CGI2021 manuscript No. 32
(will be inserted by the editor)

Automatic embroidery texture synthesis for garment design and online display

Xinyang Guan · Likang Luo · Honglin Li · He Wang · Chen Liu · Su Wang · Xiaogang Jin ✉

Abstract We introduce an automatic texture synthesis based framework to convert an arbitrary input image into embroidery style art for garment design and online display. Given an input image and some reference textures, we first extract key embroidery regions from the input image using image segmentation. Each segmented region is single-colored and labeled with a stitch style automatically. We then fill these regions with embroidery reference textures via a stitch-style-based texture synthesis method. For each region, our approach maintains color similarity before and after synthesis, along with stitch style consistency. Compared to existing approaches, our method is able to generate digital embroidery patterns with faithful details automatically. Moreover, it can accept diverse input images effectively, enabling a fast preview of the embroidery patterns synthesized on digital garments interactively, and therefore accelerating the workflow from design to production. We validate our method through extensive experimentation and comparison.

Keywords Embroidery · Non-photorealistic rendering · Image-based artistic rendering

Xiaogang Jin, jin@cad.zju.edu.cn

Xinyang Guan · Xiaogang Jin
State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310058, China

Likang Luo · Chen Liu
Zhejiang Linctex Digital Technology Co. Ltd., China

Honglin Li
Quanzhou Medical College, China

He Wang
School of Computing, University of Leeds, UK

Su Wang
School of Software and Microelectronics, Peking University, China

1 Introduction

Embroidery can be defined as the craft of decorating fabric with sophisticated design patterns woven over its surface, providing an aesthetic function for design which is commonly observed on clothing. Owing to its labor-intensive manual work and training, it is prohibitively slow to visualize an embroidery design without completing at least some of the actual work physically. Existing procedures deal with this problem via deep neural networks [23], patch-based methods [22], or image-based interactive methods [7,9], which greatly rely on reference images or require complex user interaction. We argue that this problem can be solved by automated style-transfer from design images to their embroidery-style correspondences using texture synthesis, and thus design an automated tool for embroidery design visualization.

Texture synthesis is ubiquitous in digital content generation. It can create an image of a similar style given a reference image [31]. Some approaches focus on developing a unified framework for image style transfer [20,22] in order to generate stylized results with faithful effects. However, they may fail to capture the regularity of the embroidery stitch styles, and cause undesirable ambiguities or distorted stitches in the results (see Fig. 1(c) and (d)). Some methods [7,9] rely heavily on user interaction to perform the segmentation and stitch style selection, which is very time-consuming when the content of the image is complex; while others impose restrictions on the input images, making them impractical when dealing with various types of user input. Specifically, existing methods [21,24,23] are simply designed for a particular embroidery stitch style, and cannot be easily generalized for other widely-used stitches such as the satin stitch used in folk handicraft and the cos-

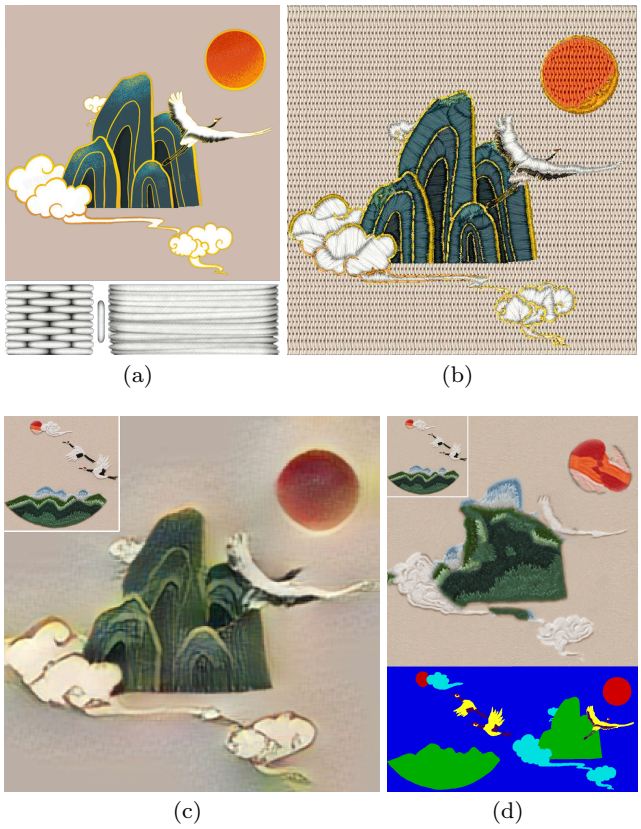


Fig. 1 Given an input image and some reference textures, our method is able to convert the input into an embroidery artwork automatically. (a) The original image along with three reference textures (bottom, from left to right: reference texture for the long-short stitch, the edge stitch, and the satin stitch, respectively) used by our system, (b) our result, (c) result of neural style transfer [20] with the reference image in the top-left corner, and (d) result of patch-based synthesis [22], bottom images are its masks, and the reference image is in the top-left corner.

tume industry. Neural-based methods [20,23] need reference images to capture semantic information in the network, which is often inconsistent with their input images. Patch-based methods [28,22] require users to specify guiding information as input, and the reference images should be sufficiently similar which not only leaves the embroidery segmentation workload to users, but can also lead to failures when the shapes of the reference and target are distinctive (see Fig. 1(d)). In general, the previous methods have limitations such as unstable results, time-consuming interactions, and insufficient generality for various types of input, which limit their applications in garment design and online display. Answering the question of how to synthesize faithful embroidery designs with fast response time remains challenging.

To address these challenges, we present a novel method to transfer arbitrary input images into embroidery artworks automatically with the assistance of auxiliary reference textures. Our framework first employs the features of three different common stitch styles (the long-short stitch, the satin stitch, and the edge stitch, which dominate the embroidery garment stitch styles) to perform the embroidery region segmentation, as well as the stitch style labeling procedures automatically without any user interaction. Second, the subsequent step of texture synthesis is developed to preserve the visual effects of each stitch style faithfully. Owing to the convenience of replacing and modifying reference textures, we can synthesize diverse results with flexibility. In this way, for the same input image, we can easily synthesize different results with different colors or stitch styles. We can also generate the corresponding normal of the resulting embroidery by taking the normal maps of reference textures as input.

In summary, our main contributions are listed as follows:

- We present an embroidery transfer framework integrated with a novel image segmentation method. This can produce virtual embroidery work with arbitrary input images automatically, and our method makes it possible to preview the embroidery effect of a casual image on digital garments interactively.
- We design a set of stitch-style-based texture synthesis methods. With the assistance of reference textures, this can flexibly adapt to the three main types of embroidery.

2 Related Work

Texture synthesis and transfer of embroidery style, usually regarded as a branch of image stylization, can be divided into parametric and non-parametric methods.

2.1 Parametric Texture Synthesis and Transfer

Deep neural networks contribute significantly to the field of image stylization. They perform stylization by statistical analysis, which finds that style and content can be separated from a given image by neural networks [13,14]. With the development of network architecture, one single trainable model can transfer arbitrary artistic styles [6], which can then be applied to the embroidery problem. This avoids the need for training on a variety of stitch styles separately. Recently, a CNN-based method [23] was designed for embroidery style transfer, which can produce results with irregularly placed stitches. However, neural networks work well only on images which contain more natural contents with clear

high-level semantic information [18]. When dealing with abstract images (e.g., sketch or cartoon images), they will fail to extract sufficient information, and hence produce poor results in embroidery works. Users usually take their own abstract design knowledge as input, which is unfriendly to the neural network.

Another alternative for neural-based style transfer is the use of recently-proposed Generative Adversarial Networks (GANs) [15]. Results are improved by the minimax game between generator and discriminator networks. Based on this framework, the concept of image translation [17] is developing rapidly to solve a variety of stylization problems. The occurrence of networks like CycleGAN [33] can even train the network with unpaired image samples, making it easier to build an embroidery image training database. However, for embroidery works, building high-quality embroidery databases is just the first step. The images should be organized by their stitch styles, and the network should be designed accordingly, rather than being applied to the embroidery problem directly.

2.2 Non-Parametric Texture Synthesis and Transfer

Non-parametric methods not only release the burden of data preparation, but also produce more predictable and controllable results. Following the concept of image-based artistic rendering [19], example-based methods belong to one category pioneered by methods like image analogies [16] used in situations such as texture-by-numbers. In example-based synthesis, patch-based methods [3] rearrange patches in the reference image in order to generate the target one, as shown in the image quilting method [11]. The ability of preserving details and significant results inspired us in using patches rather than pixels as synthesis units. PatchMatch [1] and its following methods [2,4] are proposed to accelerate the procedure of searching proper patches, and currently, the patch-based method can demonstrate competitive visual quality results at an acceptable speed [22], compared with neural-based methods. However, time-consuming efforts must be taken in optimizing search results to achieve a better mapping. As the distribution of embroidery stitches shows more regularity compared to other texture generation cases (e.g., inhomogeneous textures generation [32], text effects generation [28] or poster headline generation [29]), its matching method must be specially designed to streamline the procedure.

Stroke-based synthesis belongs to another category of methods which can be applied to the element distribution problem. Each algorithm should be specially designed according to its corresponding style. We note that embroidery with different stitch styles may require different distribution methods. Aiming at Chinese irregular needling embroidery, a series of algorithms [27, 26,21] have been developed to produce results directly from input images. However, these methods cannot be generalized for other commonly used stitch styles, such as the satin stitch, etc. Similar limitations exist in methods [25,24], as their artistic rendering methods cannot easily be applied to the usual regular stitches directly. For common stitches, the existing embroidery generation methods focus more on modeling different stitch styles and stitch rendering. Chen et al. presented a line-drawing-based method [7] which uses clean embroidery patterns as input, and relies on user interaction to decide the stitch style. Cui et al. [9] also left the segmentation of complicated reference images and the selection of stitch style as the future work. The level of human intervention in these aforementioned methods restricts their applications which require fast response. Different from these methods, our presented work can transfer input images into their embroidery styles automatically.

3 Overview

Our method aims to generate the embroidery image from an arbitrary input image without user interaction. As shown in Fig. 2, our pipeline consists of two main phases: embroidery region segmentation and stitch-style-based texture synthesis, respectively.

In the embroidery region segmentation, we first extract the main colors from the input image in order to remove unwanted noises and details, and abstract the input image as a cartoon image. Subsequently, we extract each single-colored slice from the image, and separate each slice into one or more unconnected components. For each component, our method uses the stroke width transform method to analyze its shape. With the shape information, we can divide the connected component into one or more regions via a graph-cut algorithm [10] which is designed with a set of stitch-styled-based formulations, enabling stitch style to be labeled for each region with the segmentation step simultaneously.

In the stitch-style-based texture synthesis, we fill each region with an embroidery pattern independently with three types of stitches (the long-short stitch, the satin stitch and the edge stitch) similar to Cui et al. [9]. Our system stores three types of preset reference textures, which will be recolored for each different region before synthesis. By taking both the shape and stitch style of each region into consideration, we perform the texture synthesis step by a texture mapping process based on the calculated UV value of each pixel in these regions. After all the regions are synthesized, we obtain the output embroidery image.

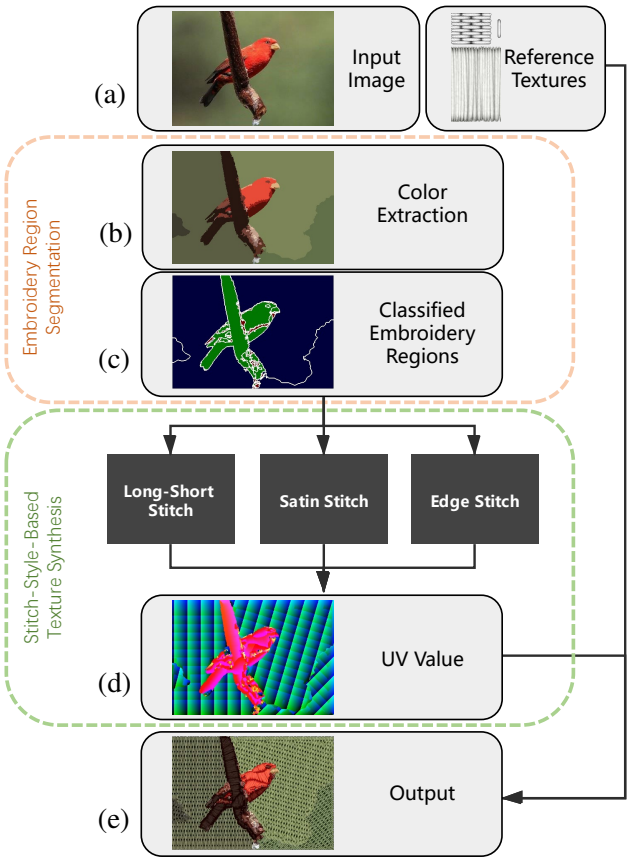


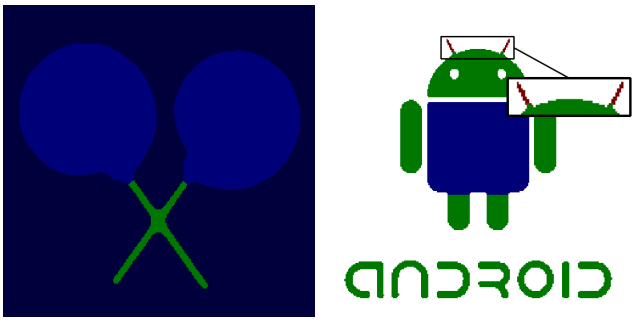
Fig. 2 Overview of our embroidery transfer framework. Given an input image (a), our framework first performs color extraction (b) and stitch style classification (c) steps to achieve the embroidery region segmentation procedure. Then, based on three different stitch styles, our method calculates the UV value map (d), and performs the texture synthesis procedure using the recolored preset reference textures, synthesizing the stylized embroidery image (e).

We will articulate the aforementioned phases in Section 4 and Section 5 in detail, respectively.

4 Embroidery Region Segmentation

Segmenting images into embroidery regions is necessary for dealing with arbitrary input images, such as sketches, photos, etc. This processing procedure is composed of two main steps, which are color extraction and stitch-style-based segmentation.

The color extraction procedure can be performed by common clustering algorithms, such as mean-shift segmentation [8], or a hierarchical clustering tree. We use median filters to preprocess the image and choose the CIELab color space to calculate the color differences. We obtain 4-connected components after analyzing each clustered single-colored slice. Each component



(a) (b)

Fig. 3 Two embroidery region segmentation results via our graph-cut algorithm. (a) Lollipop and (b) robot. The blue and dark blue areas represent the long-short stitch regions. The green areas represent the satin stitch regions. The red areas represent the edge stitch regions. The white areas are transparent in the original image.

may have irregular shapes and should be further segmented into embroidery regions.

Observations show that widths of the pattern shapes affect the stitch style selection significantly. In practice, extremely narrow shapes (see the red antennas of the robot in Fig. 3(b)) tend to use the edge stitch. Wide shapes (see the dark background and the blue candy part of the lollipops in Fig. 3(a) and the blue body in Fig. 3(b)) tend to use the long-short stitch. For shapes with moderate widths (see the green lollipop sticks in Fig. 3(a), the limbs and the text in Fig. 3(b)), the satin stitch can be chosen. The stroke width transform (SWT) method [12] is employed to measure the width by pairing two contour pixels across the component shape and getting their distance. Note that the pairing test fails in the following two cases. If one of the contour pixels is located on the boundary of the image, it will lack the gradient value required for the test; if the shape is irregular and the gradient directions of two contour pixels are too different, the test may also fail. To solve the aforementioned two special cases, the connected component should be directly assigned with the long-short stitch style, since the long-short stitch can adapt to any complex shapes more flexibly than the other two stitch styles. In our experiments, we empirically set a threshold of the pixels failing to obtain the width to $n^{0.8}$, where n represents the number of pixels of the entire component.

When the width of the pixels is fully provided in the component, we use the graph-cut algorithm [10] to perform the segmentation procedure by designing the data cost, the smooth cost, and the label cost. We formulate the data cost as follows:

$$D(n_p, LS) = \frac{k_1}{1 + e^{\frac{w - sW}{2}}}, p \in C, \quad (1)$$

$$D(n_p, ST) = \begin{cases} \frac{k_1}{1+e^{\frac{sW-w}{2}}}, & w \geq sW \\ \frac{k_2}{1+e^{2\times eW\times(w-eW)}}, & w < sW \end{cases}, p \in C, \quad (2)$$

$$D(n_p, ES) = \frac{k_2}{1+e^{2\times eW\times(eW-w)}}, p \in C, \quad (3)$$

$$D(n_p, LS) = D(n_p, ST) = D(n_p, ES) = 0, p \notin C, \quad (4)$$

where n_p is a pixel node in the graph, LS is the long-short stitch, ST is the satin stitch, ES is the edge stitch, eW is the desired width of the edge stitch region, sW is the desired width of the satin stitch region, k_1, k_2 are the weight parameters, and C represents a set of contour pixels with valid SWT values whose gradient direction difference is less than $\frac{\pi}{6}$ in the SWT contour pixel pairing test, because smaller difference indicates more credible width to classify the pixel. If a pixel cannot meet the condition of C, its data cost is set to zero regardless of the stitch style, and the pixel label is decided by the smooth cost and the label cost. Our observation shows that contour pixels are usually included in a reasonable region, while the classification of inner pixels may produce regions without contour pixels when there is noise in the SWT value. As a result, we only use the contour pixels in C.

The smooth cost formulation is defined as follows:

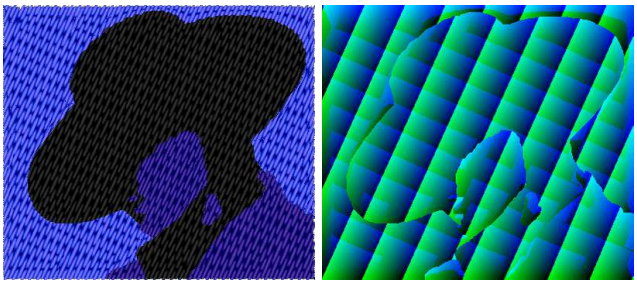
$$S(n_p, n_q) = \begin{cases} 0, & \text{if } s_p = s_q \\ k_3, & \text{if } s_p = ES \text{ or } s_q = ES \\ k_4, & \text{else} \end{cases}, \quad (5)$$

where n_p, n_q represent two neighbor pixel nodes in the graph, k_3, k_4 are the weight parameters, and s_p, s_q are the stitch styles of the pixel nodes, respectively. Note two types of node pairs can be defined as neighbors in the graph. Given the pixel **p**, the pixel **q** of its neighbor node could be its neighbor pixel, or its nearest contour pixel in the image. We experimentally set k_3 higher than k_4, which will prevent setting edge stitch pixels with different neighbors, and preserve the continuity of regions in detailed area.

The label cost formulation is defined as follows:

$$L(s_i) = c_{s_i}, \quad (6)$$

where s_i is a stitch style, and c_{s_i} is the corresponding cost. We empirically set c_{ST} the highest in the three stitch styles, to classify less regions into the satin stitch. This is because the data costs of the satin stitch and the long-short stitch are indistinguishable at a width of about sW. Using the label cost to choose the long-short stitch can adapt to more cases of irregular shapes. The edge stitch is less affected because it is mainly classified by the data cost with an extremely narrow width. We can also set the label cost to totally different values to adjust the proportion of three stitch styles. Using some extreme label costs can even generate embroidery with only certain stitch styles, such as embroidery with only the long-short stitch (see Fig. 4(a)), and embroidery without the satin stitch (see Fig. 16(e) and 16(f)), respectively.



(a) (b)

Fig. 4 The long-short stitch. (a) Embroidery work produced by our system, and (b) the UV value image as the intermediate result.

From this, each region has been segmented in a comparatively regular shape, and adapted to a specific stitch style.

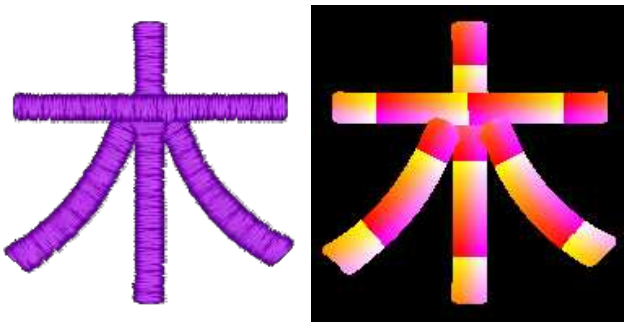
5 Stitch-Style-Based Texture Synthesis

In this texture synthesis step, we fill each region independently with the reference textures, according to the given stitch style. In the following five sub-sections, we illustrate the detailed texture synthesis procedures of the aforementioned three stitch styles, and introduce further improvements to produce colored and multi-layer embroidery, respectively.

5.1 Long-Short Stitch Synthesis

The long-short stitch (see Fig. 4(a)) is a traditional stitch, which consists of rows of stitches to show intensity graduation [7,9]. The number of repetition in each stitch row can vary with the width, making the long-short stitch flexible to fit any shapes, especially for the large areas or shapes with irregular contours. The texture synthesis step based on the long-short stitch paves the regions with rows of the stitch reference textures. During the process, the edge of each texture row should be aligned with the contours of the region. The detailed procedure is described as follows.

First, assuming that the stitch rows in one region are parallel with each other, we figure out the rotating direction of the stitches in the given region by searching along the region's external contour and averaging out the rotation of the neighboring regions. Although



(a) (b)

Fig. 5 The satin stitch. (a) Embroidery work produced by our system, and (b) the UV value image as the intermediate result.

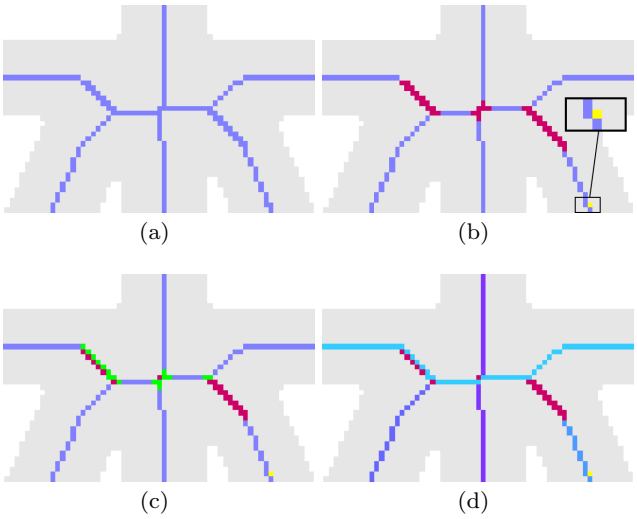
this method can maintain the consistency of direction, it may fail when neighbor rotations are absent, in which case we should calculate the bounding box for this region, and then take the main direction of the bounding box as the stitch rotation.

Next, we use the stitch rotation to define a coordinate system, where the x-axis is parallel to the stitches, and the number of stitch rows increases along the y-axis. For each pixel in the region, we calculate the value $r = \frac{y-y_{min}}{h_{LS}}$, where y is the value on the y-axis, y_{min} is the minimum of y in this region, and h_{LS} is the height of the reference texture for the long-short stitch. The value of v used in texture mapping is obtained directly from the fractional part of r, and the integer part of r represents the number of texture rows between the minimum y position and the current pixel. The u value is decided similarly along the x-axis. The difference is, only the central part of the reference texture can be repeated. Its edge part is always stretched or shrinked to be mapped to the region contour, which is reflected in the UV result (see Fig. 4(b)).

5.2 Satin Stitch Synthesis

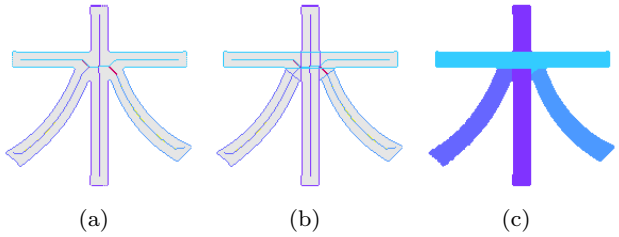
The satin stitch (see Fig. 5(a)) is suitable for striped shapes with shorter width compared with long-short stitch cases. This stitch is commonly used in real embroidery work to represent leaves, petals, text characters, etc. The satin stitch places the stitches one row after another [7,9]. Compared with previous methods [7,9] that generate parallel stitches, the stitches in our system tend to be perpendicular to the tangent of its nearby contour. The procedure is described as follows.

First, we extract the skeleton branches from each region. We use the Zhang-Suen thinning algorithm [30] to obtain the skeleton pixels (see Fig. 6(a)), and organize them into skeleton branches associated with high-level



(a) (b) (c) (d)

Fig. 6 The skeleton branches extraction procedure. (a) Initial skeleton pixels, (b) unexpected skeleton pixels, the yellow of which represent the short noise branch, the red of which represent intersection area, (c) reconnection of the cut-off branches with green pixels, and (d) the resulting skeleton branches, each of which is in a different blue color.



(a) (b) (c)

Fig. 7 The sub-region extraction procedure. (a) Nearest contour pixels for each skeleton branch, (b) contour completion, and (c) sub-regions for each skeleton branch, which are overlapped with each other.

contents, such as a stroke in a text character, or a petal in a flower. Further optimization steps are necessary however, such as abandoning the branches shorter than a given threshold (see the yellow pixel in Fig. 6(b)). Inspired by methods adopted in the field of text stroke separation, we optimize the skeleton tree by removing the pixels which connect three or more pixels to form an intersecting area (see the red pixels in Fig. 6(b)). Pixels connecting located areas can also be merged into these areas. The cut-off tree branches adjacent to the same intersecting area are then reconnected in pairs for optimal pair-wise similarities (see the green pixels in Fig. 6(c)). Using the reconnection result, we traverse the skeleton pixels again to obtain all the skeleton branches (see pixels in different blue colors in Fig. 6(d)).

Next, we assign the region pixels to the adjacent skeleton branches to get the sub-regions. One pixel can be assigned to multiple sub-regions if their branches

intersect at a point near this pixel. We suppose each sub-region consists of a contour surrounding the skeleton branch and region pixels within this contour. To generate a reasonable contour, we search around the skeleton pixels until the nearest contour pixels are collected (see Fig. 7(a)), and complete the contour with line segments or spline curves which connect these original contour pixels (see Fig. 7(b)). After the contour generation step, pixels outside the contours will be assigned to their nearest skeleton branches to ensure the regions are completely separated (see Fig. 7(c)).

In each sub-region, we fit the skeleton branch pixels with a B-spline curve, and cut the skeleton curve into pieces at the interval of the height of the reference texture. The skeleton branch should be extended if the length is not enough for fitting. Then we divide the sub-region into blocks by perpendicular cut lines at the cut points of the curve (see Fig. 8(a)). Each block will be synthesized with one reference texture along the curve which impels the number of stitch rows to grow. We calculate the UV value of each pixel with the help of contour pixel pairs. Each contour pixel pair contains two contour pixels from different sides of the skeleton curve. To match the contour pixel pairs, we sort the pixels on each side according to their relative orders on the contour, and assemble the contour pixels with similar orders into pairs. One-to-many relationships are established in case that the numbers of contour pixels on the two sides are unequal. We can thus determine the value of u of each region pixel along the line connecting the contour pixel pair, and the value of v according to their relative contour orders (see Fig. 5(b)). Note that a pixel can be surrounded by multiple contour pixel pairs. We set the weight to each pair as follows:

$$\omega_{\mathbf{p}} = e^{-\frac{(\|\mathbf{p}-\mathbf{c}_0\|_2+\|\mathbf{p}-\mathbf{c}_1\|_2)\times dist_p}{\|\mathbf{c}_0-\mathbf{c}_1\|_2}}, \tag{7}$$

where $\mathbf{p}$ is the position of a pixel, $\mathbf{c}_0$, $\mathbf{c}_1$ are the contour pixels positions, $\|\cdot\|_2$ is the L_2-norm operator, and $dist_p$ is the distance from $\mathbf{p}$ to the connecting line of the contour pixel pair.

The synthesis results will be badly affected if the cut lines intersect with each other (see Figs. 8(b) and 8(d)). Similar bad cases may arise when the skeletons are cut twice or more by a same cut line, etc. To remedy these improper results, we first try to replace the unexpected cut line (see line 3 in Fig. 8(b)) with an ideal one (see the green line segment in Fig. 8(b)), which is usually generated via the inflection point of the skeleton curve, or the corners on the original region contour detected by the Harris operator. Another remedy method is designed for the case when the skeleton branch is self-crossed (see the blue pixels in Fig. 8(c)), where our system will add temporary contour pixels (see the dark gray pixels in Fig. 8(d)) around the crossing position to adjust the unexpected cut lines (see line 3 and line 7 in Fig. 8(d)) into reasonable ones.

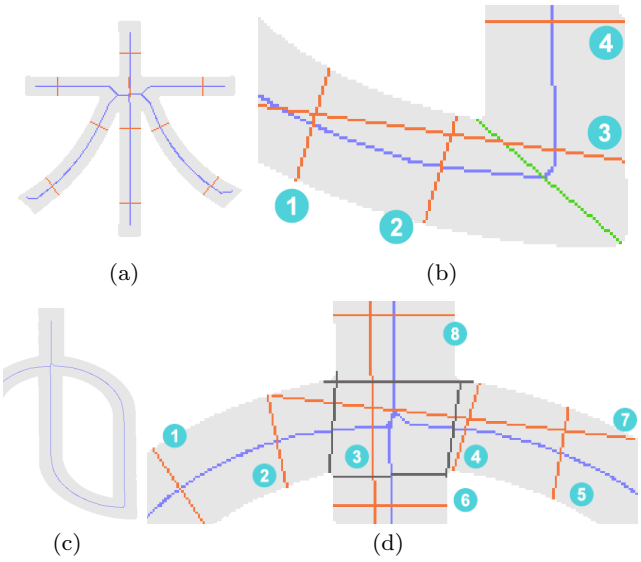


Fig. 8 Block division cases. (a) Expected block division result using the orange cut lines, (b) unexpected cut line case which can be corrected with the green ideal cut line substitute, (c) self-intersection of the skeleton which causes the unexpected cut line case in d, and (d) correction by adding the dark gray temporary contour pixels.

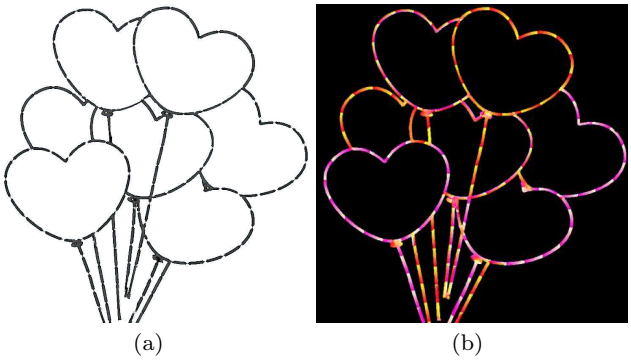


Fig. 9 The edge stitch. (a) Embroidery work produced by our system, and (b) the UV value image as the intermediate result.

5.3 Edge Stitch Synthesis

The edge stitch (see Fig. 9(a)) fills thin lines to emphasize the design's contours, or distinguish its interior details [7,9], where each stitch is synthesized with one reference texture. In addition, the stitch direction varies along the shape, which is similar to the satin stitch. Therefore, the synthesis procedure of the edge stitch can be regarded as a simplified version of the satin stitch, and can be performed as follows.

First, the skeletons of the region are extracted and optimized similar to the aforementioned steps in the satin stitch synthesis. However, we adopt simplified steps in the following phase. By assigning the region pixels to their nearest skeleton branches, we obtain the sub-regions, where the skeleton branch is cut simply according to its pixel length rather than fitting a B-Spline curve. To obtain the blocks, each region pixel is assigned to its nearest skeleton branch piece. After that, each block is synthesized with one reference texture. Finally, we use the line-based 2D-morphing method to deform each reference texture into the shape of the block, with consideration that the edge of the reference texture should be restricted within the region. The morphing result is also stored in the form of UV values (see Fig. 9(b)).

5.4 Embroidery Colorization

Our system provides only one set of reference textures with a specific color for each stitch style. Therefore, it is necessary to recolor the reference textures for each region, respectively. We perform the recolor procedure similar to Chang et al. [5]. In our framework, we additionally modify this algorithm by limiting the target color to a maximum value L to avoid the overexposure defect in white color cases, while considering that even embroidery with white threads tends to be gray.

5.5 Multi-layer Embroidery Handle

In embroidery works containing multiple layers, the background layer should be filled first to make the shapes inside them less irregular, and subsequently, the contents in the foreground layers will correctly overwrite these areas. Previous research [7,9] shows that this technique can greatly improve final visual quality. Since irregular regions with inner contours tend to be assigned with the long-short stitch, we insert the long-short stitch regions into the background layers, and test whether the pixels within their inner contours have completed the synthesis step. If not, all these pixels are merged into the region and are treated the same as the original region pixels in the synthesis step. The synthesis procedures of the other two stitch styles are performed after the long-short stitch synthesis.

Note, that the real stitches may not locate strictly close to each other, and blank areas may exist among them, especially the thin tips of the stitch threads. To fill up these areas, the adjacent regions in the real embroidery works are allowed to be slightly sewn into each other. Therefore, in our system, we dilate each region or each sub-region in different layers, to make them overlap with each other. We represent the seams with transparent pixels in the reference textures, and use the alpha channel to combine different layers. The seams in one layer are likely to be eliminated by the valid pixels in another layer, while the valid pixels may also be unexpectedly overlapped. For this reason, we set all the edge stitch regions in the foreground layers, to ensure that the fine details are not overlapped by the dilatation.

Image Size	**Example**	**Region Count**	**Time(s)**
192×195	Fig. 5(a)	1	0.2429
388×272	Fig. 2	170	0.8791
288×288	Fig. 16(e)	47	0.8957
600×400	Fig. 11(c)	295	6.3943
603×604	Fig. 11(f)	619	5.2857
875×739	Fig. 17(b)	1,117	15.4975
700×932	Fig. 14(b)	1,474	15.1318
1034×2029	Fig. 15(b)	5,374	14.5334
1600×1600	Fig. 19(c)	1,735	21.4142

Table 1 Run time performance measured in seconds.

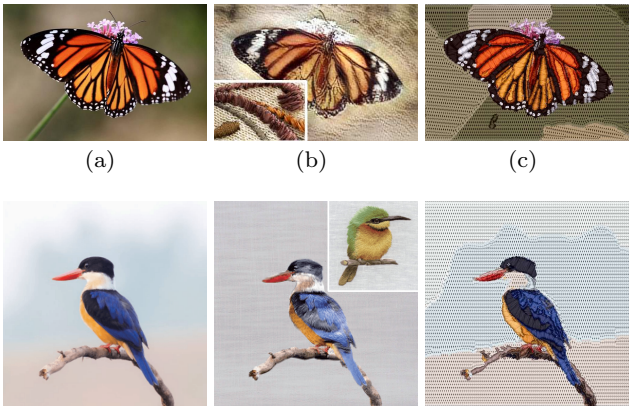
6 Results and Discussion

Our system is implemented on a PC with a 3.60 GHz CPU and 16GB memory. The texture synthesis step of different regions is implemented in parallel via the OpenMP API without GPU implementation. We conduct the experiments on challenging input images, such as real-world photos, cartoon images and illustrations, and our method achieves appealing results. The run time performance of the typical experiments is presented in Table 1. The time cost is influenced mainly by the image sizes and content complexity. Given a 425 × 425 image, which can support a design of $15cm \times 15cm$ with 72 dpi in a common garment logo design case, our system can produce the result within 5 seconds, which is acceptable for designers to receive the interactive feedback. For megapixel cases, especially the complicated images containing more regions, our proposed algorithm takes longer time, which needs to be optimized further but still outperforms the patch-based methods that cost about 2 minutes on a 500 ×400 image [22].

We compare our results to several state-of-the-art image transfer methods. Although the adapted CycleGAN method can be applied in most of the stitches, it fails to capture clear stitches (see Fig. 10(b)). On the contrary, our method is able to generate clear and well-shaped satin stitch stripes (see Fig. 10(c)). In addition, compared to our method (see Fig. 11(c) and 11(f)), other baseline methods [20,23] cannot reproduce

(a) (b) (c)

Fig. 10 Comparison to the CycleGAN method. (a) The original image, (b) the result of CycleGAN [33], and (c) our result.



(a) (b) (c)

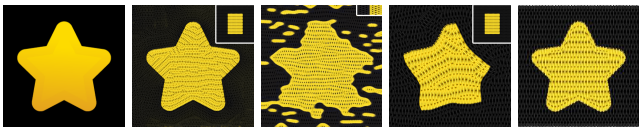
(d) (e) (f)

Fig. 11 Comparison to the two neural style transfer methods. (a)(d) The original image, (b) the result of neural style transfer [20] along with its reference image in the bottom-left corner, (e) the result of neural style transfer [23] along with its reference image in the top-right corner, and (c)(f) our results.

the object and background colors or stitches of the input images faithfully, since their results overly (see Fig. 11(b) and 11(e)) depend on the provided reference images.

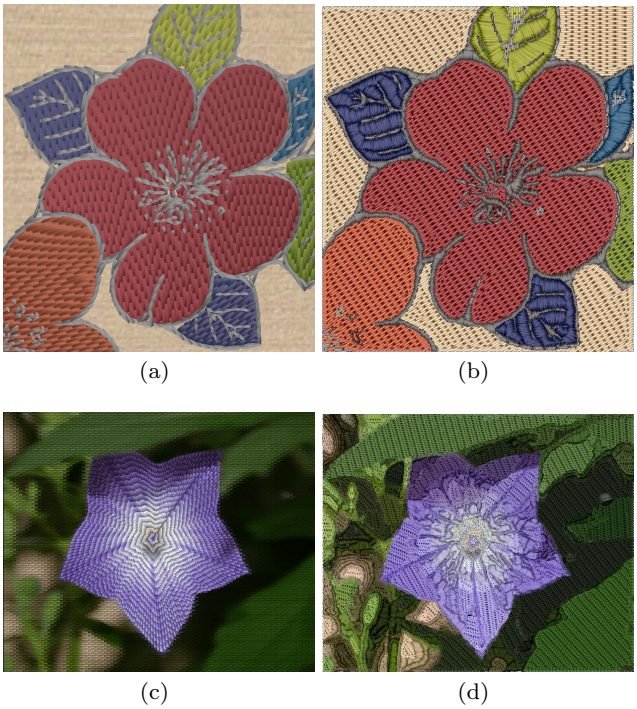
For the stitch cases, the patch-based image synthesis methods [28,32,22] failed to generate compatible stitches between the edge and the center, and thus generate either inconsistent embroidery patterns (see Fig. 12(b)) or totally different ones (see Fig. 12(c)(d)). In contrast, our method addresses this problem by adopting parallel stitch rows (see Fig. 12(e)). Although some image-based embroidery algorithms with manual interactions [7,9] can generate similar visual results with neat stitch placements (see Fig. 13(a) and 13(c)), our results can produce comparable results (see Fig. 13(b)(d)) automatically. In addition, our method can adapt to complicated input images (such as Fig. 14(a) and 15(a)) and generate visually pleasing results (such as Fig. 14(b) and 15(b)), which are difficult to handle by the aforementioned baseline methods.

In summary, our method has the following advantages compared with the previous methods. First, the



(a) (b) (c) (d) (e)

Fig. 12 Comparison to the three patch-based methods. (a) The original image, (b) result of [28] with its reference image in the top-right corner, (c) result of [32] with its reference image in the top-right corner, (d) result of [22] with its reference image in the top-right corner, and (e) our result.



(a) (b)

(c) (d)

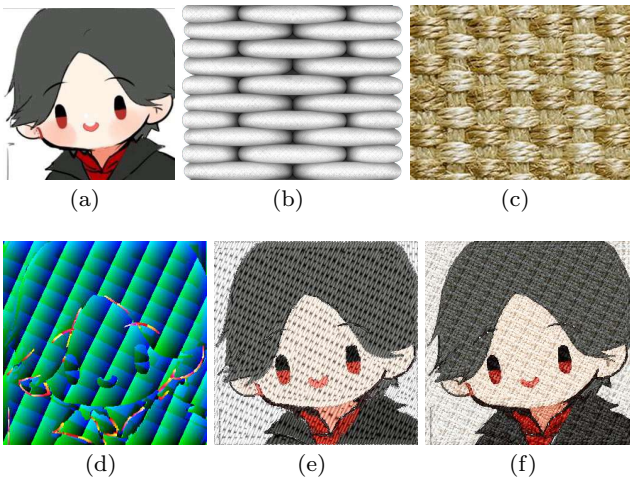
Fig. 13 Comparison to the two image-based interactive methods. (a) Result of Chen et al. [7], (c) result of Cui et al. [9], and (b)(d) our results.

(a) (b)

Fig. 14 Embroidery produced by our framework. (a) Original illustration image, and (b) our result.

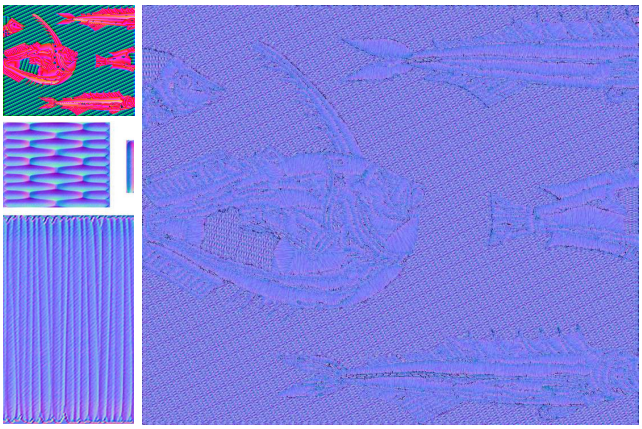
(a) (b)

Fig. 15 Embroidery produced by our framework. (a) Original photography image, and (b) our result.



(a) (b) (c) (d) (e) (f)

Fig. 16 Effects of different reference textures. (a) The original image, (b) the original reference texture for the long-short stitch, (c) another reference texture for the long-short stitch, (d) the UV value image as the intermediate result, (e) the embroidery result of b, and (f) the embroidery result of c.



(a) (b)

Fig. 17 Normal map generation via our system. (a) The UV value image and normal textures, top is the UV value image as the intermediate result, middle-left is the normal reference texture of the long-short stitch, middle-right is that of the edge stitch, bottom is that of the satin stitch, and (b) the normal map result.

stitch length parameters in our system are controllable. Rather than extracting the stitch information from the given reference image in some of the previous methods, our system can obtain the stitch lengths via the actual sizes of the input image and the physical reference textures. Thus, the system will scale the pixel sizes of reference textures to make $\frac{PixelSize_T}{PixelSize_I} = \frac{ActualSize_T}{ActualSize_I}$, where subscript T represents each reference texture, and I represents the input image. The algorithm will use the pixel size to generate stitches with expected lengths.

Second, our system can model different stitches or knitting techniques according to different reference textures (see Fig. 16(b) and (c)), and generate faithful results (see Fig. 16(e) and (f)), which is difficult for other algorithms to perform.

Last but not least, our system can generate the normal maps of the embroidery results (see Fig. 17(b)) based on the UV value image and additionally provided normal textures (see Fig. 17(a)). The normal map can greatly enhance the appearance and details of a low polygon model, and is widely used in modern rendering engines. Other common texture maps (e.g., metallic maps and emission maps) can also be synthesized in a similar way.

After exporting our embroidery work (see Fig. 18(b)) together with the corresponding normal map (see Fig. 18(c)), we can use them to decorate a scarf and achieve visually pleasing preview results (see Fig. 18(d) and (e)) for online garment display. More cases are shown in Fig. 19.

7 Conclusions

Aiming at fast garment design and online display, we have proposed a novel pipeline to synthesize three main types of embroidery images automatically by taking arbitrary images as input. Our method segments the input image into different embroidery regions, and labels them with corresponding stitch styles and colors in the segmentation procedure. Then, we conduct the stitch-

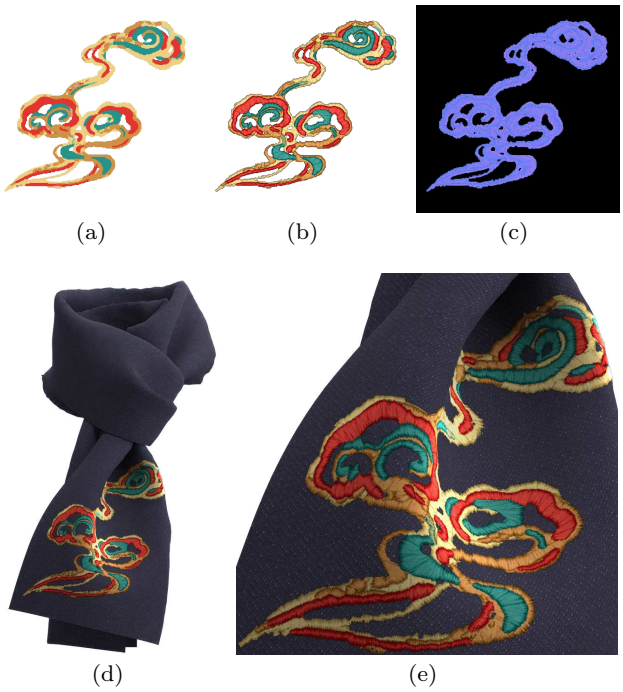


Fig. 18 Applications in the online garment display. (a) The original image, (b) the embroidery result, (c) the normal map result, (d) the rendering image of the embroidery on a scarf, and (e) its close-up detail.

Fig. 19 Rendering images of our embroidery results in different online garment display cases. (a) A beanie, (b) a shirt, (c) a hoddie, and (d) a T-shirt.

style-based texture synthesis with recolored reference textures to produce the visual effects of each stitch style. Compared to state-of-the-art embroidery generation methods, our method is competitive in generating high-quality embroidery images, and can be performed automatically with faithful details. Moreover, experimental results show that our framework can be utilized in digital or real industry applications.

For a 1 megabyte image, our method can take 10 seconds to generate the result. Moreover, we only consider the three most used stitch styles. In the future, we plan to accelerate the processing speed, and incorporate more stitch styles. In addition, we will integrate semantics guidance for better results, which will contribute to appropriate segmentation with respect to aesthetics.

Acknowledgements We thank the anonymous reviewers for their constructive comments. Xiaogang Jin was supported by the National Key R&D Program of China (Grant No. 2017YFB1002600), the National Natural Science Foundation of China (Grant No. 61732015), the Ningbo Major Special Projects of the "Science and Technology Innovation 2025" (Grant No. 2020Z007), and the Key Research and Development Program of Zhejiang Province (Grant No. 2018C01090).

References

1. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B.: Patchmatch: a randomized correspondence algorithm for structural image editing. ACM Trans. Graph. **28**(3), 24 (2009)
2. Barnes, C., Shechtman, E., Goldman, D.B., Finkelstein, A.: The generalized patchmatch correspondence algorithm. In: K. Daniilidis, P. Maragos, N. Paragios (eds.) Computer Vision - ECCV 2010, 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part III, *Lecture Notes in Computer Science*, vol. 6313, pp. 29–43. Springer (2010)
3. Barnes, C., Zhang, F.: A survey of the state-of-the-art in patch-based synthesis. Comput. Vis. Media **3**(1), 3–20 (2017)
4. Barnes, C., Zhang, F., Lou, L., Wu, X., Hu, S.: Patchtable: efficient patch queries for large datasets and applications. ACM Trans. Graph. **34**(4), 97:1–97:10 (2015)
5. Chang, H., Fried, O., Liu, Y., DiVerdi, S., Finkelstein, A.: Palette-based photo recoloring. ACM Trans. Graph. **34**(4), 139:1–139:11 (2015)
6. Chen, T.Q., Schmidt, M.: Fast patch-based style transfer of arbitrary style. CoRR **abs/1612.04337** (2016). URL http://arxiv.org/abs/1612.04337
7. Chen, X., McCool, M., Kitamoto, A., Mann, S.: Embroidery modeling and rendering. In: S. Brooks, K. Hawkey (eds.) Proceedings of the Graphics Interface 2012 Conference, GI '12, Toronto, ON, Canada, May 28-30, 2012, pp. 131–139. ACM (2012)
8. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. IEEE Trans. Pattern Anal. Mach. Intell. **24**(5), 603–619 (2002)

9. Cui, D., Sheng, Y., Zhang, G.: Image-based embroidery modeling and rendering. Comput. Animat. Virtual Worlds **28**(2) (2017)
10. Delong, A., Osokin, A., Isack, H.N., Boykov, Y.: Fast approximate energy minimization with label costs. Int. J. Comput. Vis. **96**(1), 1–27 (2012)
11. Efros, A.A., Freeman, W.T.: Image quilting for texture synthesis and transfer. In: L. Pocock (ed.) Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2001, Los Angeles, California, USA, August 12-17, 2001, pp. 341–346. ACM (2001)
12. Epshtein, B., Ofek, E., Wexler, Y.: Detecting text in natural scenes with stroke width transform. In: The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010, pp. 2963–2970. IEEE Computer Society (2010)
13. Gatys, L.A., Ecker, A.S., Bethge, M.: A neural algorithm of artistic style. CoRR **abs/1508.06576** (2015). URL http://arxiv.org/abs/1508.06576
14. Gatys, L.A., Ecker, A.S., Bethge, M.: Texture synthesis using convolutional neural networks. In: C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, R. Garnett (eds.) Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada, pp. 262–270 (2015)
15. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., Bengio, Y.: Generative adversarial networks. CoRR **abs/1406.2661** (2014). URL http://arxiv.org/abs/1406.2661
16. Hertzmann, A., Jacobs, C.E., Oliver, N., Curless, B., Salesin, D.: Image analogies. In: L. Pocock (ed.) Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2001, Los Angeles, California, USA, August 12-17, 2001, pp. 327–340. ACM (2001)
17. Isola, P., Zhu, J., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pp. 5967–5976. IEEE Computer Society (2017)
18. Jing, Y., Yang, Y., Feng, Z., Ye, J., Yu, Y., Song, M.: Neural style transfer: A review. IEEE Trans. Vis. Comput. Graph. **26**(11), 3365–3385 (2020)
19. Kyprianidis, J.E., Collomosse, J.P., Wang, T., Isenberg, T.: State of the "art": A taxonomy of artistic stylization techniques for images and video. IEEE Trans. Vis. Comput. Graph. **19**(5), 866–885 (2013)
20. Li, X., Liu, S., Kautz, J., Yang, M.: Learning linear transformations for fast arbitrary style transfer. CoRR **abs/1808.04537** (2018). URL http://arxiv.org/abs/1808.04537
21. Ma, C., Sun, Z.: Stitchgeneration: Modeling and creation of random-needle embroidery based on markov chain model. Multim. Tools Appl. **78**(23), 34,065–34,094 (2019)
22. Men, Y., Lian, Z., Tang, Y., Xiao, J.: A common framework for interactive texture transfer. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018, pp. 6353–6362. IEEE Computer Society (2018)
23. Qian, W., Cao, J., Xu, D., Nie, R., Guan, Z., Zheng, R.: Cnn-based embroidery style rendering. Int. J. Pattern Recognit. Artif. Intell. **34**(14), 2059,045:1–2059,045:24 (2020)
24. Qian, W., Xu, D., Cao, J., Guan, Z., Pu, Y.: Aesthetic art simulation for embroidery style. Multim. Tools Appl. **78**(1), 995–1016 (2019)
25. Shen, Q., Cui, D., Sheng, Y., Zhang, G.: Illumination-preserving embroidery simulation for non-photorealistic rendering. In: MultiMedia Modeling - 23rd International Conference, MMM 2017, Reykjavik, Iceland, January 4-6, 2017, Proceedings, Part II, *Lecture Notes in Computer Science*, vol. 10133, pp. 233–244. Springer (2017)
26. Yang, K., Sun, Z.: Paint with stitches: a style definition and image-based rendering method for random-needle embroidery. Multim. Tools Appl. **77**(10), 12,259–12,292 (2018)
27. Yang, K., Zhou, J., Sun, Z., Li, Y.: Image-based irregular needling embroidery rendering. In: H. Qu, W. Chen, P.T. Cox, S. Liu (eds.) The International Symposium on Visual Information Communication and Interaction, VINCI '12, Hangzhou, China - September 27 - 28, 2012, pp. 87–94. ACM (2012)
28. Yang, S., Liu, J., Lian, Z., Guo, Z.: Awesome typography: Statistics-based text effects transfer. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pp. 2886–2895. IEEE Computer Society (2017)
29. Yang, S., Liu, J., Yang, W., Guo, Z.: Context-aware unsupervised text stylization. In: S. Boll, K.M. Lee, J. Luo, W. Zhu, H. Byun, C.W. Chen, R. Lienhart, T. Mei (eds.) 2018 ACM Multimedia Conference on Multimedia Conference, MM 2018, Seoul, Republic of Korea, October 22-26, 2018, pp. 1688–1696. ACM (2018)
30. Zhang, T.Y., Suen, C.Y.: A fast parallel algorithm for thinning digital patterns. Commun. ACM **27**(3), 236–239 (1984)
31. Zhao, Y., Jin, X., Xu, Y., Zhao, H., Ai, M., Zhou, K.: Parallel style-aware image cloning for artworks. IEEE Trans. Vis. Comput. Graph. **21**(2), 229–240 (2015)
32. Zhou, Y., Shi, H., Lischinski, D., Gong, M., Kopf, J., Huang, H.: Analysis and controlled synthesis of inhomogeneous textures. Comput. Graph. Forum **36**(2), 199–212 (2017)
33. Zhu, J., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017, pp. 2242–2251. IEEE Computer Society (2017)