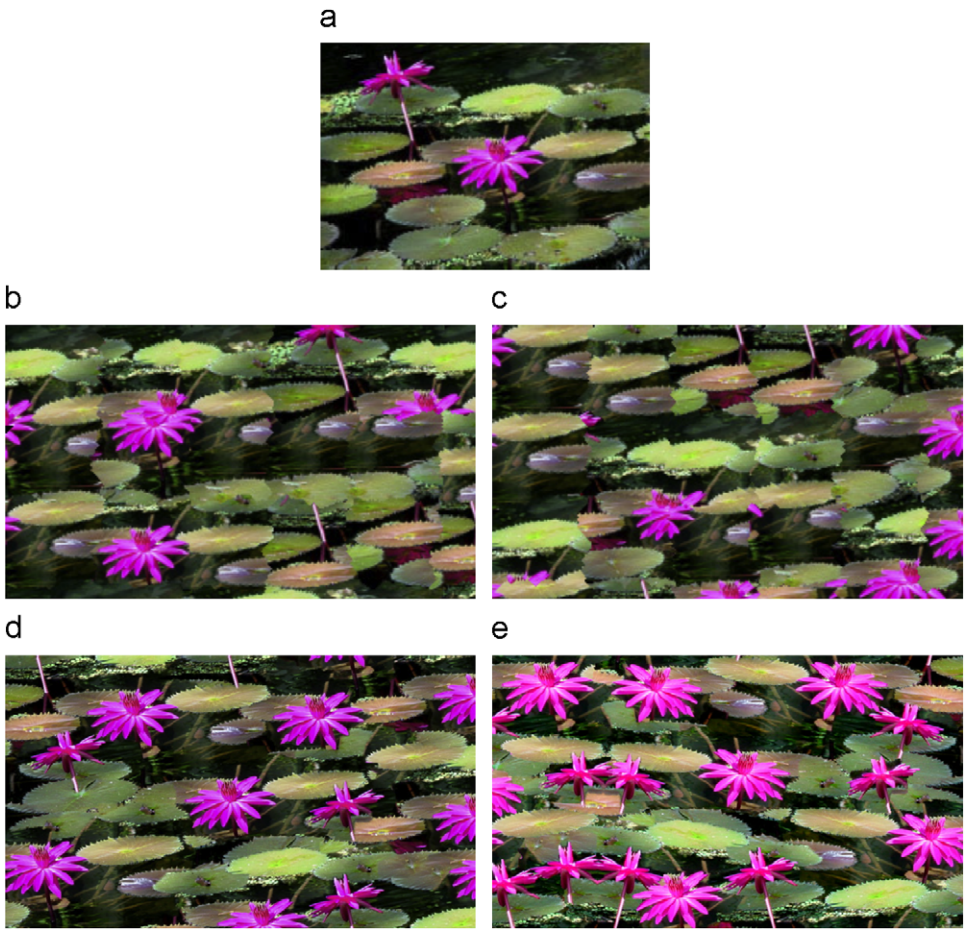

Fig. 9. Comparison of our energy optimization-based algorithm with Image Quilting [6], Wang Tiles [13], Graph-cut [15]. (a) Input. (b) Image Quilting [6]. (c) Patch-based [14]. (d) Graph-cut [15], and (e) Our method.

as follows (Fig. 7):

(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 DP or Graph-cut-based energy optimization algorithm is applied to synthesize seamlessly in horizontal direction. We denote the synthesized texture image as $I'_h = optimize(I_{h1}, I_{h2})$.
(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. We run the energy optimization algorithm to synthesize seamlessly in vertical direction. The final synthesized cyclic texture image is represented as $I' = optimize(I_{v1}, I_{v2})$.

Such a cyclic texture image can be used for seamless tiling in applications such as designing the cyclic marbling textures interactively. Marbled papers have been widely used in many ways, including picture framing, wrapping paper, stationary, collages, origami, and lamp-shades [35]. Marbling textures can decorate just about anything, even tissue boxes and cans. However, in previous systems [35], cyclic marbling textures can only be obtained by designing the pattern from scratch. Our scheme lets users create cyclic marbling textures from arbitrary existing textures. Our scheme allows designers to create cyclic marbling textures from arbitrary existing textures. Fig. 8 shows an application that uses an existing real marbling texture [35] as the base texture for creating a more intricate cyclic marbling texture.

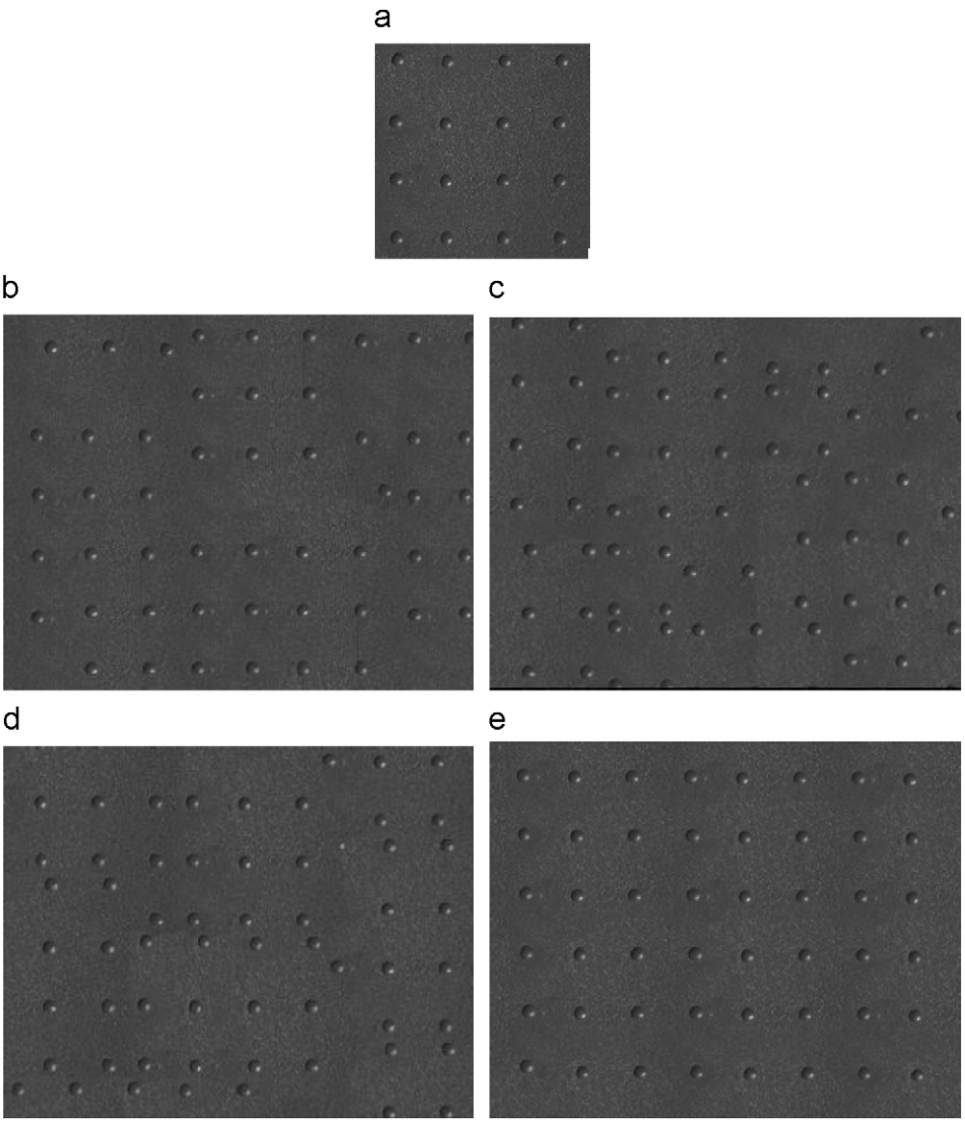

Fig. 10. A comparison of our energy optimization-based technique with the reported results of other patch-based techniques [6,14,15]. Note that our interactive texture design technique can handle the textures of a stochastic nature and preserve the global regularity, whereas other patch-based techniques usually cannot. (a) Input. (b) Image Quilting [6]. (c) Patch-based [14]. (d) Graph-cut [15], and (e) Our method.

5. Comparisons with other algorithms

We now compare our method to the other texture synthesis algorithms [6,14,13,15,27–29]. To better understand the differences with our energy optimization based approach, we look into the details of the previous methods in the following ways:

5.1. Image Quilting [6], Patch-based [14], Wang tiles [13], Graph-cut [15]

In Fig. 9, we compare our approach with other existing techniques of Image Quilting [6], Patch-based [14], Wang tiles [13] and Graph-cut [15]. The texture size is 268×230 for Fig. 9(a), and 360×360 for Fig. 9(b)–(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 [6] and Wang tiles [13], and is remarkable compared with the results produced by Graph-cut [15]. The sample texture in Fig. 9(a) consists of only two different lotus flowers. The techniques which 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. 9(d), 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. 9(e). For comparison, the density of the lotus flowers in our results (Fig. 9(e)) can be increased or decreased at the desired position according to the user's need. In Fig. 10, we further compared our energy optimization-based texture design technique to the reported results in [6,14,15]. From the designed textures, we can see that our method performs better than the reported results [6,14,15]. The global regularity is preserved in our designed results (Fig. 10(e)), whereas other techniques usually cannot.

5.2. Texture-Optimization [27]

Kwatra et al. [27] demonstrated an optimization scheme for texture synthesis using a Markov Random Field (MRF)-based similarity

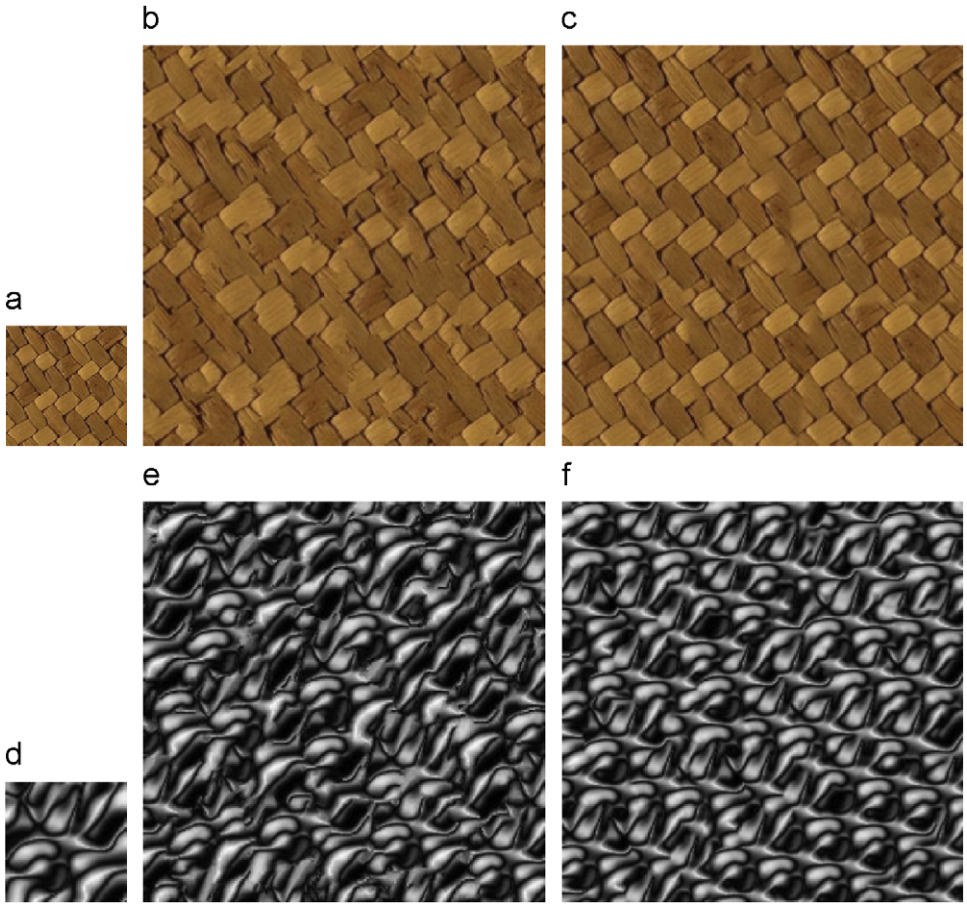

Fig. 11. Comparison of our energy optimization-based algorithm with parallel controllable texture synthesis [28].

metric. Their method formulates the texture synthesis problem as a minimization process of an energy function. A limitation of their technique is that the minimization of an energy function will get stuck in local minima when spatially distant neighborhoods communicate with each other only through intermediate overlapping neighborhoods [27]. This weakness will introduce the artifacts of the blurring or misalignment of texture elements. In this paper, the texture energy is defined using the wavelets, which alleviates the misalignment of the texture elements, better than the method in [27]. Figs. 6 and 13 give the comparison results using our method and the Texture-Optimization technique in [27], respectively. The results of Texture-Optimization are provided by the author in [27]. Our wavelet-based energy optimization approach achieves the better results than the one in [27].

5.3. Parallel controllable texture [28]

Lefebvre and Hoppe [28] presented a texture synthesis scheme based on neighborhood matching, control and parallelism. Their method have achieved the interactive speed for synthesizing the parallel controllable textures with high quality synthesis results. However, their approach required *Graphics Processing Unit* (GPU) support together with a few minutes of the preprocessing. Their method also shares the most common limitation of the neighborhood-based pre-pixel synthesis: if the semantic structures of the texture are not captured by small-neighborhoods, it will be impossible to synthesize the desired texture. Fig. 11 is another example demonstrating the effectiveness of our method, compared with the parallel controllable texture synthesis [28]. The advantage of our approach is that we produce superior synthesis quality than [28] due to our use of energy optimization, as demonstrated in Fig. 11.

Table 2
Computational statistics with [29] in Fig. 11.

Case	Output size	Patch size	Algorithm in [29]	Ours
(a)	360×360	32×32	113.349	21.675
(b)	360×360	64×64	114.532	22.193
(c)	360×360	92×92	118.712	22.216

Times are given in seconds. All performance timings are measured on the following platform: CPU (Pentium processor 2.2 GHz) with 1 GB RAM. We utilize the quadtree pyramid (QTP) acceleration technique for implementing the fast search algorithm in [29].

5.4. Completion-based texture [29]

We have combined the fast layer-based texture deformation technique into our interactive texture design scheme. This improvement using the QTP accelerating technique [14] is about five times faster than the completion-based texture synthesis method in [29], while is still capable of reproducing Shen's result. Table 2 gives the computational statistics by our method and Shen's one [29]. However, the second main limitation of Shen et al.'s method, lies in its lack of interaction over the local property of the resulting texture elements. In this article, we introduce the interactive texture design algorithm using energy optimization into our scheme, in order to allow the users design the texture as they want. Our presented method changes the

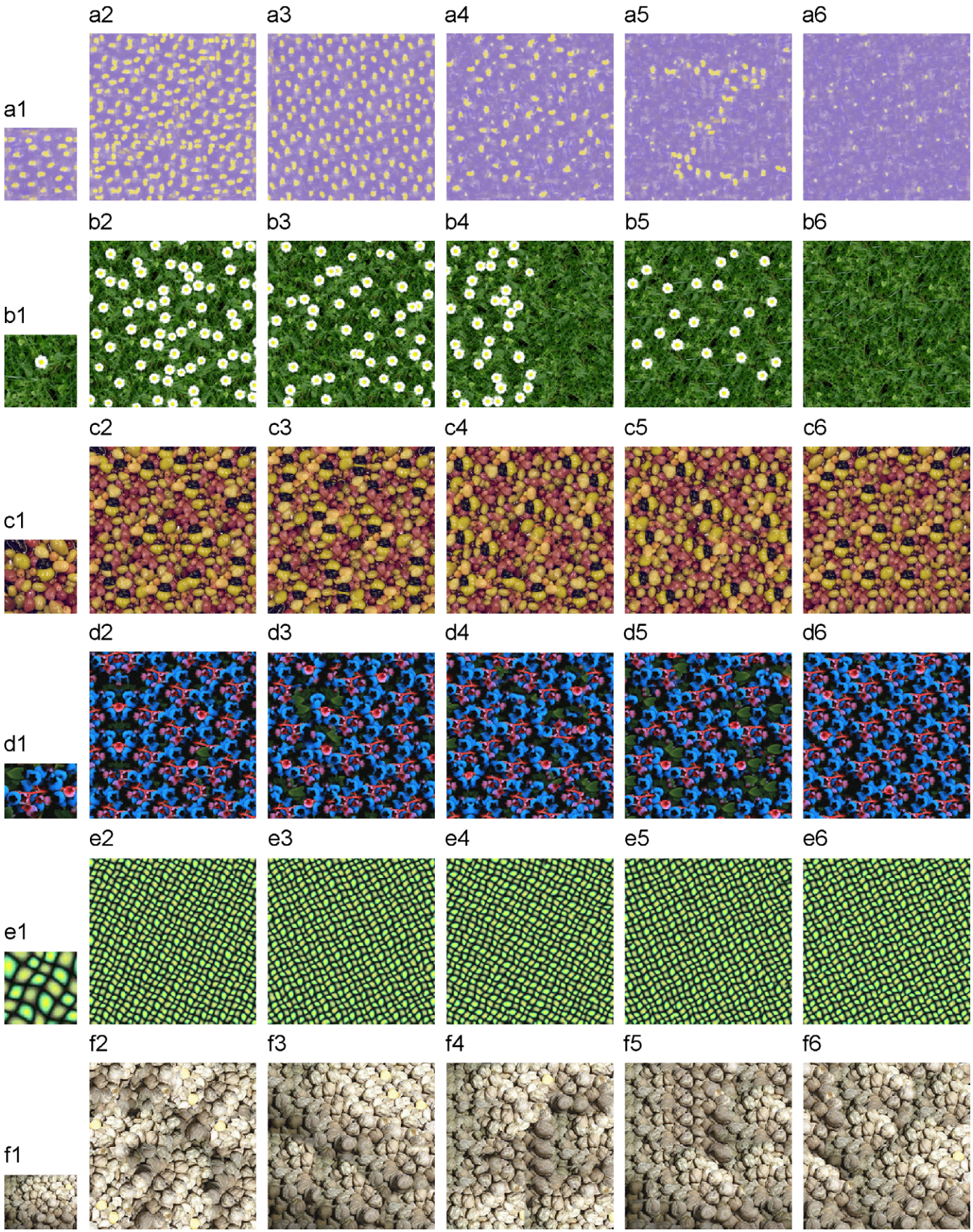

Fig. 12. Designed textures using our scheme. Left columns (a1, b1, c1, d1, e1, f1) are the input textures, the others are the deformed textures. Texture size: input: a1, b1,144 × 144; c1, 128 × 128; d1, 320 × 256; e1, 64 × 64; f1, 252 × 189; output: 360 × 360.

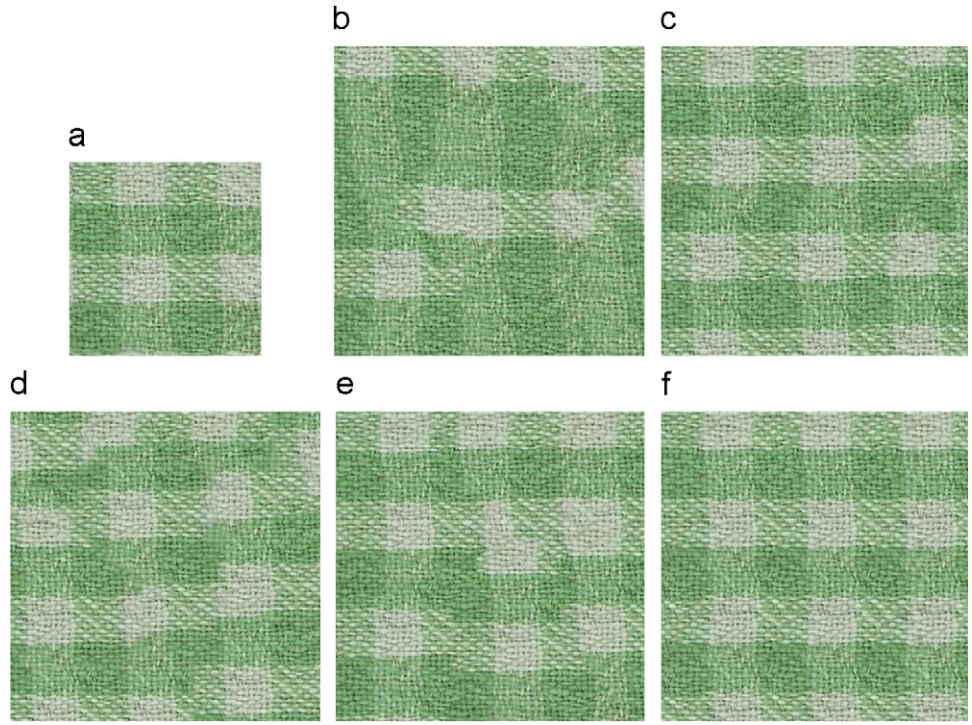

Fig. 13. Comparison of results with Image Quilting [6], Graph-cut [15], Texture-Optimization [27], and Completion Texture [29].

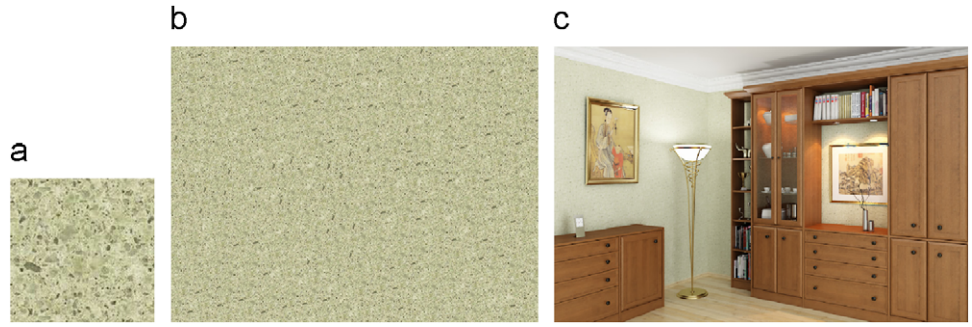

Fig. 14. A 3D rendering application using the designed textures by our scheme.

density of texture elements ("white flowers") interactively according to the designer's need. In Fig. 12(b2)–(b6), the density of the texture elements decrease gradually. We can also interactively generate the left part and the right part of the designed texture with different density (Fig. 12(b4)), create new texture elements, locally resize the texture elements (Fig. 12(b2)), design regular (Fig. 12(d6)) or irregular (Fig. 12(b2) and (b3)) texture patterns, and make the shape of designed texture look like a large "Z" shape (Fig. 12(a5)).

6. Experimental results and discussions

Our energy optimization based texture design algorithm has been applied to a variety of sample texture images. In our experiments, most of the source texture images are downloaded from the web sites.[1] For comparison, we only use those sample textures that have previously been used by other texture synthesis work [6,13–15,28,29]. All the experiments shown in this section were run on a PC with 2.2 GHz Pentium processor and 1 GB of RAM. The results are mostly sized at 360×360 pixels designed from different-sized sample textures ranging from 128×128 to 256×256 pixels.

Further results of this method are shown in Fig. 13. Fig. 13 shows an example in which the patched-based algorithm [6,15,27,29] fails to add the irregular details but which is handled by our energy optimization based algorithm very well. Although the input sample has good quality, the Image Quilting synthesis [6] (Fig. 13(b)), Graph-cut [15] (Fig. 13(c)), Texture-Optimization [27] (Fig. 13(d)) and Completion-Texture [29] (Fig. 13(e)) fails to reproduce the irregular structures of weave. Applying energy optimization based texture design scheme incorporating the deformation, the results become very pleasing (Fig. 13(f)). Fig. 12 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.

Fig. 14 demonstrates another interesting application of our technique, which shows a 3D rendering result using the designed textures by our interactive scheme. In Fig. 14, the realistic effects are

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

created for 3D rendering application using the wide variety of designed textures.

7. Conclusions and future work

We have proposed a unified framework for designing textures using energy optimization in this paper. Our experimental results demonstrate both the feasibility and effectiveness of our approach. The main advantage of our scheme over other existing texture synthesis methods lies in its capability to create a wide variety of natural textures, only from a single small sample texture, considering both the local deformation of texture elements and the texture designer's creation. By combining the fast layer-based texture deformation method, and the wavelet-based energy optimization approach into an interactive texture design scheme, we have designed textures with good stochastic property. Moreover, a novel cyclic texture design technique using energy optimization is also developed under our interactive scheme for efficiently creating large seamless marbling textures. As a result, our approach shows a particular strength in generating various deformed textures from a single sample texture while avoiding highly repetitive patterns. Our experimental results also demonstrate that the proposed scheme has been applied to other applications such as designing cyclic marbling textures.

Although our texture design scheme has produced good results, our future work is to develop more powerful energy optimization algorithms. Currently, we are extending our approach from still texture design to surface texture [16] and solid texture design [37].

Acknowledgments

We would like to appreciate the anonymous reviewers for their helpful suggestions. The work was supported by RGC research grants (Grant nos. 416007 and 412708), MS-CU-JL visiting Grant 2008, the Science and Technology Plan of Zhejiang Province (Grant no. 2008C24008), the NIH-NIAAA Grant U01-AA014809-04, and the Open Project Program of the State Key Lab of CAD&CG (Grant no. A0912), Zhejiang University.

Appendix A. Supplementary data

Supplementary data associated with this article can be found in the online version at doi:10.1016/j.patcog.2009.03.003.

References

[1] S. Mallat, S. Zhong, Characterization of signals from multiscale edges, IEEE Transactions on Pattern Analysis and Machine Intelligence (1992) 710–732.
[2] J.S. Walker, A Primer on Wavelets and their Scientific Applications, CRC Press, Boca Raton, FL, USA, 1999.
[3] L.Y. Wei, M. Levoy, Fast texture synthesis using tree-structured vector quantization, in: Proceedings of SIGGRAPH '00, New Orleans, ACM, New York, 2000, pp. 479–488.
[4] J. Portilla, E.P. Simoncelli, A parametric texture model based on joint statistics of complex wavelet coefficients, International Journal of Computer Vision 40 (1) (2000) 49–70.
[5] A. Hertzmann, C.E. Jacobs, N. Oliver, B. Curless, D.H. Salesin, Image analogies, in: Proceedings of SIGGRAPH '01, 2001, pp. 327–340.
[6] A.A. Efros, W.T. Freeman, Image quilting for texture synthesis and transfer, in: Proceedings of SIGGRAPH '01, Los Angeles, ACM, New York, 2001, pp. 341–346.
[7] G. Jakimoski, L. Kocarev, Chaos and cryptography: block encryption ciphers based on chaotic maps, IEEE Transactions on Circuits System-1: Fundamental Theory and Applications 48 (2) (2001) 163–169.
[8] J.-M. Dischler, K. Maritaud, B. Lévy, D. Ghazanfarpour, Texture particles, Computer Graphics Forum 21 (3) (2002) 401–410.
[9] S. Brooks, N. Dodgson, Self-similarity based texture editing, ACM Transactions on Graphics 21 (3) (2002) 653–656.
[13] M.F. Cohen, J. Shade, S. Hiller, O. Deussen, Wang tiles for image and texture generation, ACM Transactions on Graphics 22 (3) (2003) 287–294.
[14] L. Liang, C. Liu, Y.Q. Xu, B.N. Guo, H.Y. Shum, Real-time texture synthesis by patch-based sampling, ACM Transactions on Graphics 20 (3) (2001) 127–150.
[15] V. Kwatra, A. Schödl, I. Essa, G. Turk, A. Bobick, Graphcut textures: image and video synthesis using graph cuts, ACM Transactions on Graphics 22 (3) (2003) 277–286.
[16] J. Zhang, K. Zhou, L. Velho, B. Guo, H.-Y. Shum, Synthesis of progressively-variant textures on arbitrary surfaces, ACM Transactions on Graphics 22 (3) (2003) 295–302.
[19] A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, M. Cohen, Interactive digital photomontage, ACM Transactions on Graphics 23 (3) (2004) 294–302.
[20] Q. Wu, Y. Yu, Feature matching and deformation for texture synthesis, ACM Transactions on Graphics 23 (3) (2004) 362–365.
[21] Y. Boykov, V. Kolmogorov, An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (9) (2004) 1124–1137.
[22] Y. Liu, W.C. Lin, J.H. Hays, Near regular texture analysis and manipulation, ACM Transactions on Graphics 23 (3) (2004) 368–376.
[24] W. Matusik, M. Zwicker, F. Durand, Texture design using a simplicial complex of morphable textures, ACM Transactions on Graphics 24 (3) (2005) 787–794.
[27] V. Kwatra, I. Essa, A. Bobick, N. Kwatra, Texture optimization for example-based synthesis, ACM Transactions on Graphics 24 (3) (2005) 795–802.
[28] S. Lefebvre, H. Hoppe, Parallel controllable texture synthesis, ACM Transactions on Graphics 24 (3) (2005) 787–794.
[29] J.B. Shen, X.G. Jin, X.Y. Mao, J.Q. Feng, Completion based texture design using deformation, The Visual Computer 22 (9) (2006) 936–945.
[30] C. Rother, L. Bordeaux, Y. Hamadi, A. Blake, AutoCollage, ACM Transactions on Graphics 25 (3) (2006) 847–852.
[31] J. Jia, J. Sun, C.-K. Tang, H.-Y. Shum, Drag-and-drop pasting, ACM Transactions on Graphics 25 (3) (2006) 631–637.
[32] D. Charalampidis, Texture synthesis: textons revisited, IEEE Transactions on Image Processing 15 (3) (2006) 777–787.
[33] S. Lefebvre, H. Hoppe, Appearance-space texture synthesis, ACM Transactions on Graphics 25 (3) (2006) 541–548.
[35] J.Y. Xu, X.Y. Mao, X.G. Jin, Nondissipative marbling, IEEE Computer Graphics and Applications 28 (2) (2008) 35–43.
[36] G. Ramanarayanan, K. Bala, Constrained texture synthesis via energy minimization, IEEE Transactions on Visualization and Computer Graphics 13 (1) (2007) 167–178.
[37] J. Kopf, C.-W. Fu, D. Cohen-Or, O. Deussen, D. Lischinski, T.-T. Wong, Solid texture synthesis from 2D exemplars, ACM Transactions on Graphics 26 (3) (2007) Article no. 2.

About the Author—JIANBING SHEN is an associate professor in the School of Computer Science and Technology at Beijing Institute of Technology. Prior to that, he was a post-doctoral researcher in Department of Computer and Information Science, Indiana University-Purdue University Indianapolis, USA, from 2007 to 2008. He received his PhD degree in Computer Science from Zhejiang University in 2007. His current research interests include texture synthesis, image completion, non-photorealistic rendering, and high dynamic range imaging and processing.

About the Author—HANQIU SUN is an associate professor in the Department of Computer Science and Engineering, Chinese University of Hong Kong. She received the MS degree in electrical engineering from the University of British Columbia and the PhD degree in computer science from the University of Alberta, Canada. Her research interests include virtual and augmented reality, interactive graphics/animation, hypermedia, internet-based visualization and navigation, computer-assisted surgery, and touch-enabled simulations.

About the Author—JIAYA JIA received the PhD degree in computer science from the Hong Kong University of Science and Technology in 2004. He joined the Department of Computer Science and Engineering, Chinese University of Hong Kong, in September 2004, where he is an assistant professor. His research interests include vision geometry, image/video editing and enhancement, motion deblurring, and Markov random field analysis. He has served on the program committees of ICCV, CVPR, ECCV, and ACCV. He is a member of the IEEE.

About the Author—HANLI ZHAO is a PhD candidate of the State Key Lab of CAD&CG, Zhejiang University, China. He received his BSc degree in software engineering from Sichuan University in 2004. His research interests include non-photorealistic rendering and general purpose GPU computing.

About the Author—XIAOGANG JIN is a professor at 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.

About the Author—SHIAOFEN FANG is a professor and Chair of the Department of Computer and Information Science at Indiana University Purdue University Indianapolis. He received his PhD from University of Utah in 1992, and his MS and BS from Zhejiang University, China, in 1986 and 1983, respectively. His research interests include Visualization, Computer Graphics, Geometric Modeling and Medical Imaging. Prof. Fang's research has been funded by the US National Science Foundation, National Institutes of Health, and National Institute of Justice.