Optimized Synthesis of Art Patterns and Layered Textures

Ruobing Wu, Wenping Wang, and Yizhou Yu

Abstract—Line drawings and digital arts appear everywhere, from simple icons and logos to cartoons, maps, and illustrations. We define art patterns as the subset of line drawings and digital arts that are comprised of repeated elements. There exist textures that share characteristics with art patterns. Examples of such textures include piled discrete elements with curved contours. Inspired by recent success of exemplar-based texture synthesis, in this paper, we focus on synthesizing art patterns and textures with curvilinear features from exemplars, which we cast as a global optimization problem. Our energy function for this problem measures both the appearance similarity of color patterns and shape similarity of curvilinear features between an input exemplar and a synthesized image. We develop an overall expectation-maximization-style algorithm for minimizing this energy function. The shape similarity part of the energy is minimized through an innovative application of the level set method. We further generalize our energy function and optimization algorithm to multilayer pattern and texture synthesis. Our generalized optimization can effectively handle multiple layers and synthesize valid instances of interaction.

Index Terms—Texture synthesis, level set method, line drawing, digital arts, multilayer synthesis

1 Introduction

Line drawings and digital arts appear everywhere, from simple icons and logos to cartoons, maps, illustrations, and storyboards. They are also used for various decoration purposes. The creation of line drawings and digital arts is typically a time-consuming process that requires skilled artists. Most of us appreciate their simple and aesthetic appearances, but do not have the necessary skills to create them. There exist a subset of line drawings and digital arts that exhibit the essence of textures, i.e., a spatial arrangement of repeated elements. We use the phrase, *art patterns*, to call both such line drawings and digital arts.

Art patterns have their unique characteristics. They are typically dominated by curves and shape contours. Spaces among the curves are typically filled with constant colors, color gradients, or a smooth diffusion of a sparse set of colors. In fact, there exist textures that share characteristics with art patterns. Examples of such textures include piled discrete elements with curved contours, such as piles of candies and leaves in Figs. 5a and 7a. In this paper, we would like to investigate techniques for synthesizing both art patterns and textures with curvilinear features from exemplars.

Synthesizing the aforementioned art patterns and textures is challenging for the following reasons. First, thin and elongated curves in an art pattern or texture are not compatible with rectangular patches and neighborhoods often found in recent exemplar-based texture synthesis literature. Second, given the relatively wide span of a curve, local synthesis is unlikely to work well, often producing fragmented appearances. And it is also unclear how to place this synthesis task in a global optimization context. Third, elements in an exemplar may have mutual interactions, giving rise to layers and occlusions. It is unclear what mechanisms need to be in place to create valid instances of interactions in a synthesized image. Synthesizing high-quality novel instances of layered textures and patterns remains a hard problem.

In this paper, we present an exemplar-based synthesis framework for art patterns and layered textures. It is based on the level set method, a popular numerical technique for tracking and optimizing curves and contours. The level set method embeds a closed curve as the zero level set of a level set function whose domain covers the entire 2D image space, and performs curve tracking indirectly through the level set function. It, thus, becomes possible to perform patch-based synthesis over the level set function. Furthermore, the level set method offers a global optimization capability for our image synthesis task. As long as a global energy function as well as its derivative with respect to level sets is well defined, this energy function can be minimized through the level set method. To enable effective synthesis of complex scenarios, such as mutual dependence and interactions among image elements, we make use of multiple level set functions, each of which is assigned a distinct layer, and further generalize our synthesis framework to accommodate multiple layers of level set functions.

We summarize our contributions in this paper as follows:

- Inspired by existing work that casts exemplar-based synthesis as a global optimization problem, we formulate a new global energy function for synthesizing art patterns and layered textures that have many thin curvilinear features. This energy function measures both the appearance similarity of color patterns and shape similarity of curvilinear features between an input exemplar and a synthesized output.

- *The authors are with the Department of Computer Science, The University of Hong Kong, Pokfulam Road, Hong Kong.*

Manuscript received 8 July 2012; revised 14 Mar. 2013; accepted 5 Aug. 2013; published online 13 Aug. 2013.
Recommended for acceptance by G. Drettakis.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2012-07-0127.
Digital Object Identifier no. 10.1109/TVCG.2013.113.

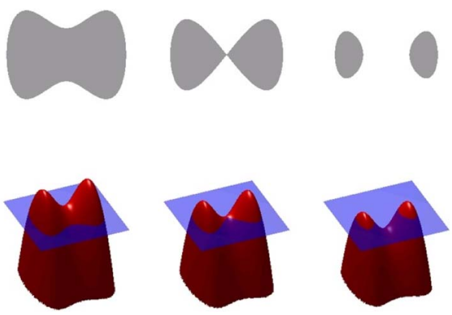


Fig. 1. A 2.5D illustration from Wikipedia [4] for the level set method. The bottom row shows that it is simply the height of the level set function that becomes lower when the 2D shape in the top row becomes disconnected.

- We develop an expectation-maximization (EM)-style optimization technique for the aforementioned global energy function. The shape similarity part of the energy is minimized through a novel and systematic level set-based algorithm, which iteratively moves the curvilinear features in the synthesized image toward nearby features in the best matching neighborhoods from the input exemplar. This algorithm maintains the integrity of every curvilinear feature during optimization even though the matching neighborhoods only contain truncated features.
- We further generalize the above global energy function and level set-based optimization to multilayer pattern and texture synthesis. Multiple layers of signed distance functions are utilized for representing the complete set of curvilinear features. Our generalized optimization can effectively handle these signed distance function (SDF) layers and synthesize valid instances of mutual dependence and interactions, such as occlusions.

2 Background and Related Work

2.1 Level Set Method

The level set method is a numerical technique for tracking and transforming dynamic surfaces and shapes. Introduction to the level set method and its applications can be found in [1], [2], [3].

In the level set method, surfaces in a fixed Cartesian grid are described using a signed distance function Φ, called the level set function. This function returns the signed distance of a point to the surface, when given its position. The surface separates the interior and exterior of an object on the grid; therefore, it is often referred to as the "interface." Fig. 1 shows a 2.5D illustration. In a two-dimensional space, the level set method amounts to representing a closed curve Γ using the level set function Φ. At any time t, the curve Γ is represented as the zero level set of Φ,

$$\Gamma(t) = \{\mathbf{x} \in R^2 \mid \Phi(\mathbf{x}, t) = 0\}. \quad (1)$$

Here, the level set function Φ typically takes negative values inside the region delimited by the closed curve Γ and positive values outside.

The evolution of the level set function follows

$$\Phi_t + \mathbf{V} \cdot \nabla\Phi = 0, \quad (2)$$

where Φ_t is the partial derivative of Φ with respect to time, and $\mathbf{V}$ is a velocity field governing the advection of the level set function over time. Note that $\mathbf{V}$ also implicitly controls the evolution of Γ because Γ is embedded as the zero level set. $\mathbf{V} \cdot \nabla\Phi$ can also be expressed as $v\|\nabla\Phi\|$, where v is the magnitude of the velocity field in the normal direction of the passing level set. The level set method allows one to manipulate Γ and perform numerical computations involving closed curves and surfaces without having to parameterize them. Because of the embedding of Γ in the level set function, topological changes of Γ can be handled implicitly without special considerations.

Thus, the level set method facilitates the representation of deformable implicit curves and surfaces as well as offers a robust numerical method for dealing with the geometric and topological evolution of shapes and closed curves. Our novel technique proposed in this paper is meant to take advantage of these properties of the level set method.

2.2 Related Work

There exists much work on developing computer algorithms for generating floral ornaments [5] and planar patterns [6], [7]. Such algorithms do not need exemplars. They can generate patterns following either mathematical (geometric) principles or design principles observed by human artists. Since we take an exemplar-based approach in this paper, in the following, we focus on previous work related to exemplar-based texture synthesis.

2.2.1 *Pixel-Based Synthesis*

Among the pixel-based region-growing techniques, Garber [8] and Efros and Leung [9] define the basic pixel-based scheme by synthesizing in scanline order to find and copy the pixels with the most similar local neighborhood in the input exemplar. Improvements include faster hierarchical synthesis [10], [11], [12], coherent synthesis [13], real-time parallel synthesis on GPUs [14], and multiscale synthesis [15], [16]. Pixel-based techniques [17], [18], [19] are by nature suitable for controllable texture synthesis because they have direct control over fine-grain pixel values. Both color and signed distance have been utilized for neighborhood matching in [19]. Nevertheless, signed distance values are used as an additional cue in neighborhood comparison, but not as variables in a global optimization with a quantitative objective function. There exists work on generalizing texture synthesis to curves [20], which only performs individual curve synthesis, and is not applicable to a 2D arrangement of curves and contours in an art pattern.

2.2.2 *Patch-Based Synthesis*

Meanwhile, patch-based techniques [21], [22], [23], [24] copy and stitch together texture patches from the input exemplar at various offsets. They are more successful in generating high-quality synthesis results because they retain patchwise spatial arrangements from the input exemplar. Among them, "Image Quilting" [22] takes a novel minimum-error boundary cut approach based on dynamic programming, while Kwatra et al. [24] compute optimal seams using a graphcut algorithm. Wu and Yu [25] define a feature map from the input exemplar and perform

Fig. 2. Closed feature curves of an input exemplar and their signed distance function. Our synthesis result of this example can be found at the top of the middle column in Fig. 9.

feature matching and alignment by measuring structural similarity. However, unlike our level set-based method, these techniques do not attempt to globally optimize the quality of the synthesized curvilinear features and contours. In addition, they typically cannot handle textures with multiple layers very well.

2.2.3 Optimization-Based Synthesis

Global optimization has become popular in texture synthesis recently. Such a method synthesizes results by optimizing a global energy function. Kwatra et al. [26] present an EM-style synthesis-by-optimization scheme. Their technique gives robust performance and fully supports constrained synthesis. More recent work includes [27], which integrates nonparametric optimization with histogram matching for synthesizing 3D texture solids, [28], which runs in the opposite direction of traditional forward synthesis and generates a small texture compaction to best summarize the original variations of a large input texture, [29], which uses control maps that are similar to SDFs in their optimization and supports multilayer textures through interactive layer definition, and [30], which provides a scheme for synthesizing repetitive discrete elements within a given large-scale structure. Our technique is a natural generalization of Kwatra et al. [26] in terms of global optimization. By applying the level set method in the optimization stage and involving layer information in the similarity metric, our technique achieves more accurate patch matching and better curvilinear feature alignment for both art patterns and layered textures.

3 Basic Algorithm

The energy function in our global optimization consists of two parts. The first part evaluates the local appearance similarity between the synthesized image and the input exemplar. The second part of our energy function measures the local similarity between signed distance transforms of curvilinear features in both the synthesized image and the input exemplar. One of our main goals is to measure the local shape similarity between these curvilinear features in the input and output. Since every set of features uniquely define a signed distance function, computing local shape similarity can be cast as computing the local similarity between two signed distance functions [31]. Note that, as shown in Fig. 2, open features, which are obtained with Canny edge-detection [32], need to be connected together manually to form closed curves before a signed distance function can be computed, especially near image boundaries. We typically connect no more than 20 open features among the detected edges in the input exemplar. The SDF value of a pixel is calculated according to the distance to its nearest closed feature.

Let X denote the synthesized output and Z denote the input exemplar. Let $\mathbf{A}_\mathbf{x}$ (or $\mathbf{A}_\mathbf{z}$) denote the appearance vector at pixel $\mathbf{x}$ (or $\mathbf{z}$) in X (or Z). In this paper, we define this appearance vector as the concatenated pixel colors within a local neighborhood centered at pixel $\mathbf{x}$ (or $\mathbf{z}$). We further use $\mathbf{D}_\mathbf{x}$ (or $\mathbf{D}_\mathbf{z}$) to denote the vector of concatenated signed distance values within a neighborhood centered at pixel $\mathbf{x}$ (or $\mathbf{z}$). Let $\mathbf{z}_\mathbf{x}$ denote the center of the neighborhood from the input exemplar, which is most similar to the neighborhood centered at $\mathbf{x}$ in X in terms of both color and signed distance under the euclidean norm. The total energy is defined with respect to a collection of neighborhoods S_X from X:

$$E_t(S_X) = \alpha \sum_{\mathbf{x}\in S_X} \|\mathbf{A}_\mathbf{x} - \mathbf{A}_{\mathbf{z}_\mathbf{x}}\|^2 + (1-\alpha) \sum_{\mathbf{x}\in S_X} \|\mathbf{D}_\mathbf{x} - \mathbf{D}_{\mathbf{z}_\mathbf{x}}\|^2, \tag{3}$$

where $\alpha \in [0, 1)$ is a weight balancing the effects of color and signed distance similarity. For black and white line drawing inputs, α is set to 0 because only signed distance values contribute to the quality of synthesized results. Let the size of a neighborhood be $w \times w$. In practice, the energy is evaluated over a subset of local neighborhoods that are $w/4$ pixels apart because it is redundant and computationally expensive to compute the energy over all neighborhoods in the synthesized image.

The energy function in (3) can be optimized in a way similar to expectation-maximization [26], [33], where $(\{\mathbf{A}_x\}, \{\mathbf{D}_x\})$ and $\{\mathbf{z}_x\}$ are alternatively optimized in the maximization (M)-step and expectation (E)-step, respectively. The EM algorithm is an iterative method for finding maximum likelihood or maximum a posteriori (MAP) estimates of unobserved latent variables in statistical models. Each EM iteration performs an E-step, which creates a function for the expectation of the log-likelihood evaluated using the current estimate of the probability density, and an M-step, which computes updated values of the latent variables maximizing the expected log-likelihood found in the E-step. Similar to Kwatra et al. [26], we perform an EM-like optimization by computing synthesized texture patches in the E-step while locating the set of input neighborhoods most similar to the synthesized patches in the M-step. In theory, due to acceleration schemes used for nearest neighbor search, our M-step cannot guarantee to find the truly optimal neighborhoods in the input exemplar. However, in practice, the located input neighborhoods are not too different from those optimal ones, and a few iterations of our E-step and M-step can always reduce the energy by a large amount even if the process does not converge at the end.

At the very beginning prior to any M-steps and E-steps in our framework, we initialize the pixelwise appearance and signed distance values in the synthesized image using "image quilting" [22]. The only deviation from the original quilting algorithm is that every patch from the input exemplar is padded with an extra signed distance channel. We further extract an initial zero level set from the synthesized signed distance channel.

The M-step. In the M-step, we search for the best matching neighborhoods in the input exemplar. That is, in (3), we fix $\{\mathbf{A}_x\}$ and $\{\mathbf{D}_x\}$ while updating $\{\mathbf{z}_x\}$ for all neighborhoods in S_X. The similarity metric for matching is based on summed squared differences (SSD) and includes both terms in (3), color and signed distance. The search hierarchy is constructed using the Approximate Nearest Neighbor (ANN) [34] algorithm and PatchMatch [35] for speedup.

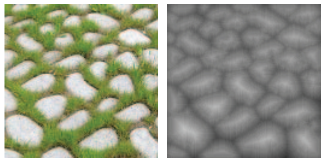
The E-step. At the beginning of every E-step, there already exist an initial appearance map (color image) and signed distance function for the output image, and every neighborhood in S_X has already been assigned a best matching neighborhood from the input exemplar during the latest M-step. During the E-step, we optimize both pixelwise appearance values and signed distance values. That is, in (3), we fix $\{\mathbf{z}_x\}$ while updating $\{\mathbf{A}_x\}$ and $\{\mathbf{D}_x\}$. Since our input exemplars are mostly line drawings and art patterns that have many curvilinear features, producing high-quality geometric patterns of such features is of higher priority than color patterns. Therefore, we optimize signed distance values and appearance values in two sequential stages, and signed distance values are optimized first. In the following, we elaborate our algorithm for optimizing signed distance values. We simply follow the technique in [26] to optimize appearance values.

The goal of signed distance optimization is to reduce the second energy term in (3). On average, this makes the signed distance values from the neighborhoods in the synthesized image closer to the signed distance values from their best matching neighborhoods in the input exemplar. However, this optimization is nontrivial in the sense that we cannot alter the signed distance value at every pixel independently to reduce the energy term because these signed distance values are derived from a signed distance transform of a set of closed curvilinear features. No matter how we alter the signed distance values, they have to remain a valid signed distance transform of some closed curves. That means the shape of these closed curves are the only degrees of freedom that we can manipulate during the optimization. Now the problem becomes how to deform the curvilinear features in the synthesized image so that their signed distance transform approaches the expectation of the signed distance values from the best matching neighborhoods in the input exemplar.

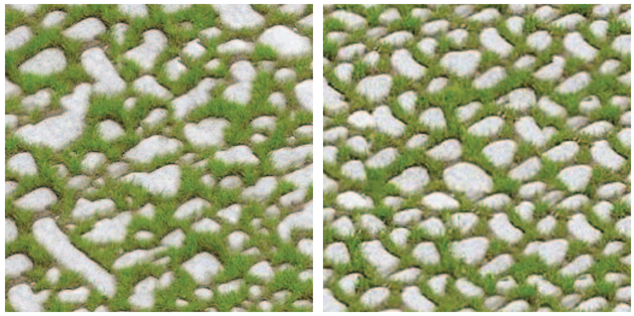
This problem can be conveniently solved using the level set method. Since the neighborhoods in S_X are overlapping, every pixel in the synthesized image is actually covered by multiple best matching neighborhoods from the input exemplar. If we break every neighborhood into a set of pixels, the second energy term in (3) can be rewritten as

$$\sum_{\mathbf{x}\in X}\sum_{j\in s_{\mathbf{x}}}\left(\Phi^o(\mathbf{x})-\Phi^i_j(\mathbf{x})\right)^2, \tag{4}$$

where Φ^o is the signed distance function for the output image that we need to optimize, $s_{\mathbf{x}}$ is the set of neighborhoods (from the input exemplar) covering pixel $\mathbf{x}$, and $\{\Phi^i_j | j\in s_{\mathbf{x}}\}$ represent the set of signed distance functions associated with the neighborhoods in $s_{\mathbf{x}}$. Note that Φ^o is also the level set function we use in our level set algorithm.



(a) Input and the initial SDF



(b) Without curvature term (c) With curvature term

Fig. 3. The curvature-based term helps remove irregularities along synthesized shape boundaries ($\alpha = 0.4$).

To derive the velocity field for the level set method, we need to redefine the energy term in (4) in a continuous image domain, where the gradient of Φ^o is well defined, as follows:

$$E_s(X)=\int_X\sum_{j\in s_{\mathbf{x}}}\left(\Phi^o(\mathbf{x})-\Phi^i_j(\mathbf{x})\right)^2 d\mathbf{x}. \tag{5}$$

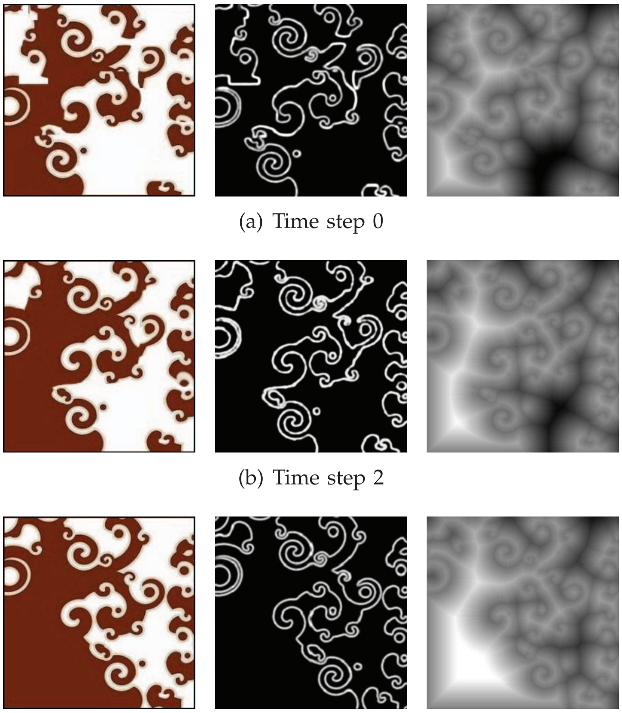
Taking the derivative of this energy with respect to the level set passing through $\mathbf{x}$, we obtain the following velocity component:

$$\mathbf{V}_s=2\sum_{j\in s_{\mathbf{x}}}\left(\Phi^o(\mathbf{x})-\Phi^i_j(\mathbf{x})\right)\frac{\nabla\Phi^o(\mathbf{x})}{\|\Phi^o(\mathbf{x})\|}. \tag{6}$$

This equation indicates that the total "force" to deform and optimize the zero level set of Φ^o is a weighted average of the "forces" from individual overlapping neighborhoods, and the magnitude of the "force" from the jth overlapping neighborhood should be set to $2(\Phi^o(\mathbf{x})-\Phi^i_j(\mathbf{x}))$, which can be either positive or negative. When the weight is positive (negative), the zero level set of Φ^o will be pushed outward (inward) to the zero level set of Φ^i_j. Note that the gradient of the level set function can be estimated robustly using the upwind scheme [36].

Although neighborhoods in $s_{\mathbf{x}}$ are overlapping, curvilinear features therein do not necessarily align with each other very well. Minimizing the energy term in (4) alone may give rise to undesired discontinuities in the zero level set of Φ^o. To make the zero level set of Φ^o sufficiently continuous and smooth while minimizing the energy in (4), it is a common practice to add a curvature-based velocity component, which achieves similar effects of a regularization term. As shown in Fig. 3, this curvature-based term helps improve the shape of synthesized boundaries. Thus, the overall velocity field is defined as

$$V=\lambda V_s+\mu V_c, \tag{7}$$



(a) Time step 0

(b) Time step 2

(c) Time step 10

Fig. 4. The evolution of the zero level set over multiple time steps ($\alpha = 0$).

where $\lambda, \mu \in [0, 1]$ are weights of the two velocity components, V_s is defined in (6), and $V_c = \kappa(\mathbf{x})\frac{\nabla\Phi^o(\mathbf{x})}{\|\nabla\Phi^o(\mathbf{x})\|}$, where $\kappa(\mathbf{x})$ is defined as $(\Phi^o_{xx}\Phi^{o2}_y - 2\Phi^o_x\Phi^o_y\Phi^o_{xy} + \Phi^o_{yy}\Phi^{o2}_x)/(\Phi^{o2}_x + \Phi^{o2}_y)^{\frac{3}{2}}$, and denotes the curvature of the level set passing through $\mathbf{x}$ [3].

Intuitively, our level set-based optimization tries to instill a small amount of distortion into the zero level set of Φ^o so that it maintains a certain level of continuity, while its signed distance transform is still locally much similar to the signed distance functions associated with the neighborhoods from the input exemplar.

During every E-step, we run the level set method multiple time steps until it converges. During each time step, we follow (7) to compute an overall velocity field, which is used for driving the zero level set. In practice, we use the narrow-band method [37] for acceleration. Fig. 4 shows the evolution of the zero level set and the level set function over multiple time steps. It can be observed that misalignments and unnatural local shapes in the initial zero level set gradually disappear as the level set-based optimization goes on.

When the zero level set of Φ^o is being deformed and optimized, the entire synthesized image should be deformed consistently. To address this issue, at the end of each level set time step, we generate a deformation vector field for the entire image from the velocity vectors within a narrow band around the zero level set of Φ^o using the image morphing technique in [38]. This deformation field is then used for warping the colors inside the entire synthesized image. The subsequent color optimization stage is actually performed over the warped image.

Let us now discuss the influence of weight α in (3) on synthesis quality. When the input exemplar is a color or grayscale image, α is typically set to a medium value between 0.4 and 0.6 to ensure both color and SDF channels are treated in a balanced manner. When the input is black-and-white, α is set to a small value between 0 and 0.1 because the SDF term is the one that really matters. Fig. 5 shows a series of results with different choices of α for a color input exemplar. A small α overemphasizes the SDF channel but neglects the color channels. As a result, texture elements with matching boundaries but different colors become blended together. A large α, on the other hand, overemphasizes the color channels but neglects the SDF channel, and gives rise to incomplete texture elements with broken boundaries. Parameter in the range of [0.45, 0.55] will robustly give reasonable results, as shown in Figs. 5c and 5e.

To demonstrate the necessity of level set propagation, we have compared our synthesis algorithm with a simplified version. In this simpler version, the input and synthesized images still have an SDF channel in addition to the color channels. Instead of level set propagation, it largely follows the E-step in [26] except, at the end of every E-step, it enforces the accuracy of the SDF channel by first extracting the zero crossings of the SDF channel of the synthesized image and then recomputing an SDF for these extracted zero crossings. Comparison results can be found in Fig. 6, where it can be verified that level set propagation is indeed crucial for maintaining the integrity of texture elements.

3.1 Multiresolution Scheme

Intuitively, large-scale features and their spatial layout should be synthesized first followed by the synthesis of smaller scale features and details. Wei and Levoy [39] first introduced such a pyramidal framework to achieve order-independent synthesis. The optimization process in our framework has been implemented following a similar multiresolution scheme. We first synthesize an output image at a coarse resolution and then repeatedly perform upsampling

(a) Input (b) $\alpha = 0.1$ (c) $\alpha = 0.45$ (d) $\alpha = 0.5$ (e) $\alpha = 0.55$ (f) $\alpha = 0.9$

Fig. 5. The influence of weight α on synthesis quality. Artifacts are marked with blue circles in (b) and (f).

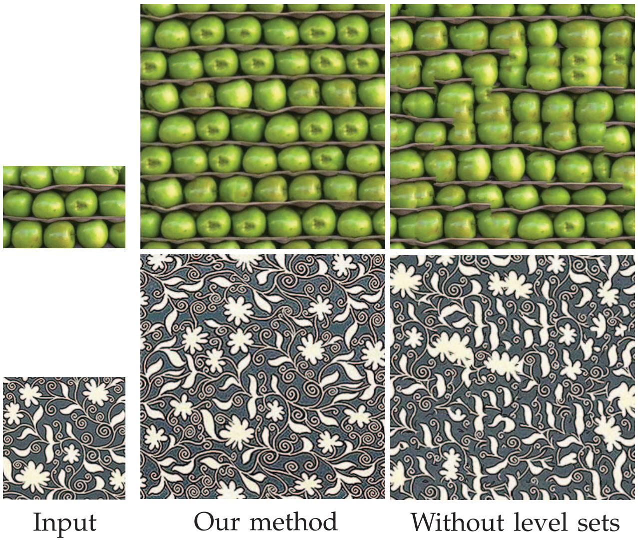


Fig. 6. Comparison between our method and a simplified version that does not support level set propagation (Top: $\alpha = 0.2$. Bottom: $\alpha = 0$).

and optimization. Upsampling to a higher resolution is implemented via interpolation, which is followed by multiple EM iterations at that resolution. Level set reinitialization is performed right after each interpolation.

During the iterative EM-style optimization at each resolution, the size of image neighborhoods starts from 128×128 and is halved in both dimensions every iteration until it reaches 8×8. Meanwhile, the weights in (7), λ and μ, are also automatically adjusted at each resolution for high-quality outputs. At first, μ is set to 0.8 to encourage a high degree of smoothness while λ is set to 0.2. In later EM iterations on the same level, λ gradually increases (no larger than 0.95) to allow better localization of the zero level set of the synthesized image, while μ decreases (no less than 0.05).

4 Multilayer Synthesis

In the previous section, we have introduced an image synthesis algorithm that relies on a signed distance function to optimally synthesize the shape of curvilinear features. By doing so, we need to model curvilinear features as closed curves. An important limitation of a single SDF is that such closed curves must divide an image into two colorable regions. There exist many scenarios where we cannot model the features using a single SDF. For example, if an image consists of elements on multiple layers and there exists occlusion between elements on different layers, a single SDF is insufficient to simultaneously delineate the contours of all elements. In another scenario, three or more image elements could be abutting each other. A single SDF is also insufficient to simultaneously represent the contours of all abutting elements.

This limitation can be overcome by increasing the number of SDFs. For example, two SDFs can already divide an image into four colorable regions. Thus, multiple SDFs have the capability of representing nonmanifold structures, where three or more regions are simultaneously adjacent to each other. In practice, we interactively define multiple layers according to the edge detection results in the input exemplar to generate multiple SDFs.

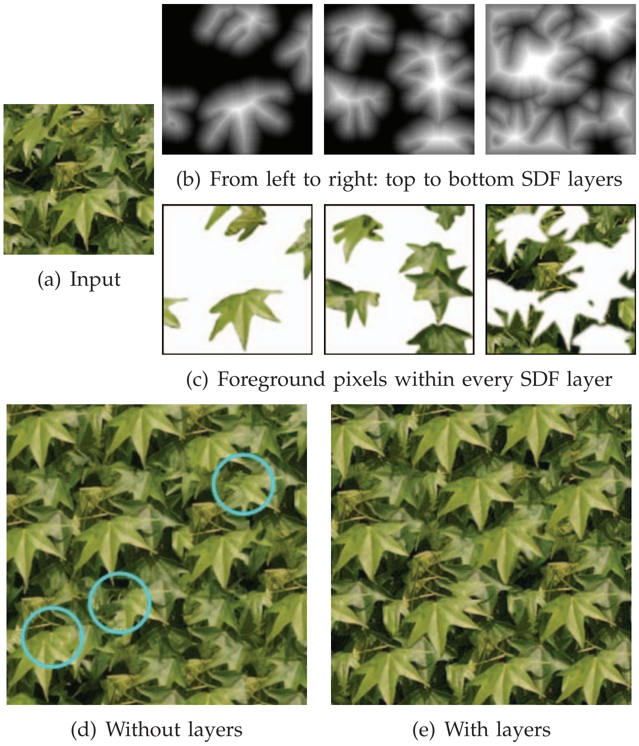


Fig. 7. (a) Input. (b) and (c) SDF Layers. Top to bottom layers are shown from left to right. An SDF layer only contains the signed distance function of a subset of closed features without any color information. We show the color of foreground pixels within each SDF layer in (c) only for better visualization. (d) Synthesis result without SDF layers. Leaves from different layers incorrectly merge together, as marked in blue circles. (e) Our multilayer synthesis result ($\alpha = 0.3$).

To accommodate multiple SDFs in our optimization framework, we allocate a distinct channel for each SDF. To construct multiple SDF channels for an input exemplar, we interactively separate curvilinear features into multiple layers (see Fig. 7b), and compute a signed distance function for the features on each layer. If part of some visually important feature from the bottom layer is occluded by a top layer, we can manually recover the occluded feature boundaries to obtain more reasonable SDF distribution on that bottom layer. Each SDF channel has an associated foreground mask, which encloses all curvilinear features on that layer. Pixels inside the mask are called foreground pixels of the corresponding SDF channel. To make the SDF channels behave like layers in a stack of images, they are associated with a predefined order, which will be useful in various scenarios. For example, to correctly handle occlusions among image elements during image synthesis, an occluding layer should precede an occluded layer in this predefined order. Note that in our multilayer representation, only the SDF channel is separated into multiple layers, while the three color channels remain intact. The color of a foreground pixel in a SDF layer is still defined by the three color channels at the same pixel location in the original input exemplar.

Suppose there are L SDF channels. Let ϕ_m^i and ϕ_m^o denote the mth SDF channel associated with the input exemplar and synthesized output, respectively. We further use $\mathbf{D}_{\mathbf{x}}^{\mathbf{m}}$ (or $\mathbf{D}_{\mathbf{z}}^{\mathbf{m}}$) to denote the vector of concatenated mth channel SDF values within a neighborhood centered at pixel $\mathbf{x}$ (or $\mathbf{z}$). The energy function in (3) is revised as follows:

$$\alpha \sum_{\mathbf{x}\in S_X} \|\mathbf{A}_{\mathbf{x}} - \mathbf{A}_{\mathbf{z}_{\mathbf{x}}}\|^2 + (1-\alpha)\sum_{m=1}^{L}\sum_{\mathbf{x}\in S_X} \left\|\mathbf{D}_{\mathbf{x}}^{m} - \mathbf{D}_{\mathbf{z}_{\mathbf{x}}}^{m}\right\|^2. \quad (8)$$

Note that all SDF layers are optimized at the same time in every iteration. To accommodate multiple SDF layers, our EM-based optimization needs to be revised as follows. During the M-Step, when we search for the best matching neighborhoods in the input exemplar, the SSD-based similarity metric needs to consider all SDF channels as well as all color channels altogether. In this way, we can not only synthesize image elements associated with individual SDF channels, but also effectively synthesize novel instances of interactions among different SDF channels as well as correlations between curvilinear features and color/appearance values.

During the E-step, when we optimize the SDF channels of an output image, we apply the level set method to every SDF channel independently and compute a distinct velocity field prescribed by (6) for every SDF channel during each time step according to the corresponding SDF channel of the matching neighborhoods from the input exemplar. Every SDF channel generates its own warped version of the color map from the previous time step. At the end of the current time step, we "collapse" all warped versions into a single image by observing the predefined order among the SDF channels. That means, if the m_ith SDF channel precedes the m_jth channel, foreground pixel colors from the m_ith warped image should overwrite colors at the same pixels from the m_jth warped image. At the end of the E-step, once level set-based optimization concludes, we further update pixelwise colors in the "collapsed" image.

The pseudocode in Algorithm 1 shows the outline of our multiresolution framework. Fig. 7e demonstrates that the synthesis result can be much improved once layers have been explicitly considered using our multilayer synthesis.

Algorithm 1. Multiresolution Multilayer Synthesis.
INPUT $\leftarrow$ input exemplar $\mathbf{Z}$, multilayer SDFs;
INPUT $\leftarrow$ weight α, target image resolution R_T;
$R_0 \leftarrow R_T/16$;
for (resolution $R_c = R_0$ **to** R_T, $R_c \leftarrow 2R_c$) **do**
  if ($R_c = R_0$) **then**
    Initialize $\mathbf{X}^0$ using image quilting enhanced with SDFs;
  else
    Upsample the resolution of $\mathbf{X}^0$ to R_c;
  end if
  for (neighborhood size $S_c = S_{UB}$ **downto** S_{LB}, $S_c \leftarrow S_c/2$) **do**
    $\lambda \leftarrow 0.2, \mu \leftarrow 0.8$;
    for (level set iteration $n = 1$ **to** N) **do**
      $\mathbf{z}_{\mathbf{x}} \leftarrow$ the neighborhood in $\mathbf{Z}$ most similar to $\mathbf{x}$, $\forall \mathbf{x} \in \mathbf{X}^{n-1}$;
      $\mathbf{X}^n \leftarrow \arg\min_{\mathbf{X}} E_t(\mathbf{X}; \{\mathbf{z}_{\mathbf{x}}\}_{\mathbf{x}\in\mathbf{X}^{n-1}})$, where E_t is the energy defined in Eq. (8);

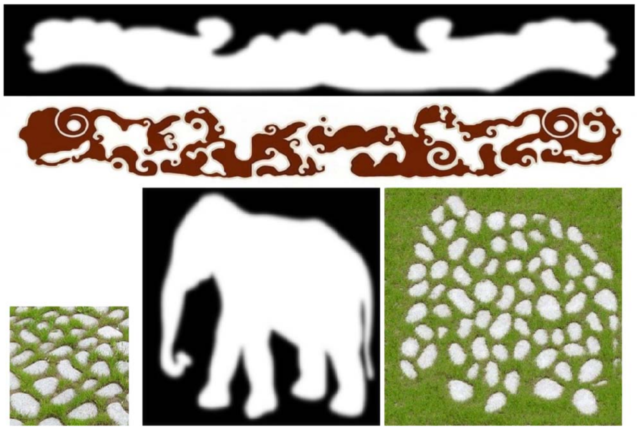


Fig. 8. Controllable synthesis results. Top: The input exemplar is shown in Fig. 2. The control mask is shown in black and white, where the white region stands for the foreground. ($\alpha = 0$) Bottom: Left is the input exemplar. Middle is the control mask. Right is the result ($\alpha = 0.4$).

      $\lambda \leftarrow \lambda + 0.75/N, \mu \leftarrow \mu - 0.75/N$;
    end for
    $\mathbf{X}^0 \leftarrow \mathbf{X}^N$;
  end for
end for
OUTPUT $\leftarrow$ synthesized texture $\mathbf{X}$

5 Controllable Synthesis

Our method can be easily modified to support controllable synthesis. Given an input exemplar and a control mask for the output (as in Fig. 8), we would like to perform exemplar-based synthesis and generate an image where the shape and boundary of the mask are well delineated by curvilinear features in the synthesized image. Here, we take advantage of our multilayer scheme and add an extra SDF layer. For the synthesized image, this extra layer contains an SDF of the mask boundary. For the input exemplar, this extra layer contains an SDF of all features. The SDF on this new layer will affect neighborhood matching during the M-step and level set-based optimization during the E-step. During neighborhood matching, if a neighborhood overlaps with the mask boundary or falls completely outside the mask, the weight of the extra SDF channel is much increased (typically four times larger) to locate a neighborhood (in the input exemplar) whose extra SDF channel matches the SDF of the mask boundary closely. When computing the velocity field during level set-based optimization, (6) is revised as follows:

$$\mathbf{V}_s' = \sum_{j\in s_{\mathbf{x}}} w_j\left(\Phi^o(\mathbf{x}) - \Phi_j^i(\mathbf{x})\right)\frac{\nabla\Phi^o(\mathbf{x})}{\|\Phi^o(\mathbf{x})\|}, \quad (9)$$

where w_j is a neighborhood-specific weight. Weights for those neighborhoods (from the input exemplar), which match neighborhoods (in the synthesized image) overlapping with the mask boundary, are also four times larger than other neighborhoods to ensure that the "forces" from such neighborhoods are sufficiently strong so that they can push the zero level set toward the mask boundary. Results from controllable synthesis can be found in Fig. 8.

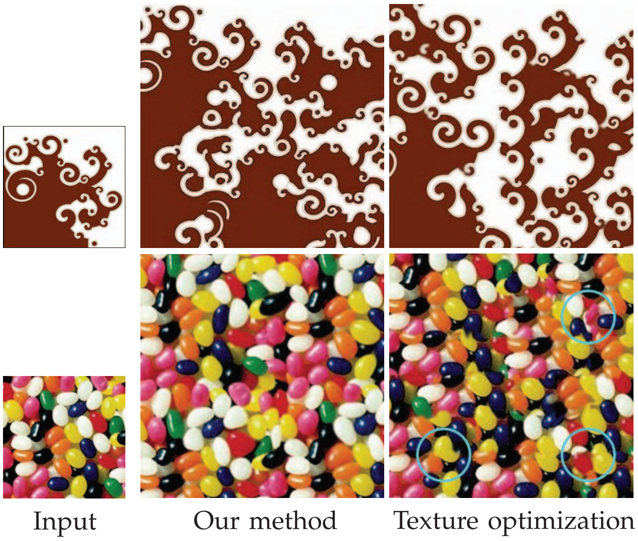


Fig. 9. Comparison between our method and the texture optimization algorithm in [26]. (Top: $\alpha = 0$. Bottom: $\alpha = 0.5$). The third column shows results from an improved version of texture optimization, which incorporates a signed distance function of features. Results in the first row were synthesized using the signed distance function only. Artifacts are marked with blue circles.

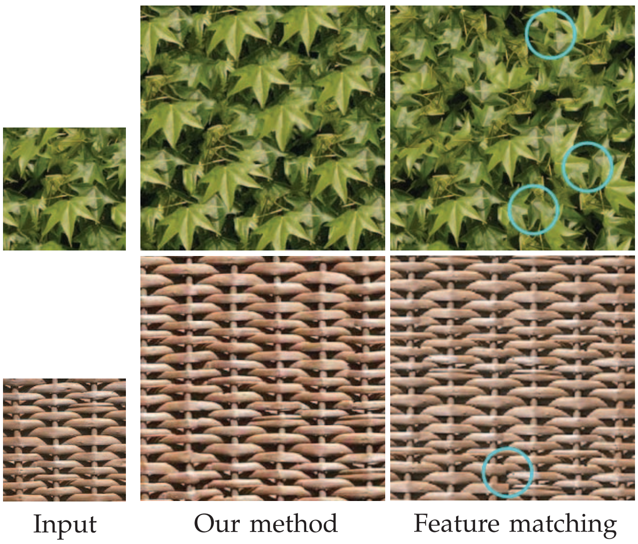


Fig. 10. Comparison between our method and the feature matching technique in [25]. Artifacts are marked with blue circles. Our method generates better results using multilayer synthesis (Top: $\alpha = 0.3$. Bottom: $\alpha = 0.3$).

6 Results and Comparisons

We have fully implemented our algorithms and successfully tested them on a variety of art patterns and layered textures. The running time for generating a 256×256 (512×512) synthesized image from a 128×128 input exemplar is around 3 (11.5) minutes on an Intel 2.00-GHz Core i7-2630M processor. In our multiresolution scheme, the output resolution typically increases from 32×32 to 256×256. At each resolution, typically 10 EM iterations are performed. The size of image neighborhoods is decreased from $1/4$ current image resolution down to 8×8 from iteration to iteration. During each E-step, five time steps of the level set method are performed. λ and μ in (7) are automatically adjusted according to the discussion in Section 3.1.

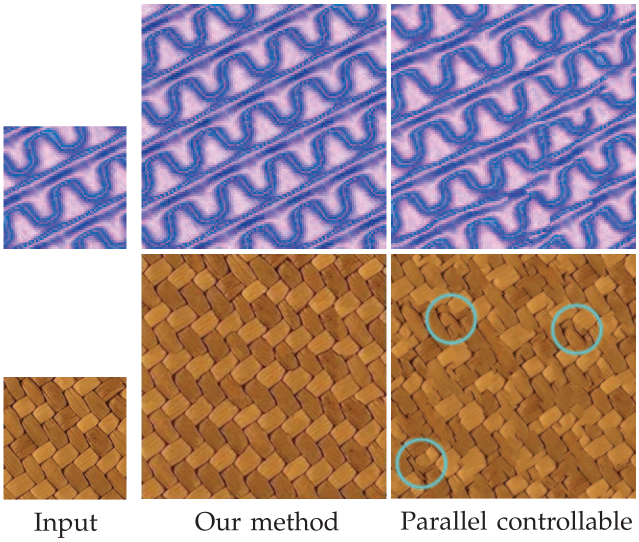


Fig. 11. Comparison between our method and the parallel controllable synthesis algorithm in [14]. (Top: $\alpha = 0.1$. Bottom: $\alpha = 0.8$). Artifacts are marked with blue circles.

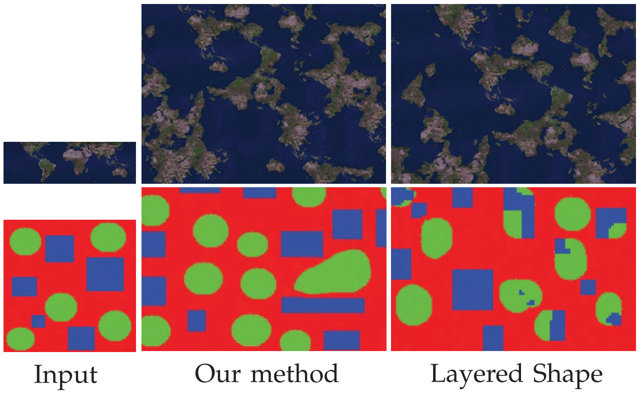


Fig. 12. Comparison between our method and the layered shape synthesis algorithm in [29] (Top: $\alpha = 0.1$. Bottom: $\alpha = 0.5$).

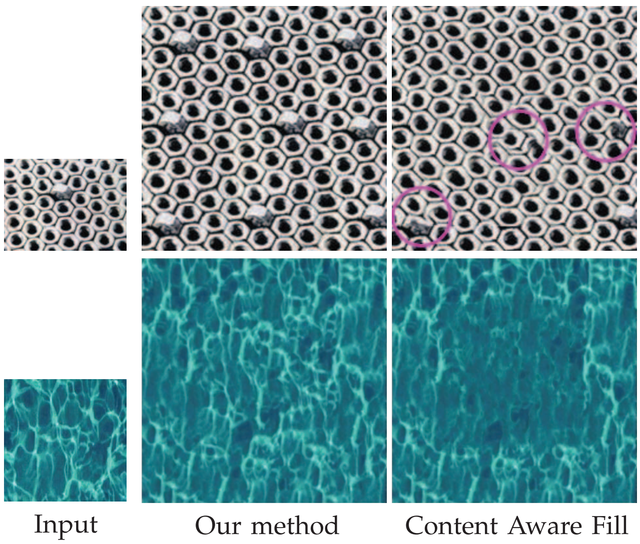


Fig. 13. Comparison between our method and content-aware fill in Adobe Photoshop CS5. First column: input exemplars. Second column: our results (Top: $\alpha = 0.1$. Bottom: $\alpha = 0.8$). Third column: results from Adobe Photoshop. Artifacts are marked with purple circles.

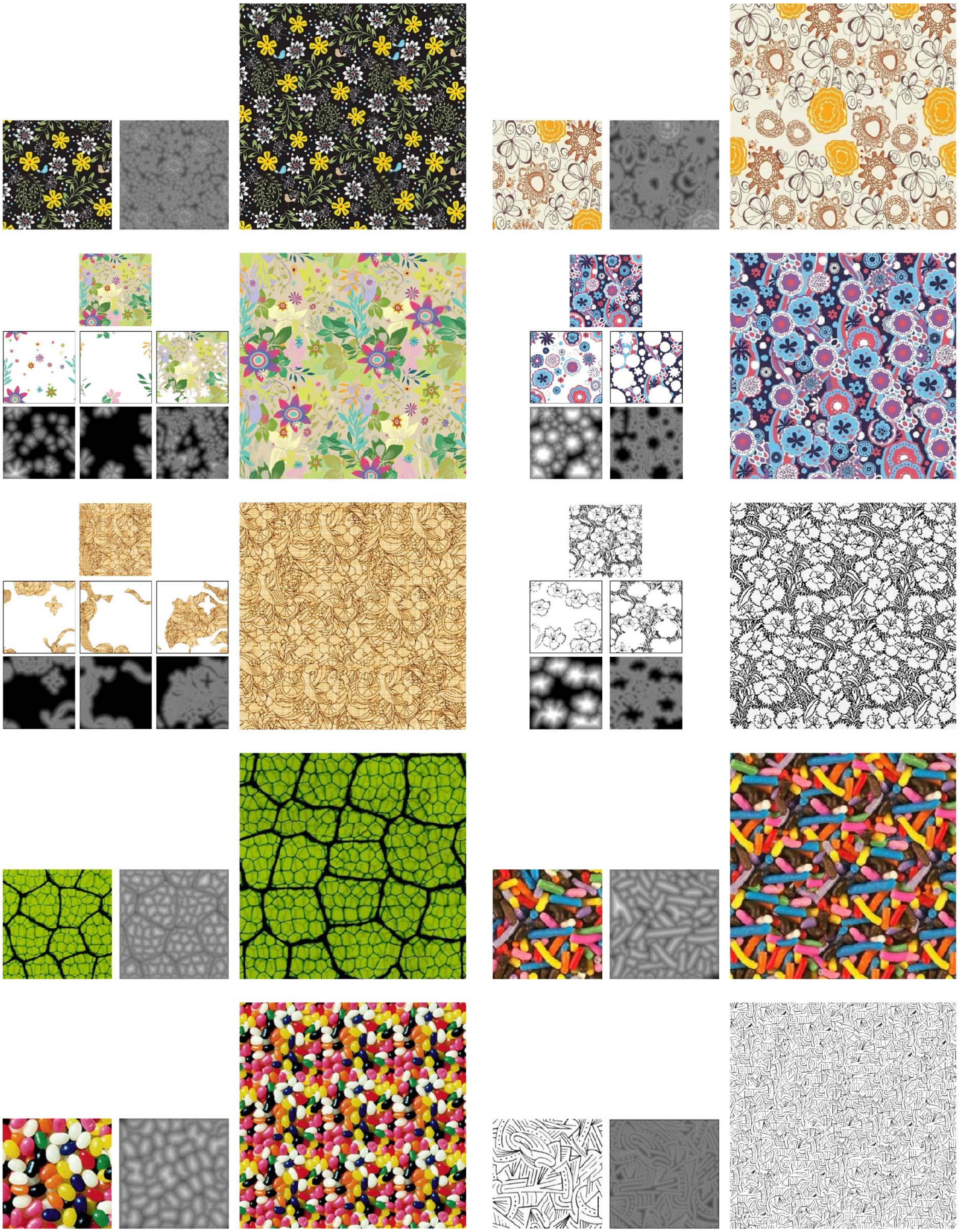


Fig. 14. More results from our method. The first row shows synthesis results for single-layer art patterns. The second and third rows show synthesis results for multilayer art patterns. High-resolution results (4×4) are shown in the last row. (From top to bottom: $\alpha = 0.3, 0.6; 0.3, 0.5; 0.2, 0; 0.3, 0.5; 0.4, 0.$)

We have also compared our method against the state-of-the-art synthesis algorithms [14], [25], [26], [29] and the content-aware fill technique in Adobe Photoshop CS5 on a set of difficult art patterns and textures. Representative comparison results are shown in Figs. 9, 10, 11, 12, and 13. Note that for fair comparison, the results obtained using texture optimization in Fig. 9 were actually generated from our improved implementation that incorporates a signed distance function for features. Thanks to the level set-based curve optimization, our method can smoothly connect curve segments from local image patches to form globally coherent patterns and textures. In comparison to results from existing synthesis techniques, our results have the least broken or misaligned curvilinear features and the least amount of visual repetition. Our method also has a strong generalization capability because it successfully generates novel local patterns that are visually similar to local patches in the original exemplars but not exactly the same. Texture optimization produces blurry regions because of its averaging operation. It does not inherently maintain continuous and smooth feature boundaries; therefore, misaligned edges often exist in their synthesized line drawings (see Fig. 9). In comparison with the feature matching technique [25], our method is clearly superior when dealing with textures and patterns that consist of multiple layers because layering is not explicitly considered in their technique. In addition, their technique is also more likely to generate misaligned features due to the lack of global optimization. The parallel controllable synthesis algorithm [14] is fast but cannot guarantee boundary continuity especially when the input exemplar is nonperiodic, as shown in Fig. 11. Compared with the layered shape synthesis method [29], our level set method can treat color and shape in a more balanced manner, as shown in Fig. 12. It can be observed that our method can not only generate competing results on terrain exemplars (top row), but also can successfully deal with the failure cases of the layered shape synthesis method (bottom row). The content-aware fill technique of Adobe Photoshop takes advantage of textured background patches when filling a user-selected region and by nature is a quilting method. It sometimes gives rise to obvious feature discontinuities along region and patch boundaries.

More results of our method can be found in Fig. 14.

7 Conclusions and Future Work

We have presented an exemplar-based synthesis framework for art patterns and layered textures. It takes a global optimization approach. Our energy function measures both the appearance similarity of color patterns and shape similarity of curvilinear features between an input exemplar and a synthesized image. We have developed an overall EM-style technique for minimizing this energy function. The shape similarity part of the energy is minimized through an innovative application of the popular level set method. We further generalize our energy function and optimization technique to multilayer pattern and texture synthesis. Our generalized optimization can effectively handle multiple layers and synthesize valid instances of interactions.

Limitations. There exist a few aspects of our algorithm and implementation that deserve further investigation. First, albeit fine-scale details in our synthesis results are nonrepetitive, sometimes a certain level of regularity is present at larger scales to maintain feature continuity due to our quilting-based initialization, as shown in Fig. 14 (bottom left), especially when the input exemplar exhibits regularity. Second, there exists manual involvement in two steps: edge connection and layer extraction. We need the signed distance function of a set of closed curves to initialize the level set function. Since edge detection does not guarantee closed results, we need to interactively connect a few open edges. An automatic scheme for closed curve construction can be developed by integrating region-based image segmentation, such as the graphcut algorithm [40], with edge detection as region boundaries are always closed. Prior to multilayer synthesis, we also need to interactively construct multiple SDF layers for the input exemplar. Automatic layer extraction needs to automatically recognize partially occluded image regions. It is, therefore, a challenging problem worth further exploration. Furthermore, parallel computing techniques could be adopted to speed up the optimization stage of our method.

Acknowledgments

This work was partially supported by Hong Kong Research Grants Council under General Research Funds (HKU718712, HKU719313).

References

[1] S.J. Osher and J. Sethian, "Fronts Propagating with Curvature Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations," *Computational Physics,* vol. 79, pp. 12-49, 1988.

[2] J. Sethian, *Level Set Methods and Fast Marching Methods.* Cambridge Univ. Press, 1999.

[3] S.J. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces.* Springer, 2003.

[4] Wikipedia, "Level Set Method—Wikipedia, the Free Encyclopedia," http://en.wikipedia.org/wiki/Level_set_method, 2012.

[5] M. Wong, D. Zongker, and D. Salesin, "Computer-Generated Floral Ornament," *Proc. SIGGRAPH,* pp. 423-434, 1998.

[6] M. Kaplan and E. Cohen, "Computer Generated Celtic Design," *Proc. 14th Eurographics Workshop Rendering,* pp. 9-19, 2003.

[7] V. Kraevoy and A. Sheffer, "Cross-Parameterization and Compatible Remeshing of 3D Models," *ACM Trans. Graphics,* vol. 23, no. 3, pp. 861-869, 2004.

[8] D. Garber, "Computational Models for Texture Analysis and Texture Synthesis," PhD dissertation, Univ. of Southern California, 1981.

[9] A.A. Efros and T. Leung, "Texture Synthesis by Non-Parametric Sampling," *Proc. IEEE Seventh Int'l Conf. Computer Vision,* pp. 1033-1038, 1999.

[10] K. Popat and R. Picard, "Novel Cluster-Based Probability Model for Texture Synthesis, Classification, and Compression," *Proc. SPIE,* vol. 2094, pp. 756-768, 1993.

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

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

[13] M. Ashikhmin, "Synthesizing Natural Textures," *Proc. Symp. Interactive 3D Graphics,* pp. 217-226, 2001.

[14] S. Lefebvre and H. Hoppe, "Parallel Controllable Texture Synthesis," *ACM Trans. Graphics,* vol. 24, no. 3, pp. 777-786, 2005.

[15] C. Han, E. Risser, R. Ramamoorthi, and E. Grinspun, "Multiscale Texture Synthesis," *ACM Trans. Graphics,* vol. 27, no. 3, article 51, 2008.

[16] E. Risser, C. Han, R. Dahyot, and E. Grinspun, "Synthesizing Structured Image Hybrids," *ACM Trans. Graphics,* vol. 29, no. 4, article 85, 2010.

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

[18] J. Zhang, K. Zhou, L. Velho, B. Guo, and H.-Y. Shum, "Synthesis of Progressively-Variant Textures on Arbitrary Surfaces," *ACM Trans. Graphics,* vol. 22, no. 3, pp. 295-302, 2003.

[19] S. Lefebvre and H. Hoppe, "Appearance-Space Texture Synthesis," *ACM Trans. Graphics,* vol. 25, no. 3, pp. 541-548, 2006.

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

[21] 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.

[22] A.A. Efros and W.T. Freeman, "Image Quilting for Texture Synthesis and Transfer," *Proc. SIGGRAPH,* pp. 341-346, 2001.
[23] M.F. Cohen, J. Shade, S. Hiller, and O. Deussen, "Wang Tiles for Image and Texture Generation," *ACM Trans. Graphics,* vol. 22, no. 3, pp. 287-294, 2003.
[24] V. Kwatra, A. Schdl, I. Essa, G. Turk, and A. Bobick, "Graphcut Textures: Image and Video Synthesis Using Graph Cuts," *ACM Trans. Graphics,* vol. 22, no. 3, pp. 277-286, 2003.
[25] Q. Wu and Y. Yu, "Feature Matching and Deformation for Texture Synthesis," *ACM Trans. Graphics,* vol. 23, no. 3, pp. 364-367, 2004.
[26] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra, "Texture Optimization for Example-Based Synthesis," *ACM Trans. Graphics,* vol. 24, no. 3, pp. 795-802, 2005.
[27] J. Kopf, C.-W. Fu, D. Cohen-or, O. Deussen, D. Lischinski, and T.-T. Wong, "Solid Texture Synthesis from 2D Exemplars," *ACM Trans. Graphics,* vol. 26, no. 3, pp. 2:1-2:9, 2007.
[28] L.-Y. Wei, J. Han, K. Zhou, H. Bao, B. Guo, and H.-Y. Shum, "Inverse Texture Synthesis," *ACM Trans. Graphics,* vol. 27, no. 3, article 52, 2008.
[29] A. Rosenberger, D. Cohen-Or, and D. Lischinski, "Layered Shape Synthesis: Automatic Generation of Control Maps for Non-Stationary Textures," *ACM Trans. Graphics,* vol. 28, no. 5, article 107, 2009.
[30] C. Ma, L. Wei, and X. Tong, "Discrete Element Textures," *ACM Trans. Graphics,* vol. 30, no. 4, article 62, 2011.
[31] H. Barrow, J. Tenenbaum, R. Bolles, and H. Wolf, "Parametric Correspondence and Chamfer Matching: Two New Techniques for Image Matching," *Proc. Fifth Int'l Joint Conf. Artificial Intelligence,* pp. 659-663, 1977.
[32] J. Canny, "A Computational Approach to Edge Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. PAMI-8, no. 6, pp. 679-698, Nov. 1986.
[33] G. Mclachlan and T. Krishnan, *The EM Algorithm and Extensions.* John Wiley & Sons, 1997.
[34] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, and A. Wu, "An Optimal Algorithm for Approximate Nearest Neighbor Searching Fixed Dimensions," *J. ACM,* vol. 45, no. 6, pp. 891-923, 1998.
[35] C. Barnes, E. Shechtman, A. Finkelstein, and D.B. Goldman, "PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing," *ACM Trans. Graphics,* vol. 28, no. 3, article 24, Aug. 2009.
[36] R. Courant, E. Isaacson, and M. Rees, "On the Solution of Nonlinear Hyperbolic Differential Equations by Finite Differences," *Comm. Pure and Applied Math.,* vol. 5, pp. 243-255, 1952.
[37] D. Adalsteinsson and J.A. Sethian, "A Fast Level Set Method for Propagating Interfaces," *Computational Physics,* vol. 118, no. 2, pp. 269-277, 1995.
[38] S.-Y. Lee, K.-Y. Chwa, and S.Y. Shin, "Image Metamorphosis Using Snakes and Free-Form Deformations," *Proc. SIGGRAPH,* pp. 439-448, 1995.
[39] L.-Y. Wei and M. Levoy, "Order-Independent Texture Synthesis," Technical Report TR 2002, Stanford Univ., 2002.
[40] Y. Boykov and M. Jolly, "Interactive Graph Cuts for Optimal Boundary and Region Segmentation of Objects in N-D Images," *Proc. IEEE Eighth Int'l Conf. Computer Vision,* vol. 1, pp. 105-112, 2001.

Ruobing Wu received the BE degree from Tsinghua University in 2010. He is currently working toward the PhD degree in the Department of Computer Science, The University of Hong Kong. His research interest includes computer graphics, computer vision, texture synthesis, image processing, and machine learning. His current research topic is object categorization through deep learning. He is currently a Hong Kong postgraduate fellowship holder.

Wenping Wang is currently a professor in the Department of Computer Science, The University of Hong Kong. His research interests include computer graphics, visualization, and geometric computing. He has recently focused on mesh generation and polyhedral surface modeling for architectural design. He is a journal associate editor of *Computer Aided Geometric Design*, *Computers & Graphics*, *Computer Graphics Forum*, and *IEEE Transactions on Visualization and Computer Graphics* (2008-2012). He is the program chair of Pacific Graphics 2003, ACM Symposium on Physical and Solid Modeling 2006, and International Conference on Shape Modeling 2009, and conference chair of Pacific Graphics 2012, ACM Symposium on Physical and Solid Modeling 2013, and SIGGRAPH Asia 2013.

Yizhou Yu received the PhD degree from the University of California at Berkeley in 2000. He is currently a full professor in the Department of Computer Science, The University of Hong Kong, and an adjunct professor at the University of Illinois, Urbana-Champaign. He received the 2002 National Science Foundation CAREER Award and the Best Paper Award at 2005 and 2011 ACM SIGGRAPH/EG Symposium on Computer Animation. He is on the editorial board of *Computer Graphics Forum* and *International Journal of Software and Informatics*. He is the program chair of Pacific Graphics 2009, Computer Animation and Social Agents 2012, and has served on the program committee of many leading international conferences, including SIGGRAPH, SIGGRAPH Asia, and International Conference on Computer Vision. His current research interests include data-driven methods for computer graphics and vision, digital geometry processing, video analytics, and biomedical data analysis.