Suitable and Style-Consistent Multi-Texture Recommendation for Cartoon Illustrations

HUISI WU, Shenzhen University, Shenzhen, China
ZHAOZE WANG, Shenzhen University, Shenzhen, China
YIFAN LI, Shenzhen University, Shenzhen, China
XUETING LIU, Shenzhen University, Shenzhen, China
TONG-YEE LEE, National Cheng-Kung University, Tainan, Taiwan

Texture plays an important role in cartoon illustrations to display object materials and enrich visual experiences. Unfortunately, manually designing and drawing an appropriate texture is not easy even for proficient artists, let alone novice or amateur people. While there exist tons of textures on the Internet, it is not easy to pick an appropriate one using traditional text-based search engines. Although several texture pickers have been proposed, they still require the users to browse the textures by themselves, which is still labor-intensive and time-consuming. In this article, an automatic texture recommendation system is proposed for recommending multiple textures to replace a set of user-specified regions in a cartoon illustration with visually pleasant look. Two measurements, the suitability measurement and the style-consistency measurement, are proposed to make sure that the recommended textures are suitable for cartoon illustration and at the same time mutually consistent in style. The suitability is measured based on the synthesizability, cartoonity, and region fitness of textures. The style-consistency is predicted using a learning-based solution since it is subjective to judge whether two textures are consistent in style. An optimization problem is formulated and solved via the genetic algorithm. Our method is validated on various cartoon illustrations, and convincing results are obtained.

CCS Concepts: • **Computing methodologies** → **Texturing;**

Additional Key Words and Phrases: Texture recommendation, texture replacement, cartoon texture, texture style-consistency, texture synthesis

ACM Reference Format:
Huisi Wu, Zhaoze Wang, Yifan Li, Xueting Liu, and Tong-Yee Lee. 2024. Suitable and Style-Consistent Multi-Texture Recommendation for Cartoon Illustrations. *ACM Trans. Multimedia Comput. Commun. Appl.* 20, 7, Article 220 (May 2024), 26 pages. https://doi.org/10.1145/3652518

This work was supported in part by grants from the National Natural Science Foundation of China (62273241 and 62002232) and the National Science and Technology Council (110-2221-E-006-135-MY3 and 111-2221-E-006-112-MY3), Taiwan.
Authors' addresses: H. Wu, Z. Wang, Y. Li, and X. Liu, College of Computer Science and Software Engineering, Shenzhen University, No. 3688 Nanhai Road, Nanshan District, Shenzhen, Guangdong, 518060, China; e-mails: hswu@szu.edu.cn, 2070276045@email.szu.edu.cn, 1810272030@email.szu.edu.cn, xtliu@szu.edu.cn; T.-Y. Lee, Dept. of Computer Science and Information Engineering, National Cheng-Kung University, No. 1, University Road, Tainan, 70101, Taiwan; e-mail: tonylee@mail.ncku.edu.tw.

ACM 1551-6865/2024/05-ART220
https://doi.org/10.1145/3652518

1 INTRODUCTION

Cartoon illustrations are broadly used in daily lives, such as book illustrations, digital cartoons, and product designs. During the creation of cartoon illustrations, drawing textured objects is the most challenging and labor-intensive step due to the difficulty in drawing the details of the textures. To this aim, a number of digital texture creation systems [2, 13] and commercial software (e.g., Adobe Photoshop, Adobe Illustrator, Coral Painter) have been released. While these systems and softwares can help in the texture design process with different design tools, creating a visually pleasant texture is still not easy even for veteran artists, let alone non-professional or novice artists. Due to the difficulty in painting new textures, several texture datasets have been released so that artists can directly use them for their artworks. This significantly reduces the effort needed by artists to create textured cartoon illustrations. However, picking a suitable texture from a large texture dataset is not easy. The artist needs to go through the textures one by one until he/she finds a good one, which is extremely time-consuming. The case is even more complicated when there are multiple textured objects. One needs to consider not only the suitability of each texture but also the style-consistency among multiple textures. An automatic texture recommendation system is vastly needed for adding textures to cartoon illustrations, especially when multiple textures are needed.

Unfortunately, texture recommendation is rarely studied in the existing literature. To help artists pick textures, various interactive texture pickers have been proposed [12, 24]. While these methods assist the user in visualizing the textures in an easier-to-pick way, users still need to interactively pick the texture for each textured object one by one by themselves. Methods have also been proposed to replace color regions with user-specified textures [25] or computer-generated textures [20]. However, these methods only aim at representing regions of different colors with different pre-defined textures. They do not solve the problem of automatically picking suitable and style-consistent textures from a large texture dataset as in our texture recommendation application. Recently, a learning-based approach has also been proposed to predict manga textures for manga characters [33]. However, this method can only be used for a limited number of textures, which is hardly the case for colorful cartoon illustrations. Besides, this method considers neither suitability nor style-consistency about the textures. Different from the existing methods, this is the first attempt in studying the suitability of each texture, as well as the style-consistency of multiple textures, when applied in cartoon illustrations.

In this article, an automatic multi-texture recommendation system is proposed to recommend a group of suitable and style-consistent textures for the user-specified textured objects in a cartoon illustration. Our research mainly focuses on two aspects. First, a metric is needed to automatically measure whether a texture is suitable for cartoon illustrations. To the best of our knowledge, there is no existing texture dataset tailored for cartoons, so it requires searching for appropriate textures from common texture datasets. However, many of these textures are photo-realistic and unsuitable for cartoon illustrations (Figure 1(a)). Besides, many of them do not have good synthesizability and are unable to synthesize visually pleasant larger textures regions (see Figure 1(b)). So it is very important to analyze whether a texture is suitable for cartoon illustrations and has good synthesizability so that it can automatically find the ones from the large texture dataset for our purpose. Second, it requires a metric to measure the style-consistency of multiple textures. While cartoon illustrations may use different textures to depict different objects, textures used in the same image are usually with the same style, especially when they are used for the same object. Here, it should be emphasized that style-consistency is quite subjective—that is, style-consistent textures might be different from each other in content and color but are still visually pleasant when placed together in one color illustration (see, e.g., Figure 1(c)). Similarly, textures with similar colors may still be different in style and therefore unlikely to be used in one cartoon illustration (see, e.g.,

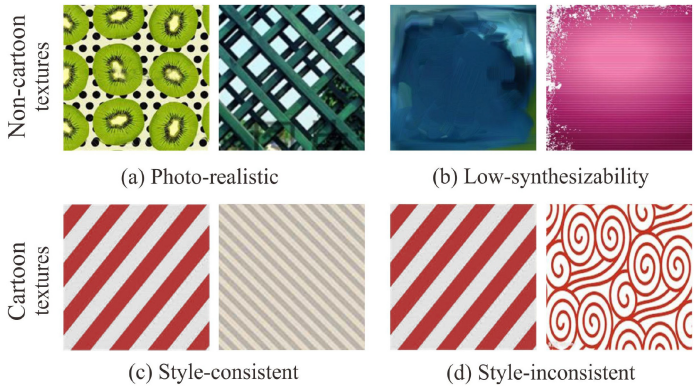

Fig. 1. (a) Photo-realistic textures are unsuitable for cartoon illustrations. (b) Low-synthesizability textures are unsuitable due to the difficulty in synthesizing high-quality texture images. (c) The two textures are visually style-consistent even if they have different colors and patterns. (d) The two textures are style-inconsistent, so they are unlikely to be used in the same cartoon illustration.

Figure 1(d)). So it is also important for us to analyze the style-consistency of multiple textures to recommend style-consistent textures for one cartoon illustration.

Our system first segments the input cartoon illustration into multiple color regions, then asks the user to choose one or several regions to replace with textures. Our system will automatically recommend a group of textures for the user-specified regions. To do so, textures the most suitable for each color region can be searched in terms of synthesizability, cartoonity, and region fitness. Synthesizability ensures the texture to be synthesizable so that high-quality texture images can be synthesized. Cartoonity ensures the texture to be cartoonic. Region fitness ensures the texture to fit the region in color and scale. With the retrieved suitable textures, it can already form a set of potential texture groups (groups of textures) and measure the style-consistency for each group. Since style-consistency is quite subjective and hard to measure based on low-level features, it can estimate the style-consistency by learning from a manually labeled training dataset using deep **Convolutional Neural Networks (CNNs)**. However, due to the large number of potential texture groups, a brute-force search would be intractable. So an optimization-based approach is further proposed to efficiently find the optimal suitable and style-consistent texture group. While our system recommends one optimal texture group, a list of texture groups can be provided for user selection when needed.

Our contributions can be summarized as follows:

— We propose the first, to the best of our knowledge, automatic multi-texture recommendation system that can recommend a group of suitable and style-consistent textures for a cartoon illustration, which can significantly reduce the cost for the artists in exploring the best combination from a large texture dataset.
— We propose a novel metric to measure the suitability of a texture for a color region based on synthesizability, cartoonity, and region fitness, which represent three most important characteristics in recommending suitable and style-consistent textures for a cartoon illustration.
— We propose a novel learning-based approach to measure the style-consistency of a group of textures, which significantly minimizes the subjective factors in recommending multiple textures.

2 RELATED WORK

The related work can be roughly classified into texture recommendation, texture synthesizability analysis, cartoon texture analysis, and CNN-based texture analysis.

2.1 Texture Recommendation

To the best of our knowledge, texture recommendation is rarely studied in the existing literature. Qu et al. [25] made the first attempt in automatically replacing the color regions in a color image with user-selected manga screening textures. Based on this work, Li and Mould [20] further proposed to replace color regions with computer-generated textures. While both methods could replace color regions with textures, they only tailored for replacing regions of different colors with different textures. Neither of them analyzed the suitability or style-consistency of textures. To help users pick textures in an intuitive way, Pang [24] proposed an intuitive texture picker where textures are arranged in a 2D plane where similar textures are spatially near each other. Ishibashi [12] proposed an interactive texture picker where the system can search for a set of textures based on user interaction and evolutionary computation. While these two methods help users pick textures in intuitive ways, users still need to interact with the system and go through a large number of textures to pick a group of suitable textures, which is time-consuming and labor-intensive.

Recently, Tsubota et al. [33] proposed to learn the texture label for a manga character line drawing and synthesize the textures based on the learned label. But this method considers neither texture suitability nor style-consistency, and can only be used for a limited number of pre-defined textures. In sharp contrast, in this article, we make the first attempt in automatically recommending a group of textures from a large texture dataset for a set of user-specified color regions based on suitability and style-consistency.

2.2 Texture Synthesizability Analysis

To generate cartoon illustrations with high-quality textures, it is important that the recommended textures are highly synthesizable. Dai et al. [6] proposed a learning-based approach to predict the synthesizability of a texture based on four features: textureness, homogeneity, repetitiveness, and irregularity. Later, methods were proposed to extract an optimal texture patch from an input image based on these synthesizability features [16, 34, 42]. While these four features can well describe the synthesizability of a texture, the values of the features are generally not bounded, which is uncontrollable when it is combined with other features, such as style-consistency. Therefore, it requires to propose new synthesizability metrics where values are bounded and therefore can be normalized. Recently, Yang et al. [39] proposed to predict the synthesizability of dynamic texture samples via a learning-based approach. But this method is tailored for dynamic textures only and cannot be directly applied in our case.

2.3 Cartoon Texture Analysis

So that the recommended textures are suitable for cartoon illustrations, it is important to measure whether the textures are cartoonic. Kopf and Lischinski [14] proposed to model printed cartoon illustrations where the textures are composed by CMYK dots. Yao et al. [40] proposed to model the manga textures that only consist of simple screentones, such as dots, stripes, and grids. To segment texture regions from manga images, Qu et al. [26] proposed to extract the Gabor features of the input image and then segment the texture region using the level-set algorithm. Liu et al. [22] improves this method in adopting the relative total variation to extract more precise texture regions boundaries. Given an input color region and a user-specified texture, Sýkora et al. [49] proposed

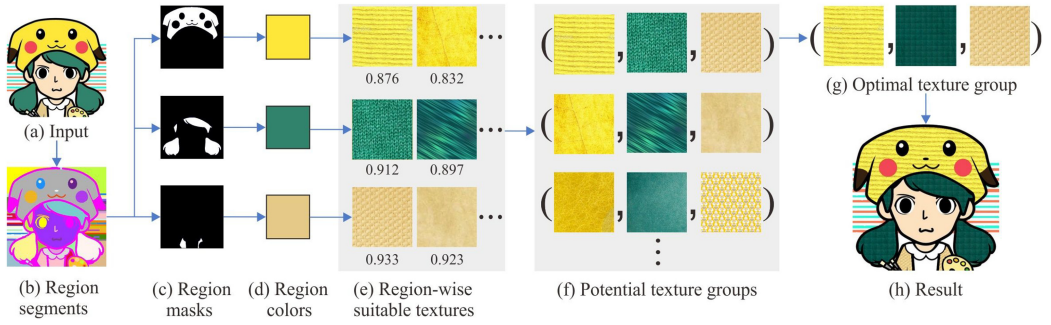

Fig. 2. System overview. Given an input cartoon illustration (a), it first segments the image into color regions (b) and asks the user to select the regions to be replaced with textures (c). For each color region, it extracts the major color (d) and ranks the textures based on their suitability scores (e). Then it can form the potential texture groups and measure the style-consistency for each group (f). The optimal group is then extracted via an optimization approach (g). The optimal textures are finally used for synthesizing the textured output image (h).

to synthesize a pseudo-3D region with textures and lighting. From these research works, it can be concluded that textures used in cartoon illustrations are generally regular and contain a limited number of colors. Therefore, it requires measuring the cartoonity of a texture based on regularity and color sparsity.

2.4 CNN-Based Texture Analysis

Recently, deep CNNs have been broadly explored in computer graphics and computer vision research fields. In terms of texture analysis, deep learning has already been applied in texture recognition [5, 37, 41, 44], texture synthesis [4, 7, 28, 29], texture classification [1, 3, 27], texture segmentation [5, 8, 32, 35], texture smoothing [48], texture optimization [21, 47], texture colorization [46], and texture interpolation [38, 43]. While deep learning has proved to be useful in the preceding texture analysis applications, none of the existing methods are tailored for analyzing the style-consistency between two textures. In this article, we propose to manually label a dataset consisting of both style-consistent and style-inconsistent textures for training an efficient deep neural network to predict the style-consistency between two textures.

3 OVERVIEW

Our multi-texture recommendation system is illustrated in Figure 2. Given an input cartoon illustration (see Figure 2(a)), we first segment the image into multiple regions where pixels in each region have similar colors (see Figure 2(b)). We adopt the cartoon segmentation method proposed by Wu et al. [36] which groups pixels via linear iterative clustering and adaptive region merging. The user can then specify a group of regions that are intended to be replaced by textures (see Figure 2(c)). Then, for each user-specified region, it can extract the main color by finding the most frequent color (see Figure 2(d)). The reason for not using the average color is to avoid the influence of boundary pixels whose colors may mildly deviate from the region's main color.

To find a group of suitable and style-consistent textures to replace the user-specified regions, it can first analyze the suitability of the potential textures for each color region based on three terms: synthesizability, cartoonity, and region fitness. Then, for each region, it can sort the textures based on the suitability scores where textures with higher scores are likely to be adopted to replace the corresponding color region (see Figure 2(e)). The formulation of the suitability measurement is detailed in Section 4.

Suitability alone is not enough for recommending a group of textures. To output a visually pleasant textured cartoon illustration, the recommended textures should be consistent with each other in style. Therefore, the potential texture groups can be first formed based on suitability (see Figure 2(f)). Then the style-consistency can be measured between every two textures via a learning-based approach. The style-consistency for a group of textures will then be measured as the averaged mutual style-consistency between every two textures. The detailed formulation of the style-consistency measure will be presented in Section 5.

With the defined suitability and style-consistency measurements, the optimal texture group with the highest suitability and style-consistency score can be found for synthesizing the result image (see Figure 2(g) and (h)). However, due to the large number of textures, an even larger number of potential texture groups may be formed. A brute-force search would be extremely time-consuming and almost intractable. Therefore, the optimal texture group can be searched by forming an optimization problem and solved using the genetic algorithm. The details of our optimization formulation are discussed in Section 6. To synthesize the resultant textured cartoon illustration, the Graphcut method [15] with CUDA speed-up is adopted to synthesize texture regions from the texture exemplars in the optimal texture group.

Unlike the previous methods which neither consider the texture suitability nor preserve the style-consistency, which can only be used for a limited number of pre-defined textures, we develop a powerful texture recommendation system, which can efficiently recommend a group of suitable textures based on style-consistency and texture suitability.

4 SUITABILITY OF TEXTURES FOR A CARTOON REGION

To find a group of suitable textures for the user-specified color regions, a suitability metric can be first proposed to measure how suitable a texture is for replacing a color region. The suitability of a texture for a color region can be defined based on three terms: synthesizability, cartoonity, and region fitness.

4.1 Synthesizability

Synthesizability measures how well a texture can be resynthesized by learning only from this texture exemplar. Since the size of the user-selected region may be larger than the size of the texture exemplar in the texture dataset, it is important to synthesize high-resolution and high-quality texture images based on the picked texture exemplar. In other words, the picked textures should be of high synthesizability. Similar to the previous method [6], it can also adopt textureness, homogeneity, and repetitiveness to measure the synthesizability of a texture. However, the homogeneity and repetitiveness metrics proposed by Dai et al. [6] are not normalized, which may result in extremely large values and affect the calculated synthesizability. Therefore, new homogeneity and repetitiveness metrics are normalized to the range of $[0, 1]$.

4.1.1 Textureness. The textureness feature is used to ensure that the picked texture images are real texture exemplars. Given a relatively small dataset, deep learning methods may greatly increase the computational cost and cause over-fitting due to the complexity of the learned texture features. Therefore, it can directly adopt the textureness feature defined in the work of Dai et al. [6]. To distinguish texture images from non-texture images, a linear SVM is trained based on the GIST features [23] of an image dataset. The positive samples in the dataset are from the UIUC texture dataset [17] and our collected 1,873 cartoonic texture exemplars from the Internet, whereas the negative samples are from the 15-Scene dataset [18]. Given any texture image t, it can feed the GIST feature of t to the trained linear SVM, and the output classification score will be denoted as the textureness $T(t)$ of t. Since the output of SVM is naturally a number in the range $[0, 1]$, the range of textureness is also $[0, 1]$. Textures of different textureness scores are presented in Figure 3(a).

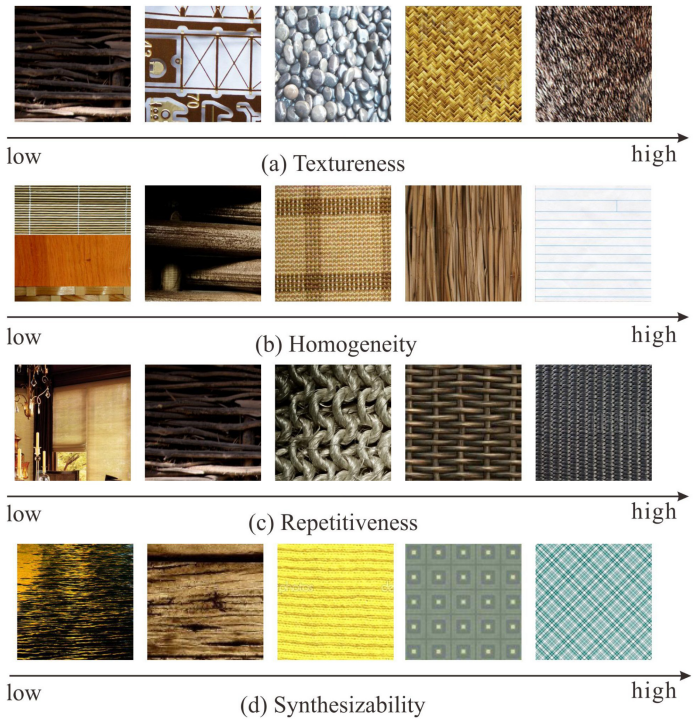

Fig. 3. Textures of different textureness, homogeneity, repetitiveness, and overall synthesizability scores.

4.1.2 Homogeneity. The homogeneity feature ensures that the picked texture images are homogeneous and therefore can be used to synthesize high-resolution and high-quality texture images. Similar to Dai et al. [6], it can also define the homogeneity of a image as the visual similarity between two randomly chosen local regions of the image. To measure the visual similarity between two regions, we first collect 30 dominant patch features from all 10×10 patches in the image via k-means clustering, and then generate the histogram of dominant patch features for each region by sampling all 10×10 patches in this region. The visual dissimilarity between two regions q_1 and q_2 can then be defined as the Euclidean distance between their histograms of dominant patch features $d\left(h(q_1), h(q_2)\right)$, where $h(q)$ is the histogram of dominant patch features of region q and $d(\cdot, \cdot)$ is the Euclidean distance operator. Finally, we propose a new homogeneity metric of a texture t by picking K pairs of regions and average their visual similarity as

$$H(t) = 1 - \frac{1}{K} \sum_{k=1}^{K} d\left(h(q_1^k), h(q_2^k)\right). \tag{1}$$

Here, $(q_1^1, q_2^1), (q_1^2, q_2^2), \ldots, (q_1^K, q_2^K)$ are the randomly picked K pairs of regions from t where the resolution of the regions is $1/3$ of the original texture image. K is empirically set to 80 in all of our experiments. The range of $H(t)$ is $[0, 1]$, where a larger value indicates better homogeneity. Textures of different homogeneity scores are presented in Figure 3(b).

4.1.3 Repetitiveness. Textures are commonly described as visual surfaces made up of repeated patterns that exhibit visual similarity. In earlier studies, the power spectrum in FFT features is closely linked to auto-correlation, where the periodic patterns exhibit a strong peak in the

auto-correlation function. Our proposed metric is a similar metric to measure the repetitions defined in the spatial domain. Similar to Dai et al. [6], it can first compute the **Normalized Cross Correlation (NCC)** matrix of the image, and then measure the repetitiveness of the image based on the NCC matrix. First, for any moderate-sized regions in the NCC matrix, the difference between its maximum value and minimum value should be large. Second, for a set of randomly picked moderate-sized regions, the minimum values of these regions should be similar to each other. Based on these criteria, a new repetitiveness metric is defined for a texture image t based on K randomly picked regions as

$$R(t) = \left(\frac{1}{K}\sum_{k=1}^{K}\left(Max(q^k) - Min(q^k)\right)\right) \times \left(1 - \frac{1}{K}\sum_{k=1}^{K}\left(\left|Min(q^k) - \overline{Min(q^k)}\right|\right)\right). \tag{2}$$

Here, $q^1, q^2, \ldots, q^K$ are the randomly picked K regions from the NCC matrix of t where the resolution of the regions is 1/5 of the original texture image. $Max(\cdot)$, $Min(\cdot)$, and $\bar{\cdot}$ are the maximum, minimum, and mean operators respectively. K is empirically set to 80 in all of our experiments. The range of $R(t)$ is $[0, 1]$, where a larger value indicates better repetitiveness. Textures of different repetitiveness scores are presented in Figure 3(c).

4.1.4 Overall Synthesizability. With the formulated metrics of textureness, homogeneity, and repetitiveness, we can measure the overall synthesizability of a texture image t by combining textureness, homogeneity, and repetitiveness as

$$S_{\text{syn}}(t) = \frac{T(t) + H(t) + R(t)}{3}. \tag{3}$$

Since $T(t)$, $H(t)$, and $R(t)$ are all in the range of $[0, 1]$, the overall synthesizability $S_{syn}(t)$ is also in the range of $[0, 1]$. In our experiments, it can empirically adopt an equal weighting mechanism to obtain a relative better synthesizability performance. It is noteworthy that textureness, homogeneity, and repetitiveness capture different properties for a texture, where our experiments also demonstrated that each of them is important to produce a texture result with a relative better synthesizability. Textures of different overall synthesizability scores are presented in Figure 3(d). Compared with previous methods, our proposed synthesizability metrics have bounded values, which can be normalized to better measure the suitability of textures, and also provide convenience for us to control the integrated features.

4.2 Cartoonity

Cartoonity measures how likely a texture can be adopted in cartoon illustrations. While to the best of our knowledge there is no existing literature that focuses on studying the characteristics of cartoons, it can be concluded from numerous real-world examples that textures used for cartoons are usually regular and composed of only a few colors. We refer to the first feature as regularity and the second as color sparsity.

4.2.1 Regularity. The regularity feature ensures that the picked texture image is regular or near-regular so as to be suitable for cartoon illustrations. It can adopt the Ensemble Composition (EC) method [6] to measure the regularity of a texture t, and the calculated regularity score is denoted as $\tilde{G}(t) = \sum_{k=1}^{K}(\sum_{i \in = q^k} D_i(l_i) + \lambda \sum_{\{i,j\} \in = N} V(l_i, l_j))$, where $D_i(l_i)$ and $V(l_i, l_j)$ are the cost function and smoothing term, respectively. However, $\tilde{G}(t)$ is not normalized in the work of Dai et al. [6], so it can further adopt the negative exponential function to normalize $\tilde{G}(t)$ as

$$G(t) = 1 - \exp(-w_G \cdot \tilde{G}(t)). \tag{4}$$

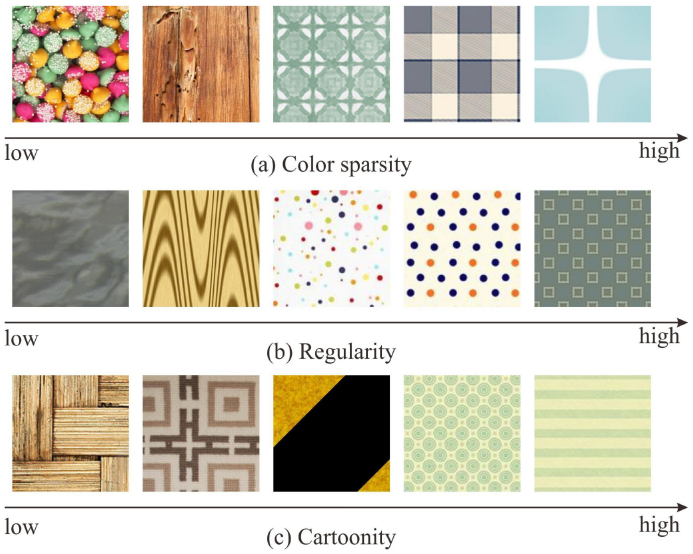

Fig. 4. Textures of different regularity, color sparsity, and overall cartoonity scores.

After normalization, the values of $G(t)$ are in the range of $[0, 1]$. The coefficient w_G is empirically set to 5 in our experiments to make the regularity scores of all textures more evenly distributed in the range of $[0, 1]$. Textures of different regularity scores are presented in Figure 4(a).

4.2.2 Color Sparsity. The color sparsity feature ensures that the picked texture image is composed of only a few colors, which is usually the case in cartoon illustrations. To count the number of colors in a texture image t, it can divide the RGB color space into $16 \times 16 \times 16 = 4{,}096$ bins and count the number of pixels for each bin. Since the colors of the pixels may be affected by anti-aliasing and JPEG compression, it can count the number of bins that contain at least 50 pixels as the number of colors $n_c(t)$ of t. Finally, the color sparsity metric of texture t is formulated using the negative exponential function as

$$N(t) = \exp(-w_N \cdot n_c(t)), \tag{5}$$

where w_N is empirically set to 0.01 so that textures with different number of colors are distinguishable based on the color sparsity metric. Obviously, the value range of $N(t)$ is between 0 and 1. Textures of different color sparsity scores are presented in Figure 4(b).

4.2.3 Overall Cartoonity. Similar with synthesizability, the overall cartoonity of a texture image t can be measured by combining regularity and color sparsity as

$$S_{\text{car}}(t) = \frac{G(t) + N(t)}{2}. \tag{6}$$

The values of the overall cartoonity $S_{car}(t)$ is in the range of $[0, 1]$. The regularity and color sparsity features contribute equal weights to the overall synthesizability metric. Textures of different overall cartoonity scores are presented in Figure 4(c).

4.3 Region Fitness

Besides synthesizability and cartoonity requirements, several region-specific requirements for each specific region need to be considered. First, during the creation of cartoon illustrations, the

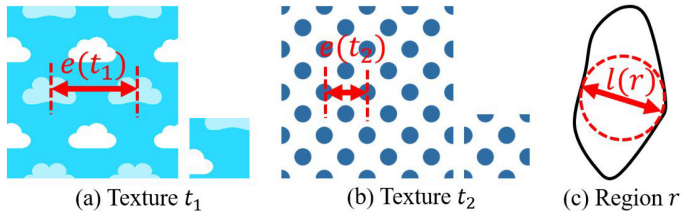

Fig. 5. (a) A large-scale texture which is not suitable for small regions. (b) A small-scale texture which can be used for small regions. (c) A region with its maximal fit circle. $e(t)$ refers to the distance between two primitives of a texture image. $l(r)$ refers to the diameter of the maximum fitting circle in the target region.

artists usually fill each region with a solid color before picking a suitable texture to ensure that colors of all regions are harmonious with each other. Therefore, it requires the color of the texture to be similar with the color picked by the artist. This feature is referred to as color similarity. Second, the scale of the picked texture should be small enough for the corresponding region for visual pleasantness. For example, the scale of the texture in Figure 5(a) is large and not suitable for small regions, whereas the scale of the texture in Figure 5(b) is small and can be used for small regions. This feature can be defined as scale fitness.

4.3.1 Color Similarity. To measure the color similarity between a texture t and a region r, we can first extract the main color $c(r)$ of this region r by finding the most frequent color. Then the color similarity can be measured as the pixel-wise difference between the texture image t and region r's main color $c(r)$, which is normalized by the negative exponential function as

$$C(t, r) = \exp\left(-w_C \cdot \frac{\sum_{p \in t} d(c(p), c(r))}{|t|}\right). \tag{7}$$

Here, p is a pixel in t, $c(p)$ is the color of pixel p, $|t|$ is the number of pixels of t, and w_C is empirically set to 0.008 in all of our experiments. $C(t, r)$ is in the range of $[0, 1]$. To show the effectiveness of our color similarity feature, the textures with high color similarity scores based on main colors are plotted in Figure 6.

4.3.2 Scale Fitness. To measure whether the scale of a texture t fits a region r, it requires that the diameter of the maximum fit circle $l(r)$ of region r (see Figure 5(c)) is larger than twice the distance between neighboring repetitive elements $e(t)$ of texture t (see Figure 5(a) and (b))—that is, $l(r) > 2 \cdot e(t)$. This is to ensure that the texture's repetitive element is repeated at least twice in the region so that the texture pattern is still observable. To measure the distance between neighboring repetitive elements $e(t)$ of texture t, we propose to approximate $e(t)$ based on the repetitiveness score of t as

$$e(t) = \frac{\sqrt{|t|}}{10 \cdot R(t)}. \tag{8}$$

The reason that we do not detect the repetitive elements is to cover the irregular textures. Based on the defined $e(t)$, it can then measure the scale fitness of a texture t for a region r as

$$F(t, r) = \begin{cases} 1, & l(r) > 2 \cdot e(t) \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

The score of scale fitness is either 0 or 1, so it lays a hard constraint on the scale of the picked texture to ensure that the texture pattern is observable after filling in the region. Recommended textures based on same query color but different scales are shown in Figure 7.

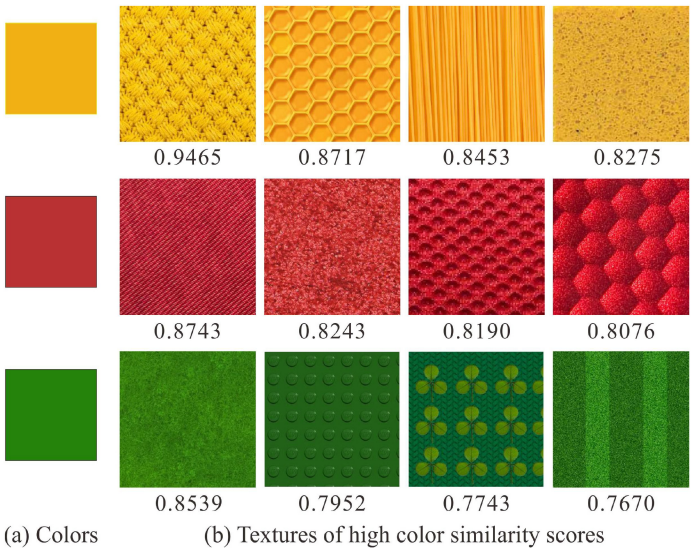

Fig. 6. Texture recommendation based on suitability (without scale fitness).

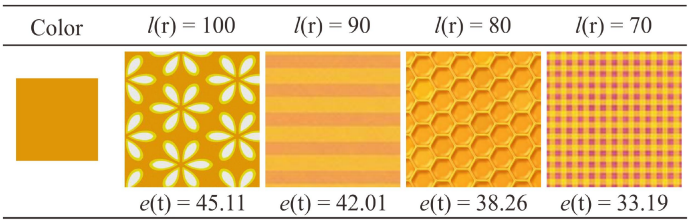

Fig. 7. Texture recommendation based on region fitness with different scales. l(r) denotes the scale of the region (i.e., the diameter of the maximum fitting circle). e(t) denotes the scale of the texture (i.e., distance between two primitives).

4.3.3 Overall Region Fitness. We measure the overall region fitness of a texture image t for a region r as

$$S_{\text{reg}}(t, r) = C(t, r) \cdot F(t, r). \tag{10}$$

When the scale of the texture fits the region ($F(t, r) = 1$), the overall region fitness is fully decided by the color similarity. On the contrary, when the scale of texture does not fit the region ($F(t, r) = 0$), the overall region fitness is 0.

4.4 Overall Suitability

With the preceding feature terms, we can formulate the overall suitability of a texture t for a region r as

$$S(t, r) = S_{\text{syn}}(t) \cdot S_{\text{car}}(t) \cdot S_{\text{reg}}(t, r). \tag{11}$$

To validate the effectiveness of the three feature terms, we show the recommended textures of an input image with different suitability measurements in Figure 8. Without synthesizability, the picked textures are more similar with the color regions in color, but the synthesizability of the picked textures are not satisfying. Without cartoonity, the color of the input can also be preserved, but the extracted textures are usually photo-realistic. Without region fitness, the similarity scores

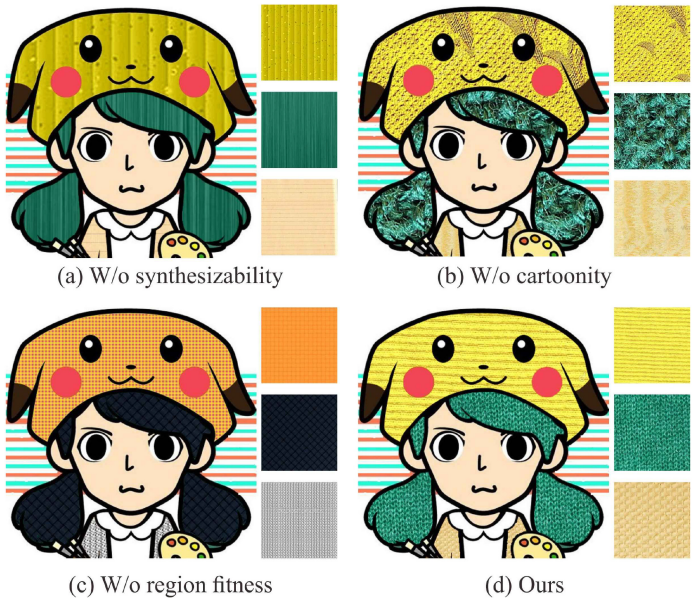

Fig. 8. Ablation study on synthesizability, cartoonity, and region fitness. Note that style-consistency is not enforced in this experiment.

of the textures are fixed since synthesizability and cartoonity are not related to regions, so we pick the three highest-score regions and use them to replace the three regions, respectively. As we can observe, both colors and scales of the picked textures may not fit the original color region. In sharp comparison, with all three feature terms, even though the colors of the textures may slightly deviate from the regions' colors, the picked textures are generally synthesizable, cartoonic, and fitted for the region.

5 STYLE-CONSISTENCY OF MULTIPLE TEXTURES

Suitability alone is not enough for recommending a group of visually pleasant textures. When artists pick textures, they consider not only whether the textures are suitable for each region but also whether the textures are consistent with each other in terms of style. For example, in Figure 9, while both textures t_{1A} and t_{1B} are suitable for color c_1 and both textures t_{2A} and t_{2B} are suitable for color c_2, texture pairs (t_{1A}, t_{2A}) and (t_{1B}, t_{2B}) are style-consistent, which artists may use for the same color illustration, but texture pairs (t_{1A}, t_{2B}) and (t_{1B}, t_{2A}) are style-inconsistent, which artists are unlikely to use for the same color illustration. Therefore, we further measure the style-consistency of a group of textures to ensure that they are visually harmonic with each other.

To measure the style-consistency of a group of textures, it can first adopt a learning-based approach to measure the style-consistency between every two textures, then measure the style-consistency for the whole group with more textures.

5.1 Style-Consistency between Two Textures

As we have discussed, style-consistency between textures is quite subjective and cannot be easily measured based on low-level features. Even if two textures have similar colors, scales, and regularity, it does not mean that they are visually consistent in style. Interestingly, we find that even though different artists may pick different textures for the same color region, they usually

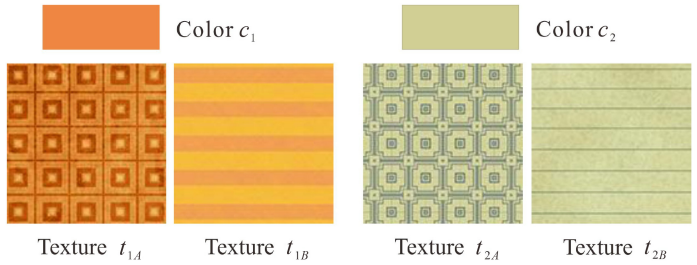

Fig. 9. Examples of style-consistent and style-inconsistent texture pairs. Both textures t_{1A} and t_{1B} are suitable for color c_1. Both textures t_{2A} and t_{2B} are suitable for color c_2. While texture pairs (t_{1A}, t_{2A}) and (t_{1B}, t_{2B}) are style-consistent, texture pairs (t_{1A}, t_{2B}) and (t_{1B}, t_{2A}) are style-inconsistent.

make similar decisions when asking whether two textures are consistent in style—that is, if they would use both textures in the same cartoon illustration. Based on this finding, we propose to predict whether two textures are style-consistent by learning from artists' decisions. Concretely speaking, the artists are first asked to label the style-consistency for a set of texture pairs as the training data. Then we predict the style-consistency for any texture pair by training a supervised CNN based on the prepared training data.

5.1.1 Training Data Preparation. To prepare the texture dataset, we first collect 21,302 texture exemplars from the synthesizability dataset [6] and 1,000 texture exemplars from the UIUC dataset [17]. However, we found that most textures collected this way are photo-realistic and unsuitable for cartoon illustrations. Therefore, we further collect 1,873 cartoonic texture exemplars from the Internet. A total of 24,175 texture exemplars are collected.

Texture exemplars collected this way are of different appearances and qualities, which complicates the learning of the neural network. For better learning performance, we filter out textures that are not synthesizable or not cartoonic, and train our style-consistency prediction network based on texture pairs with high synthesizability and cartoonity only. In particular, for any texture t, if $S_{\text{syn}}(t) < \theta_{\text{syn}}$ or $S_{\text{car}}(t) < \theta_{\text{car}}$, we regard this texture as not synthesizable or nor cartoonic and remove it from the collected texture dataset. We empirically set the thresholds θ_{syn} and θ_{car} to 0.5 and 0.6, respectively. After filtering, 548 texture exemplars are remained.

Then we can prepare the training data for our style-consistency prediction network. We first randomly pick 2,000 texture pairs from the collected texture dataset and ask the artists to label whether they will use the two textures in the same cartoon illustration. We invite 10 artists and ask all artists to label all of the prepared texture pairs. The reason we let multiple artists label the same dataset is to minimize the subjectivity of personal preferences. We only include texture pairs if 80% of the artists (i.e., 8 artists) make the same labeling. After labeling, we obtain 112 style-consistent pairs and 1,714 style-inconsistent pairs. As observed, most of the labeled pairs are style-inconsistent. To balance between positive and negative data, we attempt to prepare some style-consistent texture pairs by first categorizing the 548 textures into 12 style groups (dots, stripes, grids, semi-irregular, irregular, pictorial, etc.). Then we randomly extract 2,000 texture pairs where both textures are from the same style group and ask the artists to do the labeling the same as previously. A total of 1,148 style-consistent texture pairs are obtained. Finally, we have 2,974 labeled texture pairs in our final training dataset, including 1,260 style-consistent pairs and 1,714 style-inconsistent pairs.

5.1.2 Network Architecture. Our style-consistency prediction network takes two texture images as input and predicts a style-consistency score between the two textures. Since this

Table 1. Comparisons on Adopting Different Network Models for Our Style-Consistency Prediction Network

Model	Accuracy (training set)	Accuracy (testing set)	Running time (seconds)
VGG-11 (ours)	0.996	0.978	0.921
VGG-11 (original)	0.997	0.980	1.188
VGG-13	0.997	0.981	1.311
VGG-16	0.998	0.983	1.512
ResNet	0.953	0.935	0.912
EfficientNet	0.932	0.921	1.102

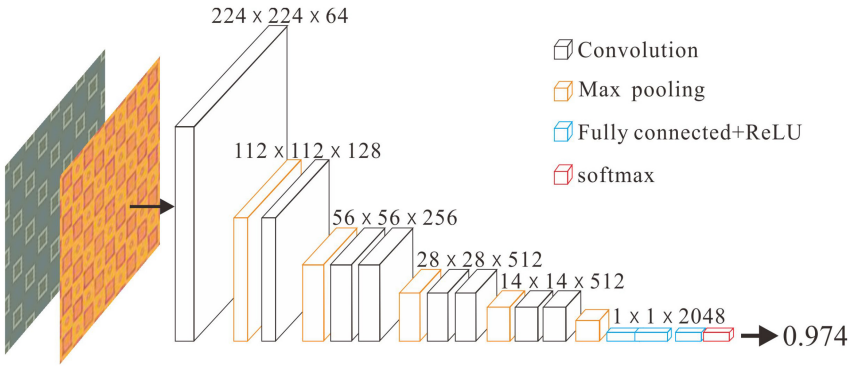

Fig. 10. Network architecture of our style-consistency prediction network.

style-consistency prediction task is relatively simple, we find that a relatively simple network model can already achieve high prediction accuracy. Besides, since we need to predict the style-consistency for a large number of texture pairs, it is desired that the network can run efficiently. With the preceding aims, we choose VGG (including VGG-11, VGG-13, VGG-16) [30], ResNet [10], and EfficientNet [31] as the potential network models and test them on our prepared dataset. The statistics are shown in Table 1.

As shown from the statistics, the VGG models perform much better than the other two models and achieve more than 98% accuracy on the test dataset. Among the three VGG models, the deeper the network model is, the better performance it achieves. However, the improvement of the performance between the deeper network (i.e., VGG-16) and the shallower one (i.e., VGG-11) is extremely minor (<1%), but leading to additional running time (>0.3 seconds per prediction). To balance between performance and computational efficiency, we choose the VGG-11 network structure for our style-consistency prediction task. To further boost computational efficiency, we reduce the parameters of the three fully connected layers to half the original amount. We find that the accuracy is not much affected with such parameter reduction (reduced by 0.1%), but the computational efficiency is evidently improved (reduced by 0.26 seconds per prediction). The detailed network architecture is shown in Figure 10. The network contains 11 hidden layers, including 8 convolutional layers and 3 fully connected layers. The softmax function is used as the activation function.

5.1.3 Loss Function. We adopt the cross-entropy loss function for our network. The cross-entropy loss function is usually used in classification problems. It describes the distance

between two probability distributions—that is, the difference between the ground-truth probability distribution and the predicted probability distribution. The cross-entropy loss function is formulated as

$$L = -\sum_{i=1}^{n} y_i log(p_i). \tag{12}$$

Here, y_i is the ground-truth distribution, p_i is the distribution predicted by the network model, and n is the number of categories. Finally, we use the softmax function to activate, which outputs the probability that the two input textures are style-consistent (i.e., the style-consistency score). The output style-consistency score is a number in the range of $[0, 1]$, where 1 indicates that the two textures are extremely style-consistent and 0 indicates that the two textures are not style-consistent at all.

Our network model is implemented using PyTorch with Python. The network is trained with the learning rate set to 0.0001 and the batch size set to 32. The network converges in about 35 epochs.

5.2 Style-Consistency for a Group of Textures

While the style-consistency between any two textures is predicted by the preceding style-consistency prediction CNN, we measure the style-consistency for a group of more than two textures based on the mutual style-consistency between every two textures. Concretely speaking, we denote the style-consistency for two textures t_i and t_j as $C(t_i, t_j)$. Then, for a group of textures $t_1, t_2, \ldots, t_M$, we measure the style-consistency of this texture group as the averaged mutual style-consistency between every two textures:

$$C(t_1, t_2, \ldots, t_M) = \frac{2\sum_{1\le i,j\le M, i\ne j} C(t_i, t_j)}{M(M-1)}. \tag{13}$$

6 OPTIMIZATION

With the defined suitability and style-consistency measurements, it can formulate the texture recommendation problem as an optimization problem. Given a texture dataset T and M user-specified regions $r_1, r_2, \ldots, r_M$, we intend to find M textures $t_1, t_2, \ldots, t_M$ from the texture dataset T so that the suitability of all textures $S(t_i, r_i), i = 1, 2, \ldots, M$ and the style-consistency $C(t_1, t_2, \ldots, t_M)$ for the picked textures are maximized:

$$\underset{t_1, t_2, \ldots, t_M}{\arg\max} \sum_{i=1}^{M} S(t_i, r_i) + C(t_1, t_2, \ldots, t_M). \tag{14}$$

Due to the large amount of textures in our texture dataset, it is intractable to do a brute-force search. Instead, we propose to solve this optimization problem using the genetic algorithm. To find the optimal solution effectively and efficiently, we first form a set of good initial seeds based on the suitability measurement. Then we generate a number of generations by selection, crossover, and mutation. Finally, we can obtain an optimal solution based on the fitness function defined in Equation (14).

To find a set of good initial seeds, we propose to form the initial texture groups using the textures with high suitability scores. In particular, for each color region r_i, we first rank the textures based on their suitability scores for this region as $t_1^{r_i}, t_2^{r_i}, \ldots$. Then we adopt the first 30 textures for each region and form 30 texture groups where textures with the same rank are in the same texture group—that is, $(t_1^{r_1}, t_1^{r_2}, \ldots, t_1^{r_M})$, $(t_2^{r_1}, t_2^{r_2}, \ldots, t_2^{r_M})$, $\ldots$, $(t_{30}^{r_1}, t_{30}^{r_2}, \ldots, t_{30}^{r_M})$, as the initial solutions for the genetic algorithm. In each generation, three operations are performed: crossover, mutation, and selection. The candidate solutions are first randomly recombined using uniform crossover

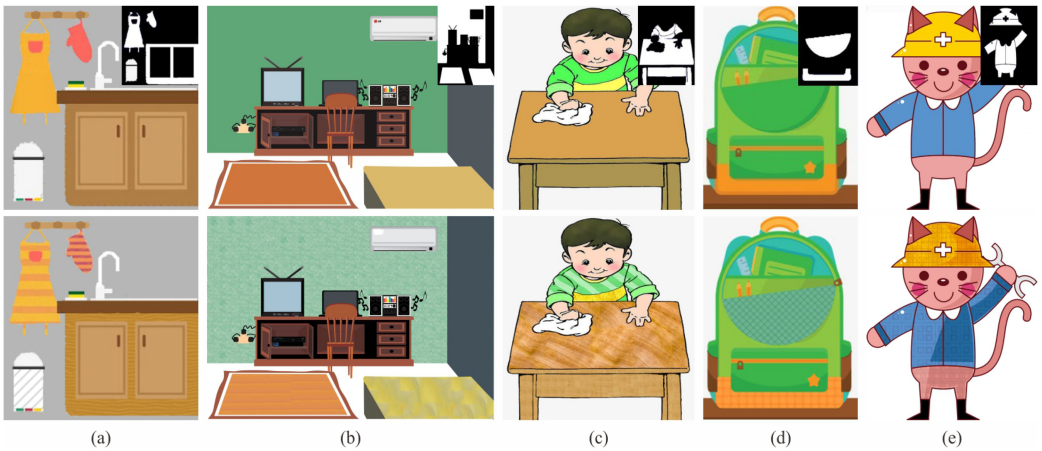

Fig. 11. Multi-texture recommendation results. The first row shows the input cartoon illustrations and the user-specified region mask. The second row shows the textured output. (a) Four user-specified regions. Running time: 7.84 seconds. (b) Three user-specified regions. Running time: 6.04 seconds. (c) Four user-specified regions. Running time: 7.64 seconds. (d) Two user-specified regions. Running time: 5.16 seconds. (e) Three user-specified regions. Running time: 6.16 seconds.

where the crossover probability is set to 0.6. The candidate solutions are then randomly mutated with the mutation probability set to 0.03. During mutation, we randomly pick a color region and change its corresponding texture to another texture with non-uniform probability—that is, textures with higher suitability scores are more likely to be picked. Finally, during selection, we only keep 30 candidate solutions with highest fitness scores based on the fitness function defined in Equation (14). We run the algorithm for 500 iterations. The solution with the highest fitness score in the final population is regarded as the optimal texture group and used to replace the color regions.

Note that as long as the texture dataset is fixed, the synthesizability and cartoonity of the textures can be pre-computed for each texture offline. The style-consistency between every two textures also can be pre-computed. Therefore, we only need to calculate these features once. Only the region fitness feature needs to be calculated online, which greatly improves the efficiency of our optimization method.

7 RESULTS AND DISCUSSION

We apply our methods on various cartoon illustrations with different drawing styles and content (character, object, scenes). Convincing results are obtained. The database used to evaluate our proposed method is obtained from the texture exemplars from the synthesizability dataset [6] and the UIUC dataset [17], where the detailed database preparing process can be found in Section 5.1.1.

7.1 Qualitative Evaluations

Figure 11 shows several results generated using our method. It is observable that textures recommended by our method are not only suitable for each color region but also mutually consistent in style. Here, we would like to emphasize again the importance of finding an appropriate texture based on the original color of the region. The advantage of texture recommendation based on color instead of laying the original color on a grayscale texture is that the original semantics of the regions can be preserved to some extent. For example, in Figure 11(c), the region of the table is

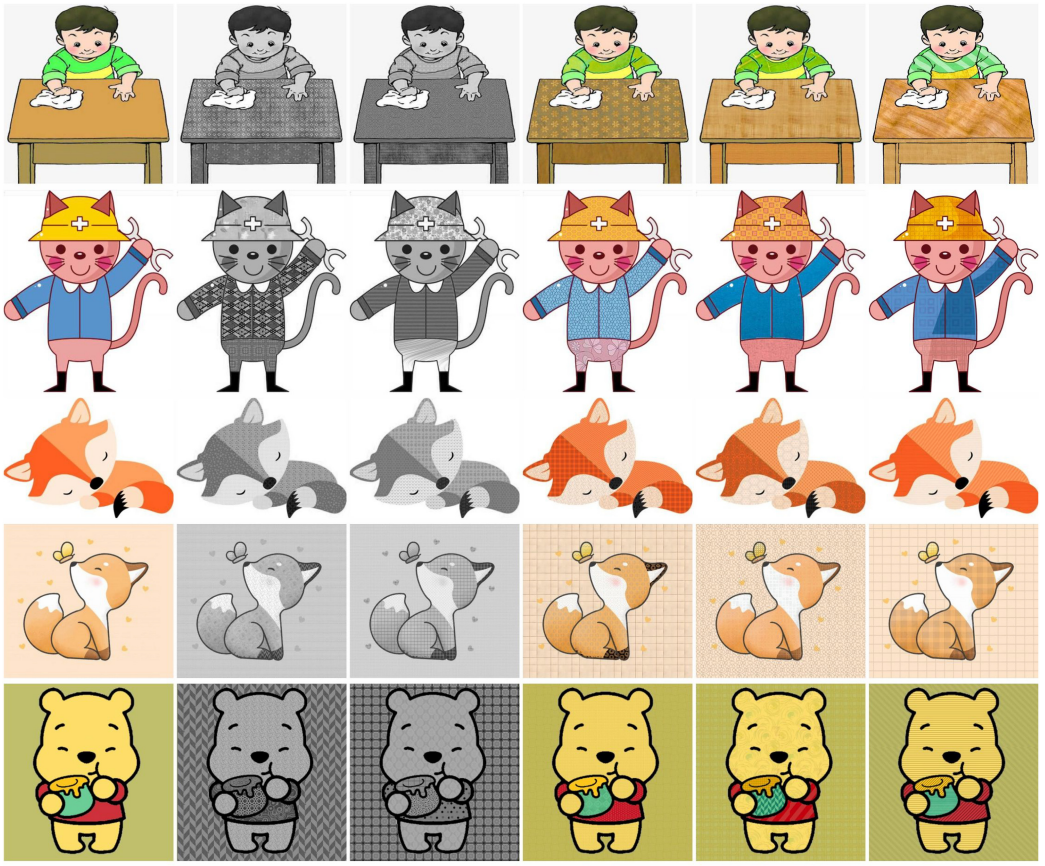

Fig. 12. Comparisons with existing state-of-the-art methods on the cartoon characters.

impressively replaced by a wood texture. Only a color-aware solution can successfully match such semantics. Besides, in all of our examples, no only do we replace each user-specified region with suitable cartoon textures, but the recommended textures are also quite consistent with each other in terms of style. Interestingly, we find that although the style-consistency is quite subjective, our network tends to output a higher style-consistency score for textures that have similar structures. For example, in the case of Figure 11(a), both the apron and the glove are replaced with regular stripe texture patterns. Besides, we need to emphasize again that our results (see Figures 11–13) may become even better and more diverse if the texture dataset becomes larger.

7.1.1 Comparisons with State-of-the-Art Methods. To the best of our knowledge, none of the existing approaches are tailored to replacing the color regions in cartoon illustrations with color textures as in our task. The most similar research works are the photo-to-manga methods that replace the color regions in a color image with black-and-white screentone patterns. Therefore, we compare our method with two state-of-the-art photo-to-manga methods: the methods of Qu et al. [26] and Li and Mould [20]. Examples are shown in Figure 12 and Figure 13. The main difference between the photo-to-manga methods and our method is that they only takes the color information in the original image as a guidance so that regions of different colors are replaced

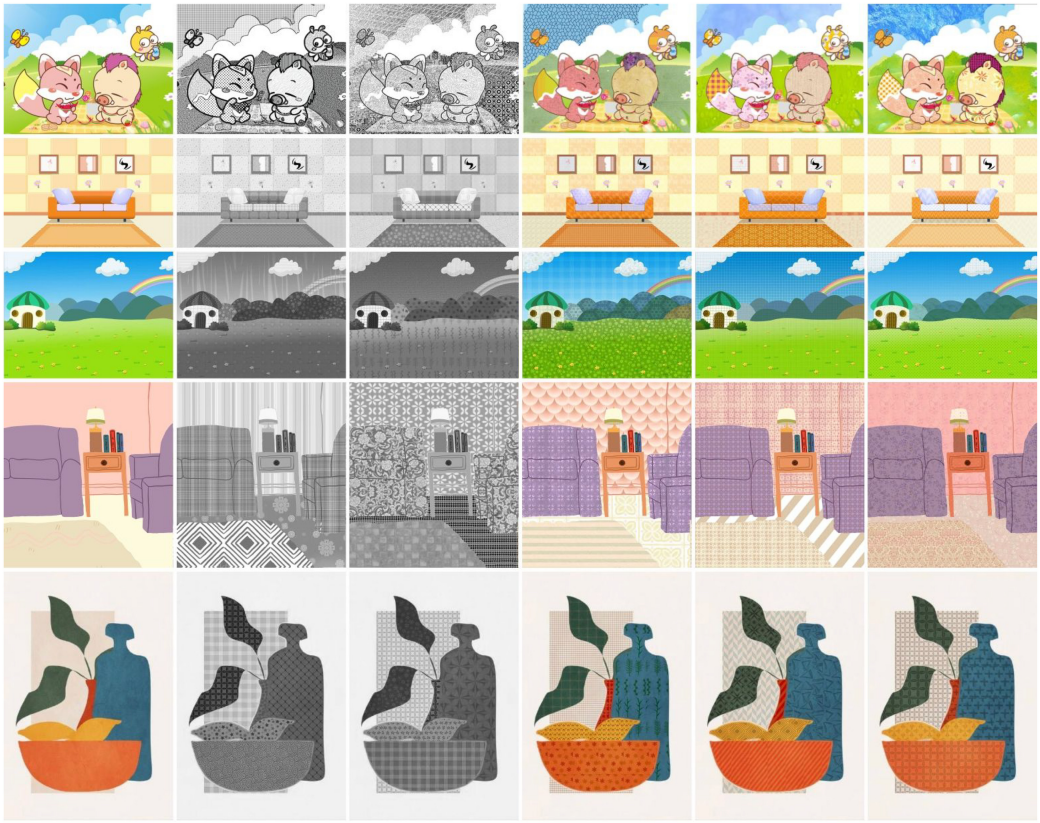

Fig. 13. Comparisons with state-of-the-art methods on the relatively complex scenes.

by different textures. However, the semantic information of the region's color is not taken into consideration, which makes some of the replaced textures unlikely to be used by artists in similar scenarios. For example, in the first case, in Figure 13(b), Qu's method replaces the sky by a dot pattern, although one may lay the original blue color on the pattern to make the texture colorful, but artists seldom use dots for sky in real cases. In comparison, our method replaces the sky with a blue irregular texture which is proper for sky (see Figure 13(f)). We believe that this is because our textures dataset is collected from real textures, so the semantics of the colors are implicitly considered. In Figure 13(c), Li's method generates texture patterns with computers instead of synthesizing from artist-created textures, so their generated textures are not stable and are unable to achieve texture consistency across pixels within the same region—for example, the textures of the grass vary a lot spatially from left to right. In sharp comparison, our method achieves texture consistency with a region-based method (see Figure 13(f)).

In addition, we compare our method with the latest texture exemplar extraction technique (the method of Wu et al. [34]) and the manually created results. In our experiments, the manually created results are obtained by inviting anonymous artists to manually produce and select results for our comparisons. Since our method is the first multi-texture recommendation method for cartoon illustration, we can only design more feasible competitors based on the latest texture exemplar extraction techniques. As shown in Figure 12 and Figure 13, we can also easily observe

Fig. 14. Comparisons with synthesizability metric of Dai et al. [6].

that our method generally outperforms the other three feasible competitors, achieving similar performance with the manually created results. By simultaneously adopting and normalizing textureness, homogeneity, and repetitiveness to improve the discriminability of the synthesizability for the input textures, our multi-texture recommendation method provides a better tool for cartoon illustrations than existing texture exemplar classification and extraction techniques. Even for the typical cases with a more complex scene in Figure 13, our method can still produce more suitable and style-consistent multi-texture recommendations for cartoon illustrations, clearly demonstrating the effectiveness of our proposed novel texture synthesizability metric.

Besides, we also compare our method with the texture synthesizability metric in the work of Dai et al. [6] to demonstrate the advantage of our proposed metric. By replacing the proposed synthesizability metric with the synthesizability metric in their work [6], we can obtain the multi-texture recommendation results in Figure 14(b). From the comparisons in Figure 14 and Figure 15, we can clearly observe that our proposed texture synthesizability metric generally outperforms Dai's metric, which achieves more suitable and style-consistent multi-texture recommendation for cartoon illustrations. By simultaneously adopting and normalizing textureness, homogeneity, and repetitiveness to construct a novel texture synthesizability metric, we can enhance the discriminability of the synthesizability for the input textures to provide a more suitable and style-consistent multi-texture recommendation for cartoon illustrations.

7.1.2 User-Selectable Results. While our system generally outputs one optimal result for the input image and user-selected regions, it is actually possible to output multiple results for users to choose. To do so, we can simply take multiple texture groups with high fitness scores from the last generation of the genetic algorithm and use them for texture replacement, respectively. Figure 16 shows one example. Figure 16(b) and (c) show two results generated from two different texture groups of high fitness scores. It can be seen that both results choose textures that are suitable for the corresponding regions in both color and scale. Besides, even though the recommended textures in

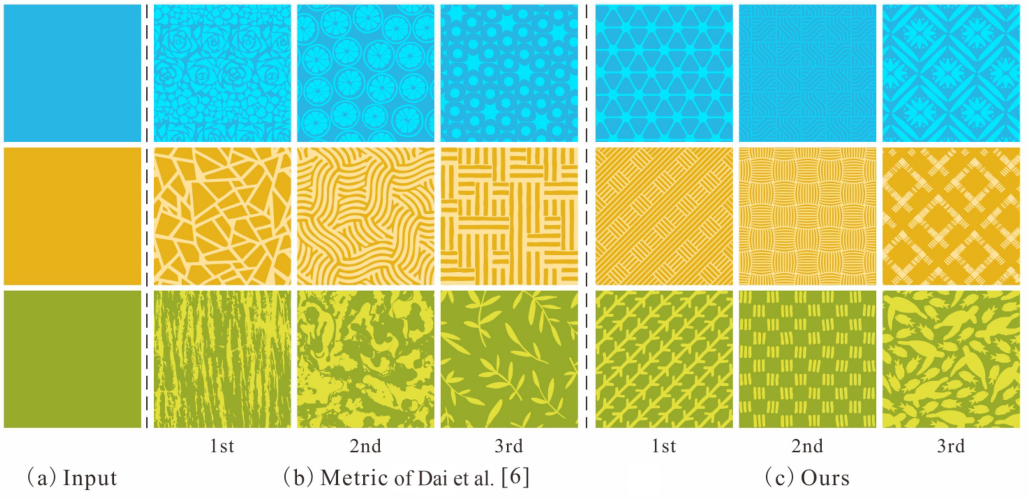

Fig. 15. Comparisons with synthesizability metric of Dai et al. [6] for the same data screening results.

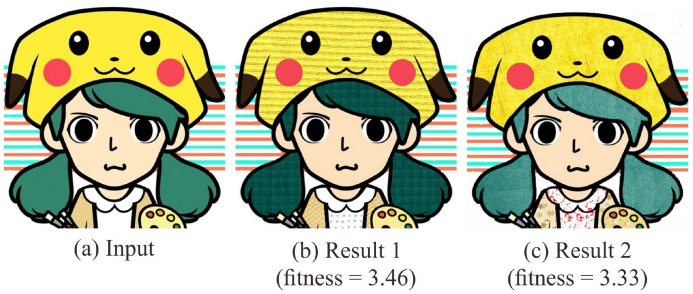

Fig. 16. Different texture recommendation results for the same input image. Fitness is the final score of the texture group. There are five user-specified regions. Our method runs for 9.59 seconds.

the two results are quite different from each other, the textures in either result are style-consistent with each other. The textures in Figure 16(b) are more irregular, whereas the textures in Figure 16(c) are more regular.

7.1.3 Ablation Study. The effectiveness of each loss term in measuring suitability has already been discussed in Section 4. To validate the effectiveness of style-consistency, we compare the results with and without the style-consistency measure. Figure 17 shows two examples. As observed from the results, when style-consistency is not considered, the recommended textures match the original color regions well in terms of color and scale. However, the styles of the textures are not quite consistent with each other (see Figure 17(b)). In comparison, after taking style-consistency into consideration, even though the recommended textures may have slightly lower suitability (e.g., the dominant color of the texture may slightly deviate from the original color), the styles of the textures are quite consistent with each other.

7.2 User Study

Since texture recommendation is quite subjective, adopting quantitative measurements to measure whether the recommended textures are correct is impractical. Instead, we try to validate the

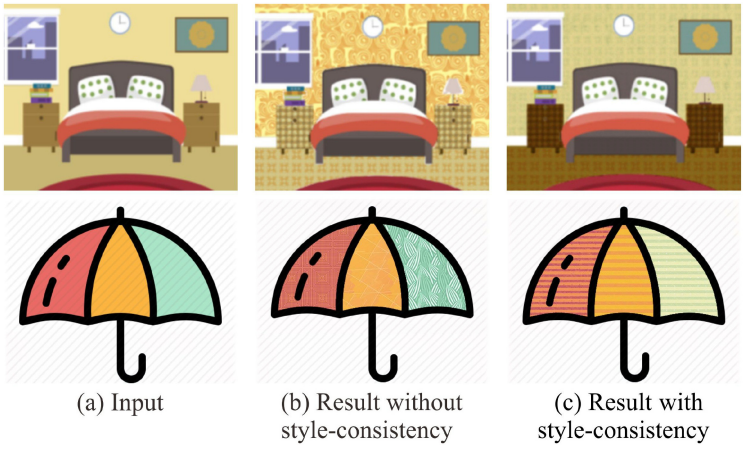

Fig. 17. Ablation study on with and without style-consistency consideration.

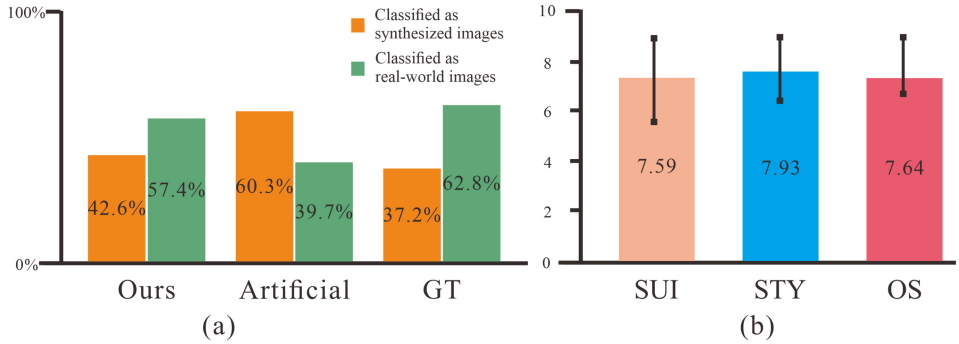

Fig. 18. (a) User study on overall satisfaction. (b) User study on satisfaction for different factors.

effectiveness of our method via auser study. We invited 20 users to our experiment, including 10 males and 10 females, aging from 20 to 40 years. We conducted two experiments, respectively measuring the overall user satisfaction with our results and the user satisfactions in terms of different factors.

7.2.1 Overall Satisfaction. To measure the overall satisfaction of our method, we designed the user study of overall satisfaction by putting the computer-generated results and user-generated results together and asking users to select the real ones. Specifically, we collected 90 images in our user study, including 30 generated by our method, 30 manually created by anonymous artists, and 30 from real-world cartoons or cartoon illustrations with multiple texture regions. In this user study, we showed the images to the users in a random order and asked them to discriminate whether an image was a computer-generated result. The collected user study results are as shown in Figure 18(a), which further demonstrates the effectiveness of our proposed method. Users generally could feel that it was challenging to distinguish between real examples (or manually created results) and the results generated by our method. More than 57% of our generated images were classified as real cartoons, further demonstrating the high quality of our multi-texture recommendations for cartoon illustrations.

Table 2. Comparisons on Adopting Different Methods in Textureness Classification

Model	Accuracy (training set)	Accuracy (testing set)	Training time (hours)	Running time (seconds)
ResNet	0.986	0.973	15.893	1.268
EfficientNet	0.943	0.935	10.527	1.135
TopFormer	0.978	0.969	12.645	1.332
SVM(ours)	0.971	0.965	0.049	0.008

Table 3. Timing Statistics for the Offline Process

Computation of texture properties (24,175 textures in total)		**Network training**	**Total**
Synthesizability	**Cartoonity**		
1.9 s/texture 12.7 h (total)	0.16 s/texture 1.0 h (total)	12.7 min	14.0 h

7.2.2 Satisfaction for Different Factors. After the previous experiment, we showed all of our results to the users again, together with the original images. Each original image and its corresponding result image were shown to the user side by side. The users were then asked to rate how they liked the replaced textures in terms of suitability to each region, style-consistency between the textures, and overall satisfaction. The range of rating was from 1 to 10, where 10 indicated the most favorite and 1 indicated the least favorite. The results are shown in Figure 18(b). In general, the users liked our results and believed that our method achieved both suitability and style-consistency.

7.3 Timing and Score Statistics

Our experiments were performed on a PC with an Intel Core i9 CPU and an NVIDIA RTX 2080 Ti GPU. The whole algorithm can be divided into two processes: an offline process and an online process. In our experiments, we compared the SVM method with several typical deep learning based methods for textureness classification, including ResNet [10], EfficientNet [31], and TopFormer [45]. As mentioned before, deep learning methods may greatly increase the computational cost and cause over-fitting due to the relatively small dataset and the complexity of the learned texture features. As shown in Table 2, we can clearly observe that the SVM method is much more efficient than other deep learning based methods while obtaining similar accuracy performance.

The offline process only needs to be executed once before feeding the input image to the system, which includes the computation of texture properties (synthesizability and cartoonity) for all textures and the training of the style-consistency prediction network. The timing statistics for the offline process is shown in Table 3. In addition, the statistics results for the scores of different metrics are also provided in Table 4, which also clearly indicates the effectiveness of our suitable and style-consistent multi-texture recommendation method. During the computation of texture properties, the most time-consuming part is in calculating the synthesizability of the textures, where calculating the repetitiveness takes more than 85% of the computational time. Fortunately, this process only needs to be calculated once. The major goal is to filter out textures that are too photo-realistic and unlike cartoon textures. If the texture dataset is further enlarged, the computational time of the offline process will also be increased accordingly.

Table 4. Score Statistics of Different Metrics

Image	Suitability			Style-consistency
	Synthesizability	Cartoonity	Region fitness	
Figure 11 "kitchen"	0.9736	0.9329	0.9831	0.9139
Figure 11 "room"	0.9412	0.8615	0.9563	0.8937
Figure 11 "boy"	0.9386	0.8937	0.9309	0.9032
Figure 11 "bag"	0.9527	0.9318	0.9261	0.9837
Figure 11 "cat"	0.9634	0.9509	0.9425	0.9228
Figure 12 "fox 1"	0.9478	0.9213	0.9371	0.9162
Figure 12 "fox 2"	0.9108	0.9427	0.9634	0.9358
Figure 12 "bear"	0.9536	0.9710	0.9461	0.9237
Figure 13 "animal"	0.9421	0.9839	0.9308	0.9433
Figure 13 "room"	0.9683	0.9523	0.9251	0.9507
Figure 13 "turf"	0.9247	0.9329	0.9463	0.9631
Figure 13 "sofa"	0.8939	0.9082	0.9167	0.9384
Figure 13 "vase"	0.9341	0.9153	0.9372	0.9467

Note that the scores of synthesizability, cartoonity, and region fitness are the average results across multiple regions in the image.

The online process needs to be calculated whenever a new input image is fed to the system, which includes the whole system flow except the offline part. The detailed timing statistics are reported in the caption of each figure. Note that, here, the time for suitability only consists of the time of computation of region fitness and the overall suitability score. The time for style-consistency only consists of the testing time. We do not show the time for manual region selection here, which is rather quick. The user only needs to click on the regions he/she wants. After being sped up by the offline process, the time for suitability, style-consistency, and optimization is rather small, where the time for style-consistency is most occupied by importing the trained network to the environment. The segmentation and texture synthesis process take a relatively long time because they directly process the images in the original resolution. We can observe that the computation time is roughly positively related to the number of user-selected regions, especially in the texture synthesis process. The total online time is generally around 10 to 20 seconds based on the number of user-selected regions, which is extremely efficient compared to manual texture crafting. Even if the texture dataset contains more textures, only the offline process will be affected, whereas the online process will have similar computational performance.

7.4 Limitations

Although our experiments can well meet the needs of designers, our method still has several areas that need further improvement. The first and most important is the problem of the dataset. In the experiment, we collected a large number of texture images and created a dataset containing 24,175 texture images. However, after analyzing the synthesizability and cartoonity, only 548 textures remained, which highly affects the quality of our generated images. If only for such a small dataset, the recommendation results can be directly sorted according to a scoring situation, without using a genetic algorithm to find the optimal solution. Due to the limited number of appropriate textures, it is sometimes difficult to find textures with similar colors (e.g., the floor region in the first case of Figure 17). Additionally, small datasets may lead to poor generalization of deep learning models. Our results in Figure 11, Figure 12, and Figure 13 also have demonstrated that our results can become even better and more diverse if the texture dataset becomes larger. In the future, we

will continue to collect larger datasets to improve the performance of the system. Furthermore, if the user selects too many regions, it will be difficult to find a set of textures that are mutually style-consistent with each other. For example, in Figures 12 and 13 where all regions are selected, it can observed that while most regions are replaced with random and irregular textures, the tail of the fox and the hair of the porcupine are replaced with dotted patterns. This problem is actually also caused by the lack of appropriate textures. To avoid integrating too many deep learning networks into the entire framework, we still adopt traditional techniques in texture region segmentation [36], texture synthesis [15], and textureness feature extraction [6], which are more efficient and widely used techniques. Since these traditional techniques can already achieve satisfactory results in the relatively simple and minor steps of our method, we believe that deep learning based methods are not a must, although there are many deep learning based methods for texture region segmentation, texture synthesis, and textureness feature extraction.

8 CONCLUSION

This article presented a novel method that can automatically replace color regions in a cartoon illustration with suitable and style-consistent textures. The suitability of a texture for a color region was measured based on synthesizability of the texture, cartoonity of the texture, and region fitness of the texture for the region. The style-consistency of multiple textures was measured with a learning-based approach. The whole problem was formulated as an optimization problem and solved using the genetic algorithm. Convincing results were obtained. Since the major limitation of the current approach lies in the lack of appropriate cartoon textures, collecting or generating more cartoon textures would be a key research direction. The challenge of directly generating cartoon texture lies in that we want the textures to look real. For example, a wood texture is unlikely to be blue. Therefore, a potential approach is to convert real-world photo-realistic textures to cartoon textures so that our texture dataset can be enriched.

REFERENCES

[1] Vincent Andrearczyk and Paul F. Whelan. 2016. Using filter banks in convolutional neural networks for texture classification. *Pattern Recognition Letters* 84 (2016), 63–69.

[2] Pascal Barla, Simon Breslav, Joëlle Thollot, François X. Sillion, and Lee Markosian. 2006. Stroke pattern analysis and synthesis. *Computer Graphics Forum* 25, 3 (2006), 663–671.

[3] Xingyuan Bu, Yuwei Wu, Zhi Gao, and Yunde Jia. 2019. Deep convolutional network with locality and sparsity constraints for texture classification. *Pattern Recognition* 91 (2019), 34–46.

[4] Bin Chen, Lingyan Ruan, and Miu-Ling Lam. 2020. LFGAN: 4D light field synthesis from a single RGB image. *ACM Transactions on Multimedia Computing, Communications, and Applications* 16, 1 (2020), 1–20.

[5] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, and Andrea Vedaldi. 2016. Deep filter banks for texture recognition, description, and segmentation. *International Journal of Computer Vision* 118, 1 (2016), 65–94.

[6] Dengxin Dai, Hayko Riemenschneider, and Luc Van Gool. 2014. The synthesizability of texture examples. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 3027–3034.

[7] W. Dong, F. Wu, Y. Kong, X. Mei, T. Lee, and X. Zhang. 2016. Image retargeting by texture-aware synthesis. *IEEE Transactions on Visualization and Computer Graphics* 22, 2 (2016), 1088–1101.

[8] Chichen Fu, Di Chen, Edward J. Delp, Zoe Liu, and Fengqing Zhu. 2018. Texture segmentation based video compression using convolutional neural networks. *arXiv:1802.02992* (2018).

[9] Yunfei Fu, Hongchuan Yu, Chih-Kuo Yeh, Tong-Yee Lee, and Jian J. Zhang. 2021. Fast accurate and automatic brushstroke extraction. *ACM Transactions on Multimedia Computing, Communications, and Applications* 17, 2 (2021), 1–24.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[11] Trang-Thi Ho, John Jethro Virtusio, Yung-Yao Chen, Chih-Ming Hsu, and Kai-Lung Hua. 2020. Sketch-guided deep portrait generation. *ACM Transactions on Multimedia Computing, Communications, and Applications* 16, 3 (2020), 1–18.

[12] Ken Ishibashi. 2018. Interactive texture chooser using interactive evolutionary computation and similarity search. In *Proceedings of 2018 Nicograph International (NicoInc '18)*. 37–44.

[13] Rubaiat Habib Kazi, Takeo Igarashi, Shengdong Zhao, and Richard C. Davis. 2012. Vignette: Interactive texture design and manipulation with freeform gestures for pen-and-ink illustration. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1727–1736.

[14] Johannes Kopf and Dani Lischinski. 2012. Digital reconstruction of halftoned color comics. *ACM Transactions on Graphics* 31, 6 (2012), Article 140, 10 pages.

[15] Vivek Kwatra, Arno Schödl, Irfan A. Essa, Greg Turk, and Aaron F. Bobick. 2003. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics* 22, 3 (2003), 277–286.

[16] Hui Lai, Lulu Yin, Huisi Wu, and Zhenkun Wen. 2017. A novel texture exemplars extraction approach based on patches homogeneity and defect detection. In *Advances in Multimedia Information Processing—PCM 2017*. Lecture Notes in Computer Science, Vol. 10736. Springer, 735–744.

[17] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. 2005. A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27, 8 (2005), 1265–1278.

[18] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. 2006. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2169–2178.

[19] Thi-Ngoc-Hanh Le, Ya-Hsuan Chen, and Tong-Yee Lee. 2023. Structure-aware video style transfer with map art. *ACM Transactions on Multimedia Computing, Communications, and Applications* 19, 3 (2023), 1–24.

[20] Hua Li and David Mould. 2011. Content-sensitive screening in black and white. In *Proceedings of the International Conference on Computer Graphics Theory and Applications*. 166–172.

[21] W. Li, H. Gong, and R. Yang. 2019. Fast texture mapping adjustment via local/global optimization. *IEEE Transactions on Visualization and Computer Graphics* 25, 6 (2019), 2296–2303.

[22] Xueting Liu, Chengze Li, and Tien-Tsin Wong. 2017. Boundary-aware texture region segmentation from manga. *Computational Visual Media* 3, 1 (2017), 61–71.

[23] Aude Oliva and Antonio Torralba. 2001. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision* 42, 3 (2001), 145–175.

[24] Wai-Man Pang. 2010. An intuitive texture picker. In *Proceedings of the 15th International Conference on Intelligent User Interfaces*. 365–368.

[25] Yingge Qu, Wai-Man Pang, Tien-Tsin Wong, and Pheng-Ann Heng. 2008. Richness-preserving manga screening. *ACM Transactions on Graphics* 27, 5 (2008), 155.

[26] Yingge Qu, Tien-Tsin Wong, and Pheng-Ann Heng. 2006. Manga colorization. *ACM Transactions on Graphics* 25, 3 (2006), 1214–1220.

[27] Swalpa Kumar Roy, Shiv Ram Dubey, Bhabatosh Chanda, Bidyut B. Chaudhuri, and Dipak Kumar Ghosh. 2018. TexFusionNet: An ensemble of deep CNN feature for texture classification. In *Proceedings of 3rd International Conference on Computer Vision and Image Processing*. Advances in Intelligent Systems and Computing, Vol. 1024. Springer, 271–283.

[28] Omry Sendik and Daniel Cohen-Or. 2017. Deep correlations for texture synthesis. *ACM Transactions on Graphics* 36, 5 (2017), Article 161, 15 pages.

[29] Gil Shamai, Ron Slossberg, and Ron Kimmel. 2019. Synthesizing facial photometries and corresponding geometries using generative adversarial networks. *ACM Transactions on Multimedia Computing, Communications, and Applications* 15, 3s (2019), 1–24.

[30] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[31] Mingxing Tan and Quoc V. Le. 2019. EfficientNet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946* (2019).

[32] Tao Tian, Hanli Wang, Sam Kwong, and C.-C. Jay Kuo. 2021. Perceptual image compression with block-level just noticeable difference prediction. *ACM Transactions on Multimedia Computing, Communications, and Applications* 16, 4 (2021), 1–15.

[33] Koki Tsubota, Daiki Ikami, and Kiyoharu Aizawa. 2019. Synthesis of screentone patterns of manga characters. In *Proceedings of the 2019 IEEE International Symposium on Multimedia*. 212–215.

[34] Huisi Wu, Xiaomeng Lyu, and Zhenkun Wen. 2018. Automatic texture exemplar extraction based on global and local textureness measures. *Computational Visual Media* 4, 2 (2018), 173–184.

[35] Huisi Wu, Zhaoze Wang, Zhuoying Li, Zhenkun Wen, and Jing Qin. 2023. Context prior guided semantic modeling for biomedical image segmentation. *ACM Transactions on Multimedia Computing, Communications, and Applications* 19, 2s (2023), 1–19.

[36] Huisi Wu, Yilin Wu, Shenglong Zhang, Ping Li, and Zhenkun Wen. 2016. Cartoon image segmentation based on improved SLIC superpixels and adaptive region propagation merging. In *Proceedings of the 2016 IEEE International Conference on Signal and Image Processing*. 277–281.

[37] Chenggang Yan, Lixuan Meng, Liang Li, Jiehua Zhang, Zhan Wang, Jian Yin, Jiyong Zhang, Yaoqi Sun, and Bolun Zheng. 2022. Age-invariant face recognition by multi-feature fusion and decomposition with self-attention. *ACM Transactions on Multimedia Computing, Communications, and Applications* 18, 1s (2022), 1–18.

[38] Han Yan, Haijun Zhang, Jianyang Shi, Jianghong Ma, and Xiaofei Xu. 2023. Toward intelligent fashion design: A texture and shape disentangled generative adversarial network. *ACM Transactions on Multimedia Computing, Communications, and Applications* 19, 3 (2023), 1–23.

[39] Feng Yang, Gui-Song Xia, Dengxin Dai, and Liangpei Zhang. 2019. Learning the synthesizability of dynamic texture samples. *IEEE Transactions on Image Processing* 28, 5 (2019), 2502–2517.

[40] Chih-Yuan Yao, Shih-Hsuan Hung, Guo-Wei Li, I.-Yu Chen, Reza Adhitya, and Yu-Chi Lai. 2017. Manga vectorization and manipulation with procedural simple screentone. *IEEE Transactions on Visualization and Computer Graphics* 23, 2 (2017), 1070–1084.

[41] Yongqiang Yao, Di Huang, Xudong Yang, Yunhong Wang, and Liming Chen. 2018. Texture and geometry scattering representation-based facial expression recognition in 2D+3D videos. *ACM Transactions on Multimedia Computing, Communications, and Applications* 14, 1s (2018), 1–23.

[42] Lulu Yin, Hui Lai, Huisi Wu, and Zhenkun Wen. 2017. Repetitiveness metric of exemplar for texture synthesis. In *Advances in Multimedia Information Processing—PCM 2017*. Lecture Notes in Computer Science, Vol. 10736. Springer, 745–755.

[43] Ning Yu, Connelly Barnes, Eli Shechtman, Sohrab Amirghodsi, and Michal Lukác. 2019. Texture Mixer: A network for controllable synthesis and interpolation of texture. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 12164–12173.

[44] Hang Zhang, Jia Xue, and Kristin J. Dana. 2017. Deep TEN: Texture encoding network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2896–2905.

[45] Wenqiang Zhang, Zilong Huang, Guozhong Luo, Tao Chen, Xinggang Wang, Wenyu Liu, Gang Yu, and Chunhua Shen. 2022. TopFormer: Token pyramid transformer for mobile semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '22)*. 12083–12093.

[46] Yushu Zhang, Nuo Chen, Shuren Qi, Mingfu Xue, and Zhongyun Hua. 2023. Detection of recolored image by texture features in chrominance components. *ACM Transactions on Multimedia Computing, Communications, and Applications* 19, 3 (2023), 1–23.

[47] Mingliang Zhou, Hongyue Leng, Bin Fang, Tao Xiang, Xuekai Wei, and Weijia Jia. 2023. Low-light image enhancement via a frequency-based model with structure and texture decomposition. *ACM Transactions on Multimedia Computing, Communications, and Applications* 19, 6 (2023), 1–23.

[48] L. Zhu, X. Hu, C. Fu, J. Qin, and P. Heng. 2020. Saliency-aware texture smoothing. *IEEE Transactions on Visualization and Computer Graphics* 26, 7 (2020), 2471–2484.

[49] Daniel Sýkora, Mirela Ben-Chen, Martin Cadık, Brian Whited, and Maryann Simmons. 2011. TexToons: practical texture mapping for hand-drawn cartoon animations. In *Proceedings of the 9th International Symposium on Non-Photorealistic Animation and Rendering*. 75–84.

Received 14 June 2023; revised 5 January 2024; accepted 9 March 2024