Efficient Example-Based Painting and Synthesis of 2D Directional Texture

Bin Wang, Wenping Wang, Huaiping Yang, and Jiaguang Sun

Abstract—We present a new method for converting a photo or image to a synthesized painting following the painting style of an example painting. Treating painting styles of brush strokes as sample textures, we reduce the problem of learning an example painting to a texture synthesis problem. The proposed method uses a hierarchical patch-based approach to the synthesis of directional textures. The key features of our method are: 1) Painting styles are represented as one or more blocks of sample textures selected by the user from the example painting; 2) image segmentation and brush stroke directions defined by the medial axis are used to better represent and communicate shapes and objects present in the synthesized painting; 3) image masks and a hierarchy of texture patches are used to efficiently synthesize high-quality directional textures. The synthesis process is further accelerated through texture direction quantization and the use of Gaussian pyramids. Our method has the following advantages: First, the synthesized stroke textures can follow a direction field determined by the shapes of regions to be painted. Second, the method is very efficient; the generation time of a synthesized painting ranges from a few seconds to about one minute, rather than hours, as required by other existing methods, on a commodity PC. Furthermore, the technique presented here provides a new and efficient solution to the problem of synthesizing a 2D directional texture. We use a number of test examples to demonstrate the efficiency of the proposed method and the high quality of results produced by the method.

Index Terms—Digital painting, example-based painting, painting style, artistic filter, painting systems, simulation, image segmentation, Gaussian pyramid, texture synthesis, directional texture, nonphotorealistic rendering.

1 Introduction

Digital painting involves the use of computers to assist or automate the process of painting. Some existing methods for digital painting simulate the characteristics of an artistic medium, while others attempt to automatically create paintings by simulating artistic processes or results [10]. Here, we consider the problem of *example-based painting*.

This problem can be formulated as follows. (Refer to Fig. 1.) Given two images as input, an *example painting* A' whose style is to be simulated and an image (often a photo) B, the task is to produce as output a *synthesized painting* B' which has the same contents of B but has the painting style of A'. We also call B the *source image* and B' the *synthesized image*. Some variants of this problem have been considered before [7], [9], [13], [18].

Among the existing methods for learning the style of an oil painting, the best results are produced by the approach of *image analogies* proposed in the seminal work of Hertzmann et al. [18]. But, the image analogies (IA) method is rather slow since it is pixel-based—it normally takes a few hours to generate a synthesized painting on a PC.

We present a new method that shortens the synthesis time greatly—our method takes less than one minute and, often, only a few seconds to generate a synthesized painting of size 512×512. For example, the image in Fig. 1c (size 800×600 pixels) was generated by our method in six seconds. Moreover, the paintstrokes generated by our method can be made to follow a direction field determined by the shapes of regions in the painting.

The basic idea of our method is as follows: The user specifies in the example painting one or more small blocks of sample textures that best represent the distinct styles to be simulated. Then, to better represent and communicate shapes and objects, we segment the image B' to be synthesized according to the contents of the source image B and define a direction field in each segmented region. Finally, we use a hierarchy of texture patches, assisted with image masks, to synthesize directional textures in each segmented region to form the final synthesized painting; a hierarchy of patches of different sizes used to achieve a balance between computational efficiency and the need for ensuring a smooth representation of changing texture directions and image masks are used to resolve irregular region boundaries. Through the quantization of texture direction, we are able to use preprocessing and Gaussian pyramids to further speed up the texture synthesis process.

2 Related Work

A large number of papers on digital painting and texture synthesis have been produced and we will review only those directly related to our work. Painterly rendering is an active research topic in NPR (nonphotorealistic rendering). Some early successful efforts include [12], [33]. Some recent work includes [11], [17], [23], [29], [31].

Example-based rendering has also been studied for different forms of artistic drawing. Freeman et al. [9]

- *B. Wang, H. Yang, and J. Sun are with Tsinghua University, Beijing 100084, P.R. China.
E-mail: {bwang, hpyang}@csis.hku.hk, sunjg@tsinghua.edu.cn.*
- *W. Wang is with the Department of Computer Science and Information Systems, The University of Hong Kong, Pokfulam Road, Hong Kong.
E-mail: wenping@csis.hku.hk.*

*Manuscript received 4 Apr 2003; revised 22 July 2003; accepted 21 Aug. 2003.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-0023-0403.*

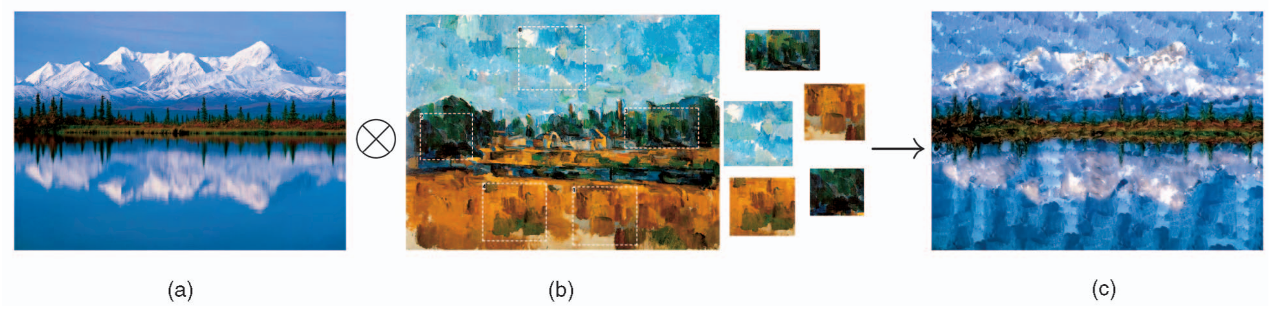

Fig. 1. The source image (a) (Photo credit: Superstock Images) is converted to the synthesized painting (c) having the style of the example painting (b). (a) B: a source image. (b) A': an example image and user-selected stroke textures. (c) B': a synthesized image.

generate line drawings using the best linear combination of training lines to fit input lines. Hamel and Strothotte [13] capture the drawing style of a 3D model and reuse it to render another 3D model by extracting and applying a nonphotorealism-template. Jodoin et al. [22] use a curve segment as the basic element in learning the hatching style of a given example based on Shannon's N-gram approach and Efros and Freeman's [7] and Efros and Leung's [8] texture synthesis model. Hertzmann et al. synthesize oil paintings by example using the framework of image analogies [18] and transfer the style of one 2D curve to another using a similar framework [19]. Note that none of the above methods represents painting styles as sample textures and uses efficient texture synthesis techniques to solve the example-based painting problem with changing brush stroke directions. We present a solution to this problem in this paper.

Texture synthesis has been an active research topic in the recent years [2], [16]. Earlier methods based on the pixel-based approach include [1], [8], [14], [18], [35], [37], [38], [39], [41] for texturing either a 2D image or a surface.

Xu et al. [40] introduce the patch-based approach, which has proven much faster than pixel-based methods. They select square patches of a uniform size randomly from the sample texture and alpha-blend the overlapping parts of adjacent patches to form a synthesized texture. Efros and Freeman [7] improve Xu et al. [40] by stitching together similar texture patches along a minimum error boundary and use their method to transfer a sample texture to form a synthesized image resembling a target image or photo, with the aid of a correspondence map. Liang et al. [25] extend Xu et al. [40] by finding the patch in the overlapping region which most closely resembles the adjacent patches already synthesized and compositing them using feather blending. This method gains real-time performance by using several speedup techniques to find the best-matching patch. All existing patch-based methods for 2D texture synthesis are fast but are limited to patches of a uniform size and a fixed direction. In this paper, we develop a fast method using a hierarchy of patches of different sizes to synthesize textures with varying directions.

Patch-based methods have also been proposed for texturing a surface [27], [32]. Soler et al. [32] use triangular patches for texturing a mesh surface, using Fast Fourier Transform to speed up the search for a best-matching patch. A straightforward application of these methods to synthesizing 2D directional textures is conceivable, but would be relatively inefficient.

3 Outline of Algorithm

Given a source image B and an example painting A', our method synthesizes a painterly rendering B' of B with the style of A'. The image B' starts as a blank image and is "painted" as the algorithm proceeds to form the final synthesized painting. Our method has the following main steps:

1. **Painting style selection:** The user specifies in the example painting A' one or more small *blocks*, called *sample textures*, which contain the painting styles that he/she wants to simulate (Fig. 1b).
2. **Preprocessing:** Let $T_1, T_2, \ldots, T_k$ denote sample textures selected in Step 1. The direction of a sample texture, defined as the average direction of brush strokes contained therein, is quantized to be one of 24 discrete directions, i.e., with the angle increment being $360^\circ/24 = 15^\circ$. For each sample texture T_i, we precompute and store 24 rotated copies of it. Each copy is stored in a Gaussian pyramid for accelerating the search for the best-matching patch, as required in Step 5 below.
3. **Segmentation and direction assignment:** In order to use different styles and different brush stroke directions to represent shapes and objects in the synthesized image, we segment the source image B into regions $R_1, R_2, \ldots, R_m$ of relatively uniform color and luminance, which in turn induce their corresponding regions $R'_1, R'_2, \ldots, R'_m$ in the synthesized image B'. We then define a *direction field* on B', that is, a texture direction (i.e., brush stroke direction) is assigned automatically to each pixel of B' according to the shape of the region that contains that pixel. This direction is also called a *pixel direction*, for short.
4. **Painting style assignment:** Assign a sample texture T_i (i.e., a style of brush strokes) to each segmented region of the synthesized image B'. This assignment can be done automatically based on luminance similarity or by user hints.
5. **Texture synthesis:** The image B' is partitioned into a hierarchy of square cells of different sizes, with each cell to be synthesized by one or more square texture patches of the same size (see Section 6 for details). A cell in a segmented region R'_j of B' is synthesized using a texture patch taken from the sample texture T_i assigned to R'_j. A patch from T_i is rendered into B'

with its color and luminance modulated by the color and luminance of the corresponding pixel of the source image B and with its direction following the direction field of B'.

4 Processing Painting Styles as Sample Textures from A'

To select a style of brush strokes, the user needs to manually specify one or more rectangular blocks of sample textures in the example painting that he/she thinks represent the painting styles to be simulated (Fig. 1b). Some care should be taken in defining the sample textures. First, a block should include only one sample texture (i.e., one group of brush strokes of similar size and shape); otherwise, different types of brush strokes will be interleaved in the synthesized image, giving unsatisfactory results. Second, a block of sample texture should be bigger than the size of a typical brush stroke in order to avoid producing fragmented strokes in the synthesized image. Since a sample texture is a group of brush strokes, it is typically anisotropic. We use the algorithm of Tamura et al. [34] to compute the direction of a sample texture, which is the average direction of brush strokes contained in the sample texture.

To speed up patch searching, we quantize texture directions to 24 discrete angles, i.e., $i \times 15^\circ$, $i = 0, 1, \ldots, 23$. This quantization enables us to precompute and store 24 different rotated copies of a sample texture. We therefore merely need to search for a patch in one of the precomputed rotated copies of the sample texture. By exploiting symmetry about the coordinate axes, memory space can be saved by storing only six rotated copies of a sample texture image. A similar strategy of preprocessing textures is also used in [32].

5 Processing of Source Image B

5.1 Segmentation of Source Image

The basic structures in a painting of a scene are defined by objects in the scene and these structures are determined by the colors, styles, and directions of brush strokes in different parts of the painting. We use the mean-shift method [4] to segment the source image into regions of relatively uniform color and luminance, which in turn induce a segmentation of the synthesized image B', through the pixel-wise one-to-one correspondence between B and B'. We then use these segmented regions to assign styles and directions of the brush strokes so as to best represent and communicate the shapes present in the final synthesized image B'.

5.2 Assignment of Sample Textures to Segmented Regions

It is desirable to assign painting styles (i.e., sample textures) to different regions efficiently or automatically. This can be done in two ways. The first possibility is to use luminance similarity, assigning to a region a sample texture that is closest to it in mean luminance. Alternatively, the user can provide some hints by manually assigning sample textures to some regions. Then, an unassigned region will get the same sample texture as an assigned region that is closest to the unassigned one in mean luminance. This saves the user the trouble of manually assigning sample textures to a large number of regions.

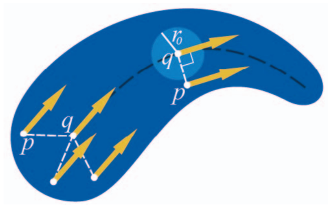

Fig. 2. The brush stroke direction at a pixel p is set to be the direction of a pixel q on the medial axis nearest to p.

5.3 Computation of Paint Stroke Direction

Appropriately defined brush stroke directions can help make a synthesized painting look less artificial. There are several possible ways of defining brush stroke directions across the synthesized image B'. A simple treatment is to assign the same direction for all pixels in a region; this direction can be a user-specified direction or the main direction of the region's shape [24]. This treatment is easy to implement and has proven effective for small regions or elongated regions with a distinct main direction. However, for large regions, such as the one shown in Fig. 2, it is better to use a varying stroke direction following the medial axis to reflect the shape of the region. (Note that the medial axis is known to communicate shape information effectively [28].) To this end, we first compute the medial axis of a region R, assign to each pixel q on the medial axis a direction along the medial axis curve at q, and then set the stroke direction at each pixel p of the region R to be the direction of the pixel q on the medial axis that is closes to p. See Fig. 2.

The medial axis of a region is also used by Gooch et al. [11] to place stroke brushes. Although there exist many algorithms for medial axis computation, extracting the medial axis from a region with an irregular boundary is a tricky task. As pointed out in [11], the thinning algorithm is too sensitive to the boundary noise and tends to produce undesirable spurs along the medial axis. The distance transform does not preserve the connectedness of the medial axis and often leads to double lines. Therefore, these two algorithms are used in combination in [11]; the distance transform is first used to find the medial axis and a thinning algorithm is then applied to remove double lines from the medial axis.

We adopted the method used in [11] with some modifications and improvements. We first apply the Euclidean Distance Transform (EDT) [6] to a region R to obtain the distance field of R. (This EDT algorithm is very fast; it takes only two passes to produce a distance map for all the regions in the segmented image.) We then compute the distance gradient vector at each pixel of R from the distance field. It is observed that, for a region $\bar{R}$ defined in 2D Euclidean plane E^2, a point x on the medial axis $\bar{R}$ is characterized by that there exist two points x_1 and x_2 in any open neighborhood of x such that the gradient vectors at x_1 and x_2 are opposite to each other. Hence, for our case where the region R is defined in the discrete 2D image plane, we regard a pixel p as a pixel on the medial axis of R if the largest angle difference among all the gradient vectors of the eight neighboring pixels of p is greater than a certain threshold α_M. Our tests show that satisfactory results are produced in most cases by setting $\alpha_M = 0.7\pi$.

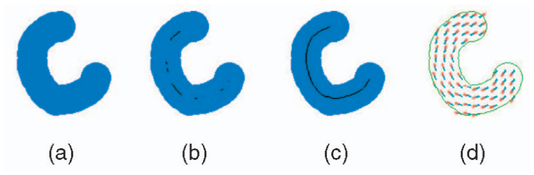

Fig. 3. An example of medial axis computation. From left to right: (a) the original segment, (b) the medial axis extracted by Gooch et al.'s method, (c) the medial asix extracted by our method, and (d) its associated direction field.

After finding all the pixels on the medial axis, we run a thinning algorithm to remove double lines, if there are any, to obtain a "leaner" medial axis. Our tests show that this method produces satisfactory results; for example, the disconnectedness of the medial axis computed by our method is much less of a problem than the result produced by the method of [11], as shown in Fig. 3. Once the medial axis is obtained, each pixel q on the medial axis is assigned the direction along the medial axis at q. This direction is computed as the main direction of the set S of pixels that are both on the medial axis and in a "disk" of radius r_0 centered at the pixel q (see Fig. 2). The main direction of a set of points S is computed as the eigenvector associated with the largest eigenvalue of the covariance matrix of S [24].

Finally, we use the EDT algorithm to compute the distances of all pixels of the region R from its medial axis curve. Because of the incremental nature of the EDT algorithm, it can easily be adapted to find, for each pixel p in the region R, the pixel q on the medial axis that is nearest to p. Then, the brush stroke direction at pixel p is set to be the direction of the pixel q. See Fig. 2 for illustration. Finally, a direction field thus obtained is made smoother by applying a Gaussian filter. The last step of our algorithm bears some similarity with the orientation field computation method presented by Hausner [15], which computes the minimum distance of a point from a prespecified curve using a Z-buffer algorithm.

The brush stroke directions computed above enhance the visual appearance of a synthesized painting for large segmented regions with a distinctive shape feature, but appear to make little difference for small regions. Hence, to strike a balance between the visual appearance and computational efficiency, we use the directions based on the medial axis only for regions containing over $M = 1{,}000$ pixels; for a smaller region R, the main direction of R is assigned to all pixels of R.

6 Directional Texture Synthesis for B'

6.1 Procedure of Texture Synthesis

In this part, we explain the procedure and acceleration techniques for synthesizing directional textures in a segmented image. There are three main steps in this process.

1. Decompose the image B' into regular cells, with each cell of the same size as the largest patch size specified (typically 32×32).
2. Recursively subdivide each cell into four smaller cells of equal size until the directions of pixels belonging to the same region in every cell do not differ from each other by a prespecified tolerance; this is called the *direction constraint*. This step produces a hierarchy of cells of different sizes.
3. Synthesize each cell by compositing patches of different sample textures contained in that cell. All cells are synthesized in the depth-first order, with the cells at the same level visited from bottom to top and left to right.

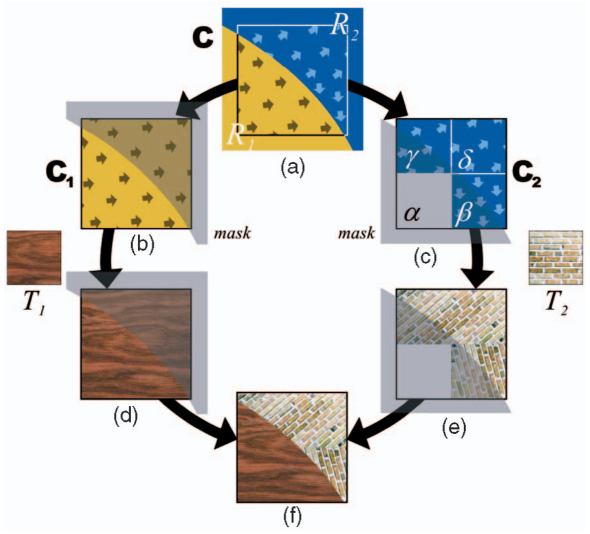

Fig. 4. The cell C crossing a boundary is synthesized by superimposing two cells C_1 and C_2, with the aid of image masks.

Since the brush strokes in the same patch are required to have approximately the same direction, an abrupt change of texture direction might cause a cell to be subdivided in order to satisfy the direction constraint. We set the smallest cell size to be 4×4 pixels, and subdivision is applied only to cells of size larger than 4×4 pixels. The subdivision of a cell is terminated if and only if the cell satisfies the direction constraint or its size is 4×4.

For a cell crossing a region boundary, one might also want to recursively subdivide it into smaller cells such that each cell belongs to one region. But, this would produce too many small cells near region boundaries. We therefore use image masks to assist in synthesizing a cell crossing a boundary, without resorting to cell subdivision. The basic idea is illustrated in Fig. 4. The cell C in Fig. 3a contains pixels from two regions R_1 and R_2, corresponding to sample textures T_1 and T_2, respectively. We may think of C as obtained by superimposing a cell C_1 of texture T_1 and a cell C_2 of texture T_2, with the aid of image masks, as shown in Fig. 3b and Fig. 3c. We now need to check C_1 and C_2 separately to see if they need to be subdivided to satisfy the direction constraint.

We first check C_1. As all of its pixels in R_1 satisfy the direction constraint, a patch of the same size from texture T_1 will be found and pasted onto the cell C_1. Next, we check the cell C_2 and find that its pixels belonging to R_2 do not satisfy the direction constraint. C_2 is therefore subdivided into four smaller cells, α, β, γ, and δ (Fig. 4c). At this level of subdivision, some of these four cells contain no pixels in the region R_2 (i.e., α) and others contain only pixels meeting the direction constraint (i.e., β, γ, and δ); thus, we terminate cell subdivision and use three patches of the same size but

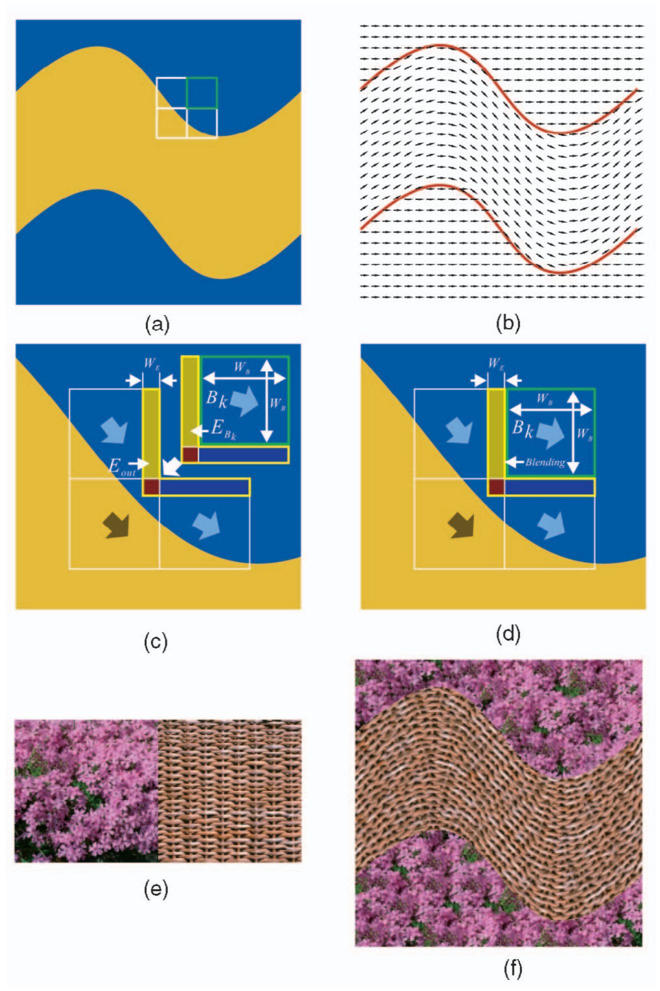

Fig. 5. The procedure of finding the best-matching patch.

possibly different directions to fill in the cells β, γ, and δ (see Fig. 4e). Finally, with the aid of image masks, the synthesized cells C_1 and C_2 are superimposed to yield the synthesized cell C (Fig. 4f).

A cell may contain pixels from different segmented regions. The region boundaries in a cell are rendered precisely through the use of image masks, instead of using cell subdivision. In order to represent the smooth variation of the directional field, the number of texture rotations should be large enough and the patch size should be small enough. An adequate number of texture rotations make it possible to accurately approximate the arbitrary orientation of the directional field, while small-sized patches are needed to capture a localized radical direction change of the directional field. In our experiments, we find that 24 discrete angles and a minimum patch size of 4×4 pixels are sufficient for satisfactorily accommodating the change of texture direction in most cases.

The procedure of searching for a best-matching patch in a rotated copy of a sample texture is similar to that used in [25], except that we use two-level Gaussian pyramids for acceleration. This procedure is explained in Fig. 5. Fig. 5a shows a source image B. Fig. 5b shows a direction field of the segmented image B'. Fig. 5e shows two sample textures to be applied. The three white cells in Fig. 5a have already been synthesized and a patch is now to be found to

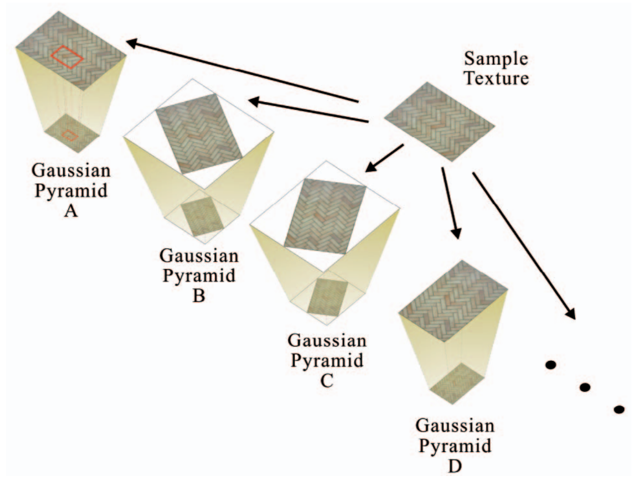

Fig. 6. Gaussian pyramids for acceleration.

synthesize the green cell (Fig. 5a). Fig. 5c is a zoom-in view of Fig. 5a. Let E_{B_k} denote the L-shaped boundary strip of a candidate patch B_k and let E_{out} denote the corresponding L-shaped boundary strip of the union of the three white cells. The boundary strip E_{B_k} of the best-matching candidate B_k should minimize its difference with E_{out}. Setting the width of boundary strip E_{B_k} to be four pixels was found, in [25], to work well as a balance between quality and speed. We also set this width to be four pixels in our experiments. Fig. 5d shows the blend between the best-matching patch B_k with the already synthesized texture. Fig. 5f is the final synthesis result following the direction field in Fig. 5b.

6.2 Acceleration by Gaussian Pyramid

We use two-level Gaussian pyramids [3] to speed up the search for the best-matching patch. We choose the Gaussian pyramid instead of the three acceleration techniques recommended by Liang et al. [25] (i.e., Optimized KD-tree, Quad-tree pyramid, and PCA) because the Gaussian pyramid consumes less memory than those methods. Our experimental results confirm that the use of Gaussian pyramids speeds up patch searching by $5 \sim 10$ times. Note that the Gaussian pyramid is also used in the pixel-based texture synthesis methods in [18], [38], but in a way somewhat different from ours.

Fig. 6 shows a collection of two-level Gaussian pyramids used to represent rotated copies of a sample texture. A two-level Gaussian pyramid stores a sample texture, or a rotated copy, at the high level and a low-resolution version of the sample texture at the low level. The low-level copy (size $n \times n$) has only half the resolution of the high-level copy (size $2n \times 2n$) and it is obtained from the latter by applying a 3×3 Gaussian filter. We may therefore assign a patch of size $k \times k$ at the low level to correspond to four patches of size $2k \times 2k$ at the high level.

In search of the best-matching patch from a sample texture, we first construct the low-resolution version of the L-shaped boundary region E_{out} (see Fig. 5c). With a relaxed matching error tolerance, we use the low-resolution version of E_{out} to find up to 10 low-resolution candidate patches of the best-matching patch in the low level of a Gaussian pyramid. We then get, at the high level, the candidate patches (40 at most) corresponding to the low-resolution

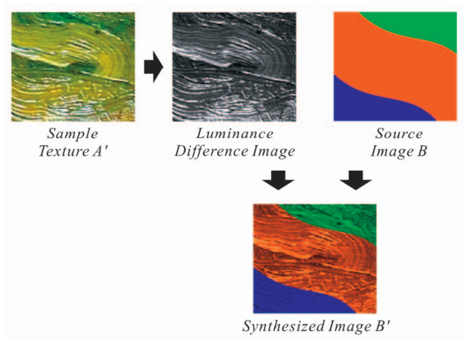

Fig. 7. The synthesized image B' has the luminance variation of the sample texture A' but the color of the source image B.

candidate patches and choose among them the one with the smallest matching error with E_{out} as the best-matching patch B_k.

6.3 Rendering a Painting Style

Luminance variation is an important characteristic of brush strokes of an oil painting. Therefore, when rendering a patch of a sample texture from the example painting A' onto the synthesized image B', the luminance variation of the sample texture is preserved and the color of the rendered patch is determined by the color of the corresponding pixel of the source image B. In this way, we obtain a synthesized image B' that has the color contents of the source image B and the painting style of the example painting A'. Fig. 7 illustrates this operation of obtaining a synthesized result.

Using the YIQ color space, we transfer luminance variation and color as follows: Let $Y(b')$ be the luminance of a pixel b' in the synthesized image B'. Let $Y(a')$ be the luminance of a pixel a' in the example painting A', where a' is to be mapped to the pixel b'. Let $\mu_{A'}$ be the mean luminance of a sample texture in A' and μ_{B_R} be the mean luminance of a region R in B which contains the pixel b corresponding to b'. Then, $Y(b')$ is computed by

$$Y(b') = \kappa(Y(a') - \mu_{A'}) + \mu_{B\_R}, \qquad (1)$$

where κ is a constant. Setting $\kappa = 1$ makes the synthesized image inherit the luminance variation of the example painting. Smaller or larger values of κ can be used to lessen or strengthen the luminance variation of brush strokes in the synthesized image. Setting $\kappa = \sigma_B/\sigma_{A'}$ yields the formula used in [18] for luminance remapping, where σ_B and $\sigma_{A'}$ are the luminance standard deviations of B and A', respectively. Another possibility is to use the formula $Y(b') = \kappa(Y(a') - \mu_{A'}) + Y(b)$ to make the synthesized result also depend on the luminance variation of the source image. For all the examples shown in this paper, Figs. 11 and 12 were generated using (1) and all the other examples were generated using $Y(b') = \kappa(Y(a') - \mu_{A'}) + Y(b)$.

7 Experimental Results

In this section, we present some synthesized images generated by our method. We set $\kappa = 1.0$ for illuminance mapping defined by (1) for all the following examples, except that we set $\kappa = 1.2$ for Fig. 8c. As for the parameters in Comaniciu-Meer's segmentation algorithm [4], we used

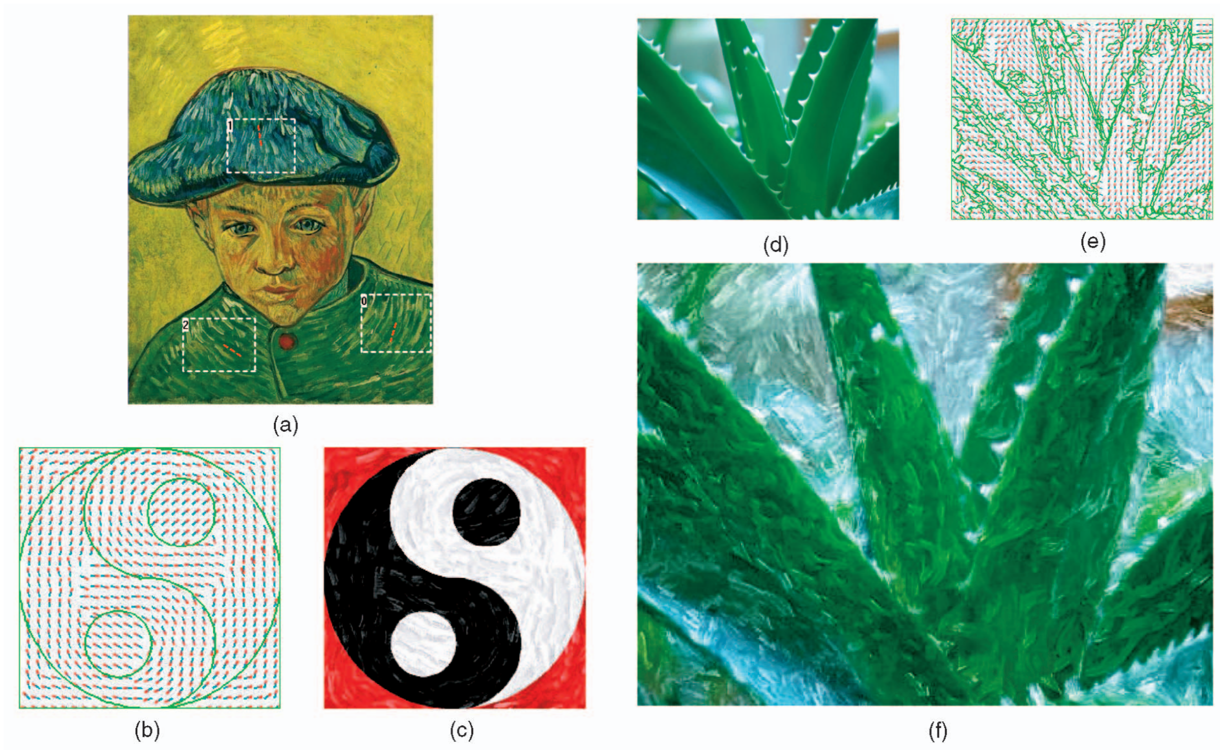

Fig. 8. (a) Vincent Van Gogh's Portrait of Camille Roulin. (b) A direction field of TaiJi following the medial axis. (c) TaiJi painted in a style learned from (a) ($\kappa = 1.2$, size 256×256, generated in six seconds). (d) An aloe image (size 416×320). (e) A direction field following the medial axis. (f) The aloe image painted in a style learned from (a) ($\kappa = 1.0$, generated in 15 seconds).

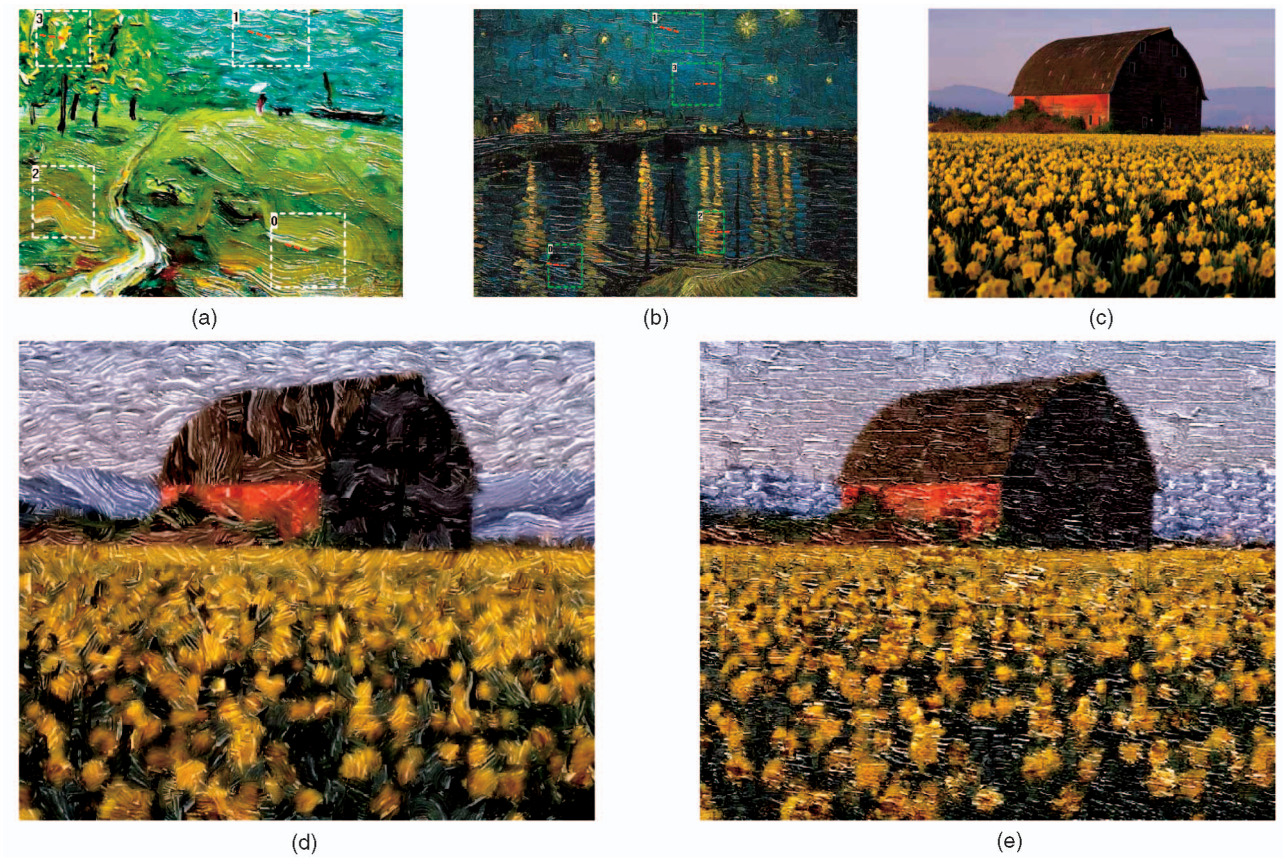

Fig. 9. (a) An oil painting by Suhuo (http://www.suhuo.com). (b) Van Gogh's *Starry Night above the Rhone*. (c) A source image *The Barn* (size 576×480) (photo credit: Corbis Images). (d) *The Barn* painted in a style from (a) ($\kappa = 1.0$, generated in six seconds). (e) *The Barn* painted in a style from (b) ($\kappa = 1.0$, generated in five seconds).

the default values (spatial bandwidth = 7, color bandwidth = 6.5, and minimum region = 100) and produced acceptable segmentation in all the following examples, except that we set spatial bandwidth to 11 and color bandwidth to 5 for Fig. 13b.

Fig. 8a is Vincent Van Gogh's Portrait of Camille Roulin. The three blocks shown in Fig. 8a are the selected sample textures and the red line segments indicate the orientations of the sample textures. Fig. 8b is the direction field of a TaiJi, which is a traditional symbol of the oriental culture, following the direction of the medial axis (see Section 5). Fig. 8c is the corresponding synthesized image (size 256×256), which was generated in six seconds. The timings for all the test examples presented in this paper were taken on a PC with a 2GHz CPU.

Fig. 8d is another source image, with its segmentation and direction field shown in Fig. 8e; here, the direction field also follows the medial axis of each region. Fig. 8f is the final synthesized image (size 416×300), which was generated in 15 seconds.

Next, we show an example in which two synthesized images are generated from the same source image by following two different painting styles. Fig. 9a and Fig. 9b show two example paintings of different styles. Fig. 9a is an oil painting by Su Huo, a Chinese artist (http://www.suhuo.com) and Fig. 9b is Vincent Van Gogh's famous oil painting *Starry Night above the Rhone*. Fig. 9c shows a source image. Fig. 9d and Fig. 9e are the painterly renderings generated by our method from the same source image (Fig. 9c), with their painting styles following those of Fig. 9a and Fig. 9b, respectively. The main direction is used for defining the direction field for each region for generating Fig. 9d. A fixed horizontal direction is used across the entire image for defining the direction field for generating Fig. 9e. The synthesized painting Fig. 9e resembles the synthesized image generated by the *Image Analogies* method [18] from the same example image and source image. However, we note that our method took only five seconds to generate Fig. 9e (size 576×480) on a PC with a 2GHz CPU, while the *Image Analogies* method [18] used a few hours on a PC with a 1GHz CPU to generate a similar result.

Fig. 10a is Emil Nolde's painting, *Autumn Sea VII* (http://www.artchive.com/artchive/N/nolde.html). Fig. 10d shows a synthesized image of the source image Fig. 1a following the painting style of Fig. 10a with the direction field being the main direction for each segmented region. Fig. 10e shows another synthesized image of the same source image, also following the painting style of Fig. 10a, but with the direction field (shown in Fig. 10b) being based on the medial axis for each segmented region. It is evident that the brush strokes of the latter image using medial-axis-based directions demonstrate much less undesirable periodicity than the former using the main direction. The generation of Fig. 10d and Fig. 10e took 8 seconds and 55 seconds, respectively. Fig. 10g shows another example of a synthesized painting of the source image in Fig. 10f, following the style of Fig. 10a and with the direction field determined by the medial axis for each region (shown in Fig. 10c). Fig. 10g was generated in 56 seconds.

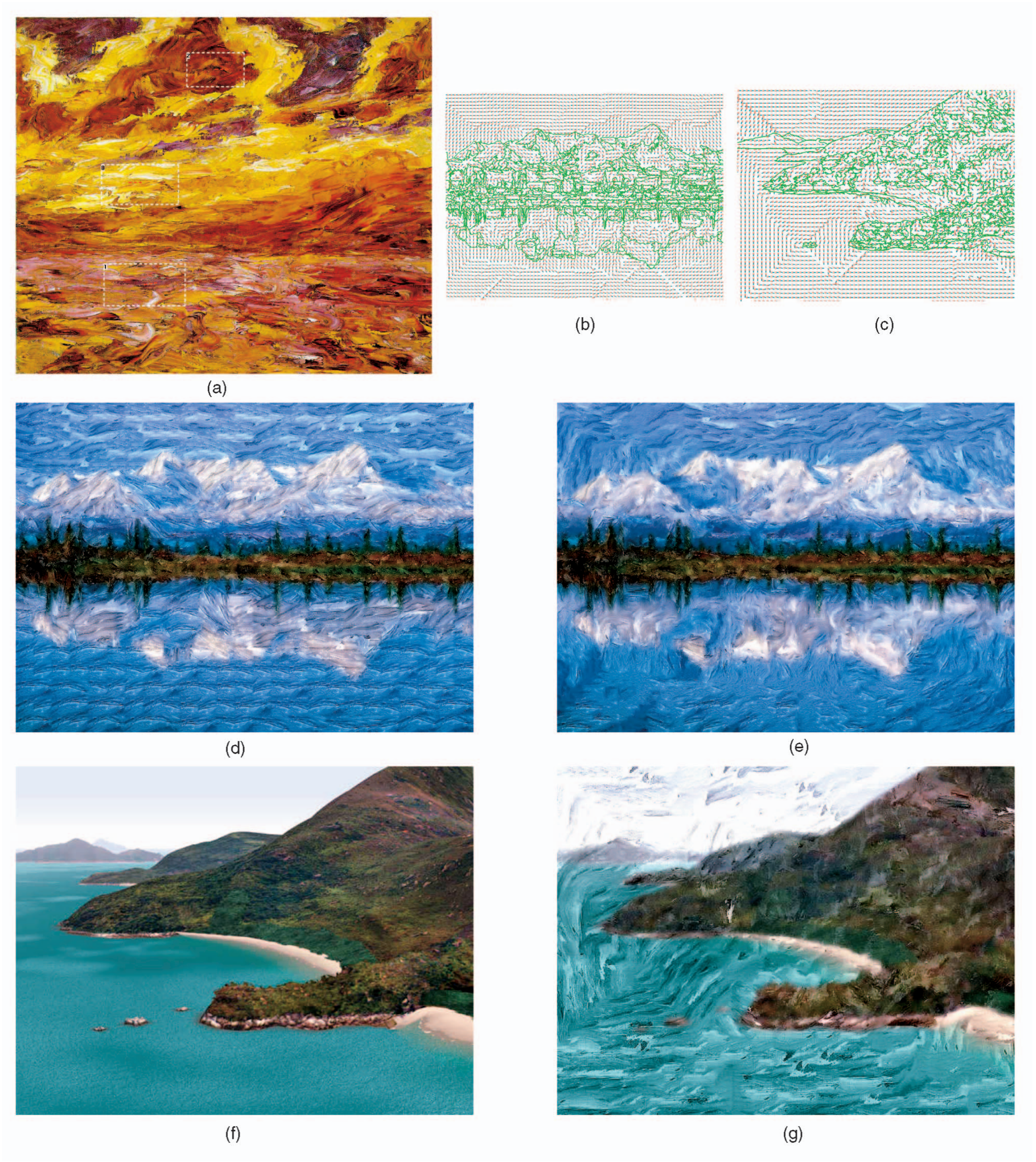

Fig. 10. (a) Emil Nolde's *Autumn Sea*. (b) A direction field of Fig. 1a following the medial axes of segmented regions. (c) A direction field of (f) following the medial axes of segmented regions. (d) A synthesized painting of the source image Fig. 1a, with the brush stroke directions following the main direction ($kappa = 1.0$, size 800×600, generated in eight seconds). (e) A synthesized painting of the same source image with the brush stroke directions following the direction field shown in (b) (generated in 55 seconds). (f) Another source image (size 672×512). (g) A synthesized painting of the source image in (d) with the bursh stroke directions following the direction field shown in (c) ($\kappa = 1.0$, generated in 56 seconds).

In general, we observe that defining the brush stroke directions based on the medial axis produces better results, but requires longer time than using the main direction to define the brush stroke directions; the processing time for the former case is normally less than one minute for an image of size 512×512. Longer processing time is required when the direction field is given by the medial axis because more small-sized texture patches are generated in the texture synthesis process in order to accommodate the varying directions of the brush strokes.

Besides oil paintings, our method is applicable to other types of artwork as well, including watercolor, pastel, and engraving, as will be illustrated below. Fig. 11a is a watercolor [5]. Fig. 11d and Fig. 11e are the respective

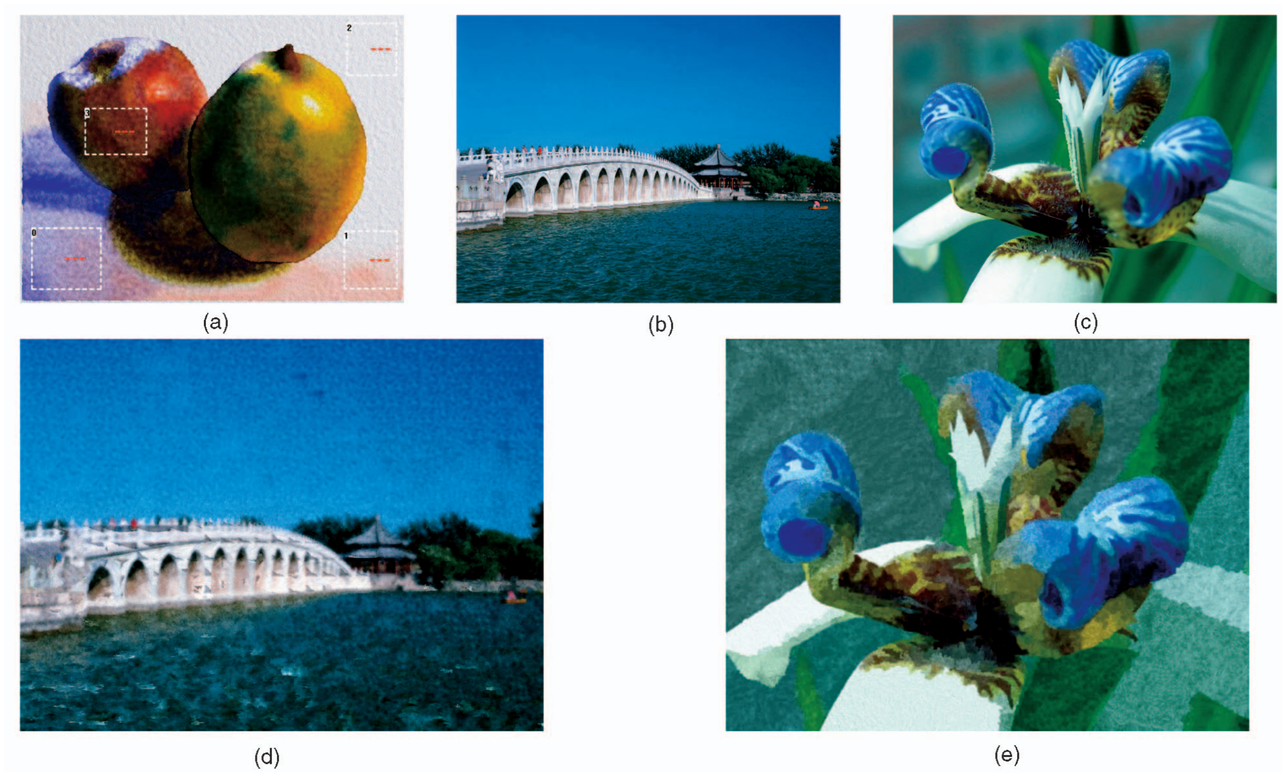

Fig. 11. (a) A watercolor painting. (b) A source image *The Bridge* size (992×744). (c) Another source image *The flower* (size 800×600). (d) *The Bridge* painted in the style of watercolor ($\kappa = 1.0$, generated in 14 seconds, with the brush stroke directions following the region main direction). (e) *The flower* painted in the style of watercolor ($\kappa = 1.0$, generated in 4 seconds, with the brush stroke directions following the medial axis direction).

synthesized images of the source images in Fig. 11b and Fig. 11c, following the painting style of Fig. 11a. The generation of Fig. 11d and Fig. 11e took 14 seconds and 40 seconds, respectively.

Fig. 12a is a pastel artwork [30]. Fig. 12d and Fig. 12e are the respective synthesized images of the source images in Fig. 12b and Fig. 12c [26], following the painting style of Fig. 12a. Fig. 12d and Fig. 12e were both generated in eight seconds and their visual appearance is comparable to that of the results synthesized by the *Image Analogies* method [18] from the same source images (available from http://www.mrl.nyu.edu/projects/image-analogies/pastel.html). Horizontal brush stroke directions are used in Fig. 12d and Fig. 12e for making the comparison with the results generated by the IA method.

Fig. 13a is an example engraving artwork. Fig. 13b is a source image, with its segmented image shown in Fig. 13c. Fig. 13d is the synthesized image (size 384×576), generated in one second. This result compares favorably in visual appearance with the result generated by the *Image Analogies* method [18] (available from http://www.mrl.nyu.edu/projects/image-analogies/dorotea.html). Again, for comparison with the IA method, a fixed direction is used to define brush stroke directions in Fig. 13d.

Our method can also be applied to synthesizing a directional texture, provided that a sample texture and a direction field are given. Fig. 14 shows three synthesized images of directional textures (size 512×512), with their corresponding sample textures and direction fields. The generation of each of these images in Fig. 14 took 15 seconds. These examples suggest the potential application of our method in vector field visualization.

As we pointed out earlier, our method is applicable to many but certainly not all painting styles. In general, the algorithm would fail for paintings whose styles cannot be represented by directional textures. This is the case for some oil paintings in which the brush textures are not obvious or some works by impressionists or post-impressionists in which brush textures have strong chrominance variations.

Fig. 15 shows a failure example in which we tried to learn the style of Monet's *The Japanese Bridge* (http://www.artchive.com/artchive/M/monet.html). This work possesses a distinct style; the brush strokes are short and curved, with a variety of different colors within a brush stroke. Our framework is incapable of simulating such a style since we use only the variation of luminance. Further research is needed to develop a more sophisticated model to simulate this kind of impressionistic painting.

8 Conclusion

We have presented an efficient method for generating a synthesized painting from a photo or image that possesses the painting style of a given example painting. The method treats painting styles as sample textures that the user can select from the example painting and generates the synthesized painting using a hierarchical patch-based approach to synthesize directional textures. A synthesized painting (size 512×512) is typically generated in a time ranging from a few seconds to one minute, depending on the complexity of the brush stroke directions used.

Two features of our algorithm contribute to the significant speedup over previous work. The first is the use of a patched-based algorithm for directional texture synthesis. The second is the use of a small search space for finding the best-matching textures through the user selection of sample textures. This allowance of the user specification of painting styles helps reduce the size of the search space for texture

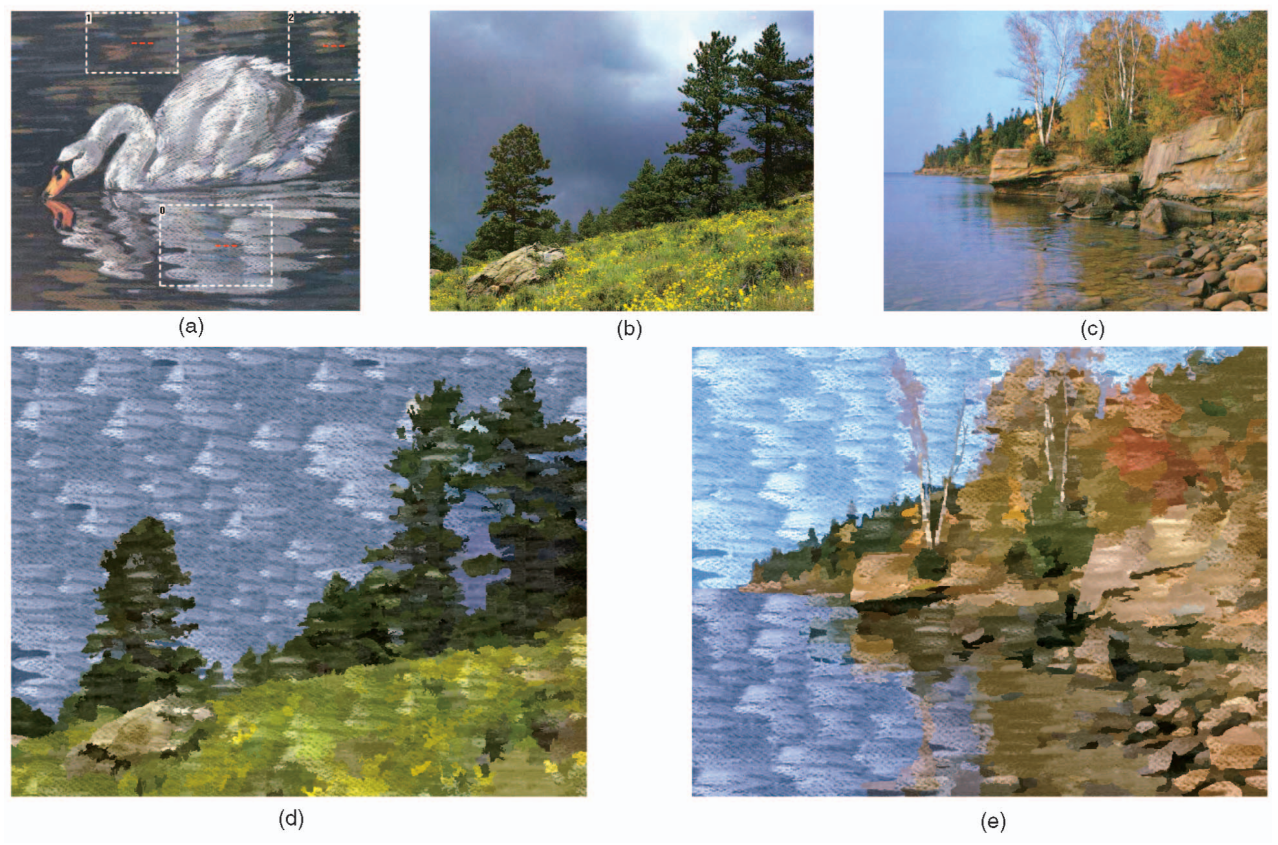

Fig. 12. (a) A pastel painting. (b) A source image *The Dark Cloud* (size 640×480). (c) Another source image *The Shore* (size 640×480). (d) *The Dark Cloud* painted in the style of pastel by our method ($\kappa = 1.0$, generated in eight seconds). (e) *The Shore* painted in the style of pastel by our method ($\kappa = 1.0$, generated in eight seconds). Horizontal stroke directions are used for both (d) and (e).

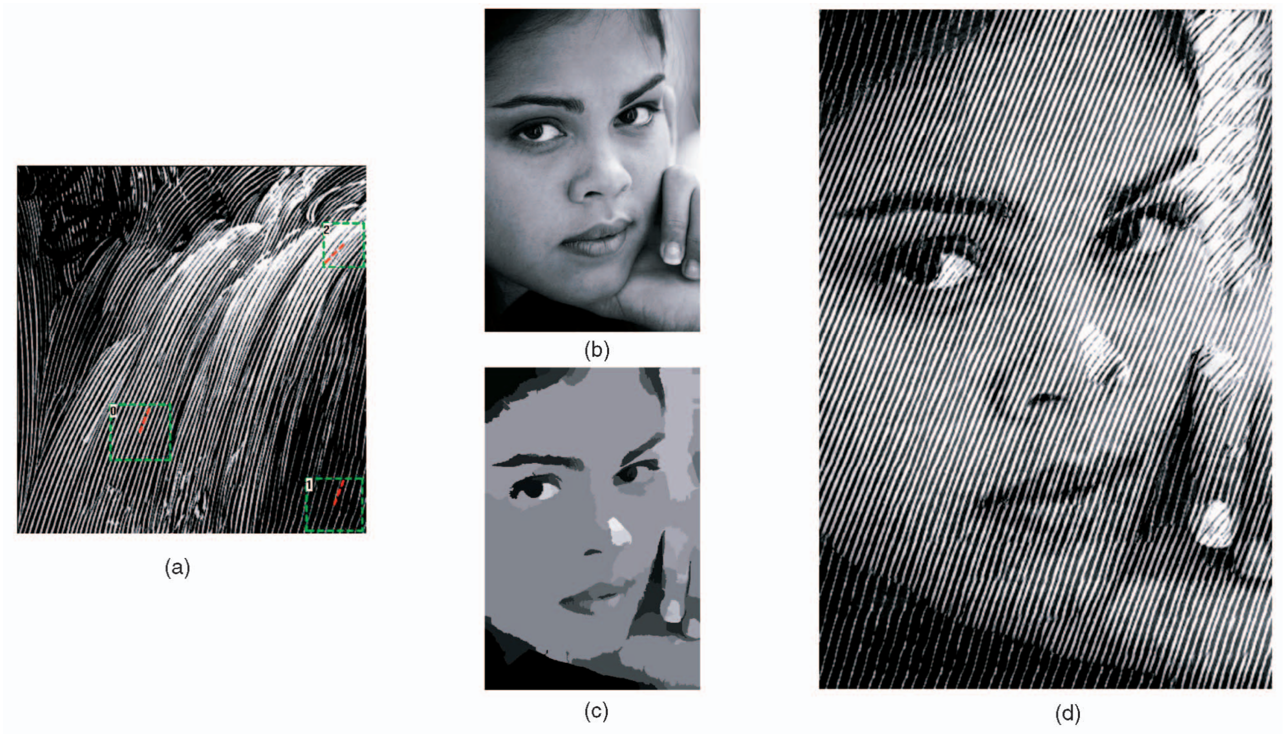

Fig. 13. (a) An engraving artwork. (b) A source image *The Girl's Face* (size 384×576) (photo credit: Getty Images). (c) A segmented image of *The Girl's Face*. (d) *The Girl's Face* painted in the style of engraving ($\kappa = 1.0$, generated in one second, with a fixed stroke direction).

synthesis and, thus, together with the other speed-up techniques, contributes to the remarkable reduction in painting synthesis time. The resulting superior efficiency makes our method particularly suitable as a tool for

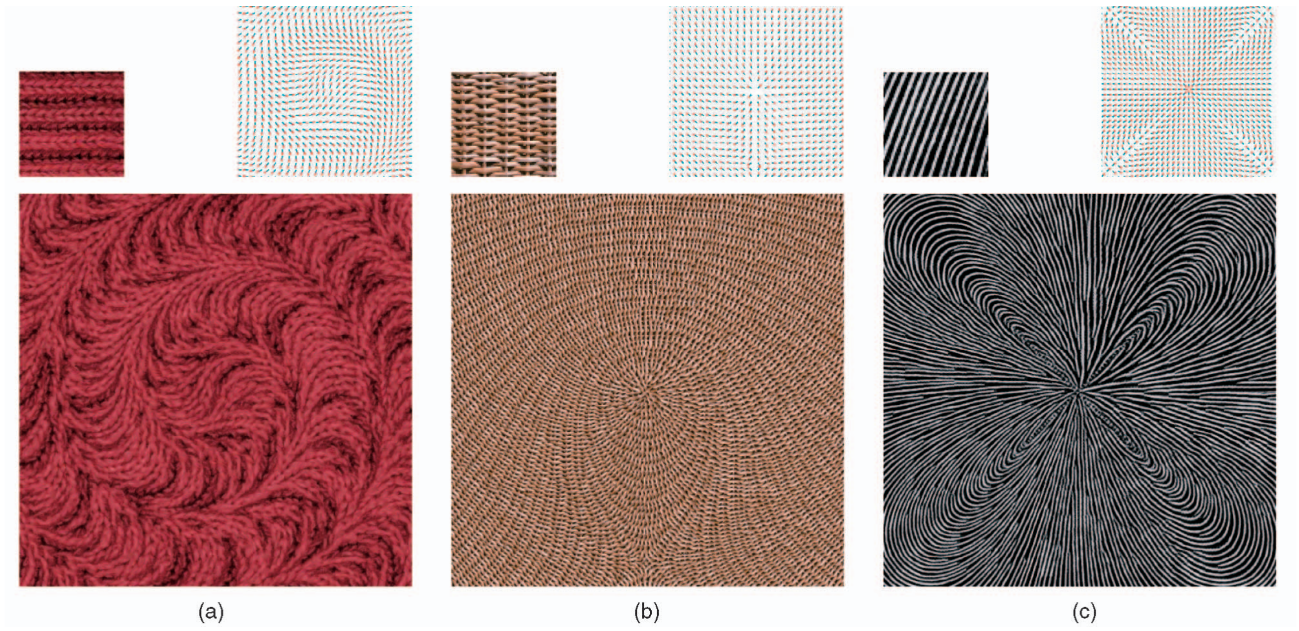

Fig. 14. Directional texture synthesis for vector field visualization. (a) A vector field visualized with knitwear texture (size 512×512, generated in 15 seconds). (b) A vector field visualized with cane texture (size 512×512, generated in 15 seconds). (c) A vector field visualized with engraving texture (size 512×512, generated in 15 seconds).

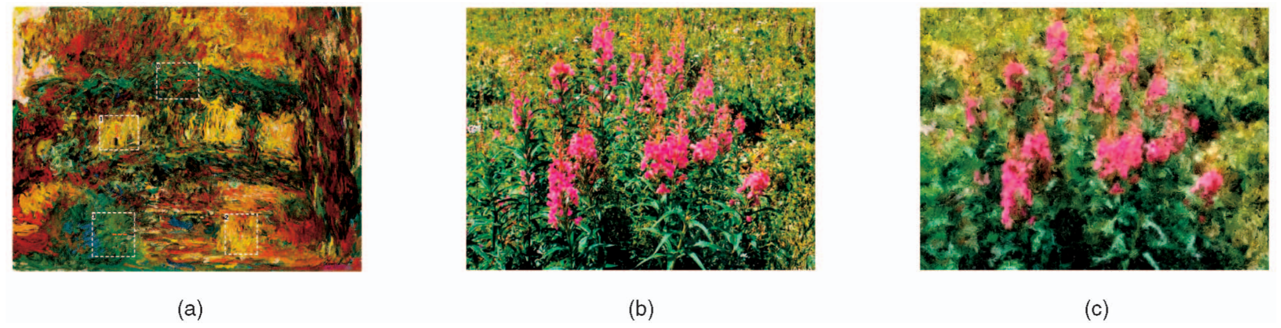

Fig. 15. A failure example. (a) Monet's *The Japanese Bridge*. (b) A source image. (c) An unsuccessful synthesis result.

interactive design of digital media. This application is further enhanced by the flexible user control in selecting the style of brush strokes and defining brush stroke directions.

Our method is not fully automatic since some user input and control are needed, such as selecting and assigning sample textures (i.e., painting styles). Fortunately, these tasks can be easily performed with a few mouse clicks and they provide the user with the flexibility to select different styles that might be found in the same or different example paintings and use them to synthesize the synthesized painting.

Apparently, our method does not require, as part of the input, the original photo A of the example painting A' as in the IA method. However, the texture samples selected by the user from A' in our method provide similar information content to that by the original photo A in the IA method for searching for the best-matching patch or pixel. A notable difference is that the distance between a square neighborhood in A and a square neighborhood in B is a critical term in the error metric used in the IA method for searching for the best pixel candidate, while our method does not involve such a term but still produces satisfactory results.

We also showed how our method can be used to synthesize a 2D directional texture following a given directional field. A possible application of the method is thus in the area of vector field visualization. For example, our algorithm may be useful for placing long streamlines [36], as demonstrated in Fig. 14.

Further research is needed to address the limitations of our method. Our method synthesizes a painting with the brush directions being either fixed or being determined by the media axes of segmented regions. However, artists may use other brush directions that are not related to the medial axis. Thus, brush directions should also be considered as an important aspect of a painting style and should be learned from an example painting as well. Apparently, learning the brush direction as part of a style is much harder than simply capturing and synthesizing the stroke texture. Another related challenging task is creating scalable strokes to represent objects, as discussed in [17].

Acknowledgments

The authors thank Prof. Myung-Soo Kim, Mr. Bin Chan, and Ms. Loretta Choi for their great help in this work. They also thank the referees for their constructive comments. They acknowledge the support of the Hi-Tech Research and Development Program of China under grant 2003AA4Z1010 and the support of the National Key Basic Research Project of China under grant 2002CB312106.

REFERENCES

[1] M. Ashikhmin, "Synthesizing Natural Textures," *Proc. 2001 Symp. Interactive 3D Graphics,* J.F. Hughes and C.H. Sequin, eds., pp. 217-226, 2001.

[2] J.S.D. Bonet, "Multiresolution Sampling Procedure for Analysis and Synthesis of Texture Images," *Proc. SIGGRAPH 1997,* pp. 361-368, 1997.

[3] P.J. Burt and E.H. Adelson, "The Laplacian Pyramid as a Compact Image Code," *IEEE Trans. Comm.,* vol. 31, no. 4, pp. 532-540, 1983.

[4] D. Comaniciu and P. Meer, "Mean Shift: A Robust Approach toward Feature Space Analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 5, pp. 603-619, May 2002.

[5] C.J. Curtis, S.E. Anderson, J.E. Seims, K.W. Fleischer, and D.H. Salesin, "Computer-Generated Watercolor," *Proc. SIGGRAPH 1997,* pp. 421-430, 1997.

[6] P.E. Danielsson, "Euclidean Distance Mapping," *Computer Graphics and Image Processing,* vol. 14, pp. 227-248, 1980.

[7] A. Efros and W.T. Freeman, "Image Quilting for Texture Synthesis and Transfer," *Proc. SIGGRAPH 2001,* pp. 341-346, 2001.

[8] A. Efros and T. Leung, "Texture Synthesis by Non-Parametric Sampling," *Proc. IEEE Int'l Conf. Coupter Vision (ICCV '99),* pp. 1033-1038, 1999.

[9] W.T. Freeman, J.B. Tenenbaum, and E.C. Pasztor, "An Example-Based Approach to Style Translation for Line Drawings," *ACM Trans. Graphics,* vol. 22, no. 1, pp. 33-46, 2003.

[10] B. Gooch and A. Gooch, *Non-Photorealistic Rendering.* A.K. Peters Ltd., 2001.

[11] B. Gooch, G. Coombe, and P. Shirley, "Artistic Vision: Painterly Rendering Using Computer Vision Techniques," *Proc. Int'l Symp. Non-Photorealistic Animation and Rendering (NPAR 2002),* 2002.

[12] P. Haeberli, "Paint by Numbers: Abstract Image Representations," *Proc. SIGGRAPH 1990,* pp. 207-214, 1990.

[13] J. Hamel and T. Strothotte, "Capturing and Re-Using Rendition Styles for Non-Photorealsitic Rendering," *Computer Graphics Forum (Eurographics '99),* vol. 18, no. 3, pp. 173-182, 1999.

[14] P. Harrison, "A Non-Hierarchical Procedure for Re-Synthesis of Complex Textures," *Proc. Int'l Conf. in Central Europe Computer Graphics and Visualization (WSCG 2001)* pp. 190-197, 2001.

[15] A. Hausner, "Simulating Decorative Mosaics," *Proc. SIGGRAPH 2001,* pp. 573-580, 2001.

[16] D.J. Heeger and J.R. Bergen, "Pyramid Based Texture Analysis/Synthesis," *Proc. SIGGRAPH 1995,* pp. 229-238, 1995.

[17] A. Hertzmann, "Painterly Rendering with Curved Brush," *Proc. SIGGRAPH 1998,* pp. 453-460, 1998.

[18] A. Hertzmann, C.E. Jacobs, N. Oliver, B. Curless, and D.H. Salesin, "Image Analogies," *Proc. SIGGRAPH 2001,* pp. 327-340, 2001.

[19] A. Hertzmann, N. Oliver, B. Curless, and S.M. Seitz, "Curve Analogies," *Proc. 13th Eurographics Workshop Rendering,* 2002.

[20] R. Jain, R. Kasturi, and B. Schunck, *Machine Vision.* McGraw-Hill, 1995.

[21] A.K. Jain, *Fundamentals of Digital Image Processing.* Prentice-Hall, 1989.

[22] P.M. Jodoin, E. Epstein, M. Granger-Piche, and V. Ostromoukhov, "Hatching by Example: A Statistical Approach," *Proc. Int'l Symp. Non-Photorealistic Animation and Rendering (NPAR 2002),* 2002.

[23] R.D. Kalnins et al., "WYSIWYG NPR: Drawing Strokes Directly on 3D Models," *ACM Trans. Graphics (Siggraph 2002),* vol. 21, no. 3, pp. 755-762, 2002.

[24] J.-G. Leu, "Computing a Shape's Moments from Its Boundary," *Pattern Recognition,* vol. 24, no. 10, pp. 949-957, 1991.

[25] L. Liang, C. Liu, Y.-Q. Xu, B. Guo, and H.-Y. Shum, "Real-Time Texture Synthesis by Patch-Based Sampling," *ACM Trans. Graphics,* vol. 20, no. 3, pp. 127-150, 2001.

[26] F. Petrie and J. Shaw, *The Big Book of Painting Nature in Watercolor.* Watson-Guptill, 1990.

[27] E. Praun, A. Finkelstein, and H. Hoppe, "Lapped Textures," *Proc. SIGGRAPH 2000,* pp. 465-470, 2000.

[28] J.C. Russ, *The Image Processing Handbook.* CRC Press, 1995.

[29] M.P. Salisbury, M.T. Wong, J.F. Hughes, and D.H. Salesin, "Orientable Textures for Image-Based Pen-and-Ink Illustration," *Proc. SIGGRAPH 1997,* pp. 401-406, 1997.

[30] S.A. Schaeffer and J. Shaw, *The Big Book of Painting Nature in Pastel.* Watson-Guptill, 1993.

[31] M. Shiraishi and Y. Yamaguchi, "An Algorithm for Automatic Painterly Rendering Based on Local Source Image Approximation," *Proc. Int'l Symp. Non-Photorealistic Animation and Rendering (NPAR 2000),* pp. 53-58, 2000.

[32] C. Soler, M.-P. Cani, and A. Angelidis, "Hierarchical Pattern Mapping," *ACM Trans. Graphics (Siggraph 2002),* vol. 21, no. 3, pp. 673-680, 2002.

[33] S. Strassmann, "Hairy Brushes," *Proc. SIGGRAPH 1986,* pp. 225-232, 1986.

[34] H. Tamura, S. Mori, and T. Yamawaki, "Textural Features Corresponding to Visual Perception," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 8, pp. 460-472, 1978.

[35] X. Tong, J. Zhang, L. Liu, X. Wang, B. Guo, and H.-Y. Shum, "Synthesis of Bidirectional Texture Functions on Arbitrary Surfaces," *ACM Trans. Graphics (Siggraph 2002),* vol. 21, no. 3, pp. 665-672, 2002.

[36] G. Turk and D. Banks, "Image-Guided Streamline Placement," *Proc. SIGGRAPH 1996,* pp. 453-460, 1996.

[37] G. Turk, "Texture Synthesis on Surfaces," *Proc. SIGGRAPH 2001,* pp. 347-354, 2001.

[38] L.-Y. Wei and M. Levoy, "Fast Texture Synthesis Using Tree-Structured Vector Quantization," *Proc. SIGGRAPH 2000,* pp. 479-488, 2000.

[39] L.-Y. Wei and M. Levoy, "Texture Synthesis over Arbitrary Manifold Surfaces," *Proc. SIGGRAPH 2001,* pp. 355-360, 2001.

[40] Y.-Q. Xu, B. Guo, and H.-Y. Shum, "Chaos Mosaic: Fast and Memory Efficient Texture Synthesis," Technical Report 32, Microsoft Research Asia, 2000.

[41] L.-X. Ying, A. Hertzmann, H. Biermann, and D. Zorin, "Texture and Shape Synthesis on Surfaces," *Proc. 12th Eurographics Workshop Rendering,* pp. 301-312, 2001.

Bin Wang is a PhD candidate in the Department of Computer Science and Technology, Tsinghua University, Beijing, People's Republic of China. He received the BS degree in chemistry from Tsinghua University in 1999. From November 2001 to April 2003, he worked as a research assistant in the Computer Graphics Laboratory of the University of Hong Kong. His current research interests include computer graphics and computer-aided design.

Wenping Wang received the BSc and MEng degrees in computer science from Shandong University, China, in 1983 and 1986, respectively, and the PhD degree in computer science from University of Alberta, Canada, in 1992. He is an associate professor of computer science at the University of Hong Kong, China. His research interests include computer graphics, geometric computing, and computational geometry (http://www.csis.hku.hk/~wenping/).

Huaiping Yang received the BE degree in hydraulic engineering from Tsinghua University. He is a PhD student in the Department of Computer Science and Technology, Tsinghua University, Beijing, People's Republic of China. From November 2001 to April 2003, he worked as a research assistant at the Computer Graphics Laboratory of the University of Hong Kong. His current research interests include computer-aided design and computer graphics.

Jiaguang Sun received the BS degree in computer science from Tsinghua University in 1970. He was a visiting scholar at the University of California at Los Angeles from 1985 to 1986. He is a professor in the Department of Computer Science and Technology at Tsinghua University. He is also director of the National CAD Engineering Center at Tsinghua University and Academician of the Chinese Academy of Engineering. His current research interests include computer-aided geometric design, computer graphics, and software engineering.