Deep Texture Exemplar Extraction Based on Trimmed T-CNN

Huisi Wu, Wei Yan, Ping Li, *Member, IEEE,* and Zhenkun Wen

***Abstract*—Texture exemplar has been widely used in synthesizing 3D movie scenes and appearances of virtual objects. Unfortunately, conventional texture synthesis methods usually only emphasized on generating optimal target textures with arbitrary sizes or diverse effects, and put little attention to automatic texture exemplar extraction. Obtaining texture exemplars is still a labor intensive task, which usually requires carefully cropping and post-processing. In this paper, we present an automatic texture exemplar extraction based on Trimmed Texture Convolutional Neural Network (Trimmed T-CNN). Specifically, our Trimmed T-CNN is filter banks for texture exemplar classification and recognition. Our Trimmed T-CNN is learned with a standard ideal exemplar dataset containing thousands of desired texture exemplars, which were collected and cropped by our invited artists. To efficiently identify the exemplar candidates from an input image, we employ a selective search algorithm to extract the potential texture exemplar patches. We then put all candidates into our Trimmed T-CNN for learning ideal texture exemplars based on our filter banks. Finally, optimal texture exemplars are identified with a scoring and ranking scheme. Our method is evaluated with various kinds of textures and user studies. Comparisons with different feature-based methods and different deep CNN architectures (AlexNet, VGG-M, Deep-TEN and FV-CNN) are also conducted to demonstrate its effectiveness.**

***Index Terms*—deep learning, texture convolutional neural network, trimmed convolutional neural network, texture exemplar recognition, texture exemplar extraction**

I. Introduction

Under a virtual reality booming era, texture synthesis technique is still widely used in generating virtual background scenes or modeling appearances for virtual objects. As an efficient tool, example-based texture synthesis [1] can generate seamless textures with different illumination or deformation distribution effects. In the last two decades, a series of example-based texture synthesis methods were developed to improve the synthesizing effect and efficiency, so we can easily synthesize desired textures based on the input texture exemplar. However, extracting high quality texture exemplars from natural images is still a labor intensive task and heavily relies on the manually processing of the artists. Obviously, manual texture exemplar extraction is tedious, which usually requires carefully photography selection, patiently cropping and post-processing. Currently, conventional texture synthesis methods usually put more emphasis on generating optimal target textures with arbitrary sizes or diverse effects based on the given exemplar input, where is still a lack of attention on automatic texture exemplar extraction. As the quality of the input texture exemplar may mostly determine the effect of final synthesis results, there has been great interest in developing automatic systems that can assist artists in the tedious texture exemplar extraction.

Recently, deep learning obtained a great success in the areas of recognition [2], [3], detection [4], [5], prediction [6], retrieval [7], [8], super-resolution [9], video analysis [10], [11] and so on. In particular, Convolutional Neural Network (CNN) is widely used as filter banks to extract features and also frequently applied for object recognition. Similarly, we found that filter bank can also be a powerful tool to extract useful texture exemplar features, which also can be efficiently applied in texture exemplar recognition and classification. Compared with the texture descriptors based on traditional feature detection, texture analysis and classification, deep texture exemplar extraction framework can achieve better accuracy and efficiency performance based on sophisticated neural network parameter settings. Furthermore, deep learning based method can also achieve automatic texture exemplar extraction without any user interactions.

In this paper, we present the first deep learning architecture for automatic texture exemplar extraction, namely Trimmed T-CNN. Specifically, we can see our Trimmed T-CNN architecture shown in Fig. 1. It has three convolutional layers, and two pooling layers to be our filter banks for texture exemplar classification and recognition. Another contribution of our work is that we have designed the first Trimmed T-CNN to evaluate the quality of the given texture exemplar. To train our Trimmed T-CNN, we have setup a standard ideal exemplar dataset for deep texture exemplar extraction, which contains thousands of desired texture exemplars collected and cropped by our invited artists. Given a natural image, we first perform a selective search algorithm to extract a number of texture exemplar candidates. All candidates are then put into our Trimmed T-CNN for learning ideal texture exemplars. Finally, best exemplars are obtained based on the scoring and ranking scheme. Our method is evaluated with a variety of kinds of textures and compared with different feature-based texture exemplar methods, including different deep CNN architectures (AlexNet, VGG-M, Deep-TEN and FV-CNN). In addition, user studies are also conducted to demonstrate the effectiveness of our method.

Manuscript received Mar 02, 2020; revised September 11, 2020; accepted December 01, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61973221, the Natural Science Foundation of Guangdong Province of China under Grant 2018A030313381 and Grant 2019A1515011165, and The Hong Kong Polytechnic University under Grant P0030419 and Grant P0030929.

H. Wu, W. Yan, and Z. Wen are with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China (E-mail: wenzk@szu.edu.cn).

P. Li is with the Department of Computing, The Hong Kong Polytechnic University, Kowloon, Hong Kong (Email: p.li@polyu.edu.hk).

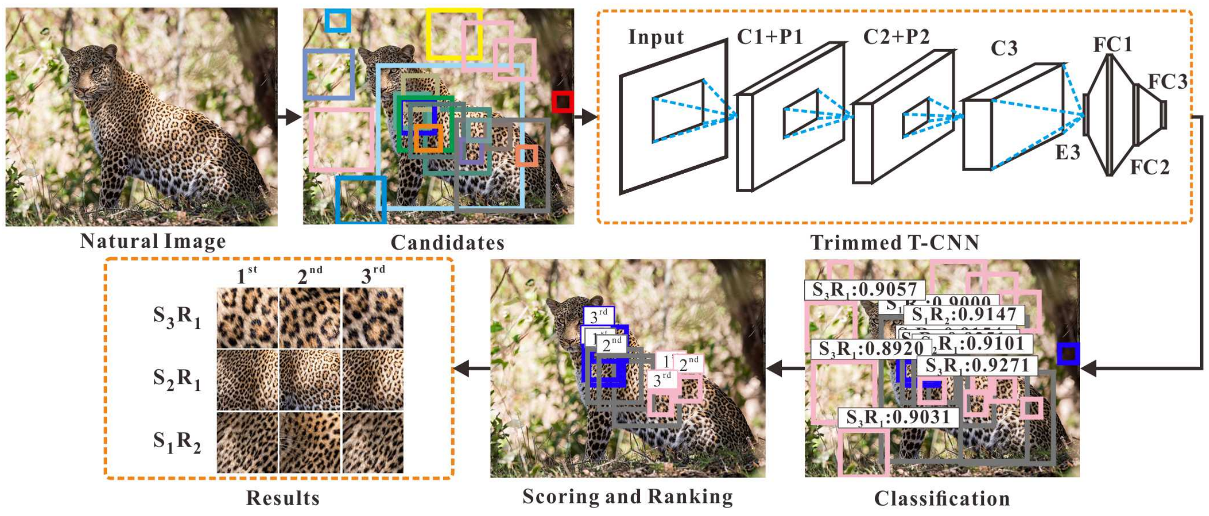


Fig. 1. Framework of our system. Given a natural image, we first perform a selective search algorithm to extract a number of texture exemplar candidates. All candidates are then put into our Trimmed T-CNN model for learning ideal texture exemplars. Best exemplars are obtained based on the final scoring and ranking scheme.

To summarize, the main contributions of this paper are as follows:

- We propose the first deep learning framework for automatic texture exemplar extraction, which is based on a trimmed texture convolutional neural network (Trimmed T-CNN).
- We propose a novel texture exemplar dataset for learning ideal texture exemplars with different scales and regularities, which contains thousands of well-organized ideal texture exemplars with various texture objects, scales and regularities under different conditions.
- Based on the novel strategy of six divisions for the standard exemplar dataset according to scale and regularity, we can not only achieve multi-scale and multi-regularity texture exemplar extraction from a single internet image, but also provide a more flexible selection of texture exemplars for the users.

II. Related Work

A. Exemplar-based texture synthesis

Unlike procedural synthesis methods [12] which create textures with a set of mathematics functions, exemplar-based texture synthesis is the process of generating new texture images with arbitrary size from an input texture exemplar. According to Raad et al. [1], exemplar-based texture synthesis methods can be divided into three types: statistics-based methods [13], [14], patch re-arrangement [15]–[17] methods and hybrid methods [18]. Statistics-based methods produce different texture images with a random sampling strategy based on the exemplar texture characterized by a statistical signature. Patch re-arrangement methods stitch together copies of subregions in the exemplar to generate a new texture based on a clever copy-paste procedure. Hybrid methods combine ideas of the previous two approaches to create new textures based on a given exemplar. On the other hand, convolutional neural networks [19], [20] and generative adversarial networks [14], [21] also provide a new tool and open a new space for exemplar-based texture synthesis. However, researchers are more focused on texture applications, such as the texture image style-transfer [22], [23]. Most of existing texture synthesis methods are still more emphasized on generating optimal target textures and its applications, but pay little attention to automatic texture exemplar extraction from a natural image.

B. Texture exemplar extraction

To efficiently extract the texture exemplars from nature images, several methods were proposed to extract dominant texture descriptors based on traditional feature detection, texture analysis and classification. Given a large globally varying texture, Wei et al. [24] first proposed an inverse texture synthesis technique by summarizing the input texture into a small texture exemplar, which requires the input texture should be well synthesized. Wang et al. [25] also proposed a method to extract dominant textures with multi-scale hue-saturation-intensity histograms. Dai et al. [26] provided a tool to measure the synthesizability of texture exemplars. On the other hand, Lockerman et al. [27]–[29] proposed to create texture exemplar by diffusion manifolds [30]. Moritz et al. [31] also suggested to employ local histogram matching to extract texels from the input photographs. More recently, Wu et al. [32] proposed an automatic texture exemplar extraction method based on global and local textureness measures. However, most of above approaches are based on traditional feature detections for texture recognitions, which also usually require a specific image feature selection for different types of texture exemplar extractions.

C. CNNs for texture recognition

Since AlexNet [33] achieved great success in the ImageNet LSVRC-2012 contest, many researches have paid attention to using CNN for texture descriptor extraction and texture classification. Cimpoi et al. developed a FC-CNN [34] for texture object recognition based on a set of non-linear filter banks. They also improved it with a better texture description ability by developing a FV-CNN [35] for the challenge tasks of texture classification and scene recognition. Similarly, Lin et al. [36] proposed another bilinear model to achieve fine-grained texture recognition. More recently, deep learning techniques also achieved a remarkable breakthrough in the field of texture object detection. According to Liu et al. [37], two-stage object detection framework [38], [39] may potentially be applied to texture extraction. Andrearczyk and Whelan [40] also developed an effective network architecture named Texture CNN (T-CNN) for texture recognition. Unfortunately, above CNN architectures still cannot directly used for automatic texture exemplar extraction. To achieve more efficient texture recognition, we develop a Trimmed T-CNN for automatic texture exemplar extraction.

III. Method

The framework of our proposed method is as shown in Fig. 1. Given an input image, we first use a selective search algorithm to extract a number of image patches, which may have different resolutions and usually contain one or several kinds of texture. To identify the features of the ideal texture patches, we have collected and defined a standard exemplar dataset which contains more than thousands of desired texture exemplars cropped by the invited artists. So we can train our trimmed texture CNN (T-CNN) models with the standard exemplar dataset. The extracted image patches are then classified based on our T-CNN models, where each image patch will be classified as one of the six kinds of desired exemplars with a possibility. Finally, the optimal texture exemplars are identified with a scoring and ranking scheme in the refinement step based on the patch size and classification accuracy.

A. Texture Exemplar Candidates Selection

Given an input image, one may use an exhaustive randomly search strategy to crop a number of texture exemplar candidates. Obviously, exhaustive randomly search is time consuming because it usually includes many redundant searching and cropping. More importantly, it cannot guarantee the randomly search algorithm covering the whole image. As it also cannot automatically determine the patch size, so exhaustive randomly search strategy usually requires a pre-defined patch size as input.

In this paper, we employ a selective search algorithm [41] to extract the texture exemplar candidates more efficiently. We first obtain a number initial regions using graph-based superpixel image segmentation. Then, we iteratively calculate the color, texture, size and fill similarity between two neighboring regions and merge them until the whole region containing a single meaningful object. $S(r_i, r_j)$ is the similarity between two regions. Specifically, we use fill similarity to measure how well region r_i and r_j fitting into each other. If r_i is contained in r_j, we should merge them in order to avoid any holes. Otherwise, if r_i and r_j are incompatible with each other, we should not merge them as they will likely form a strange region. The overall similarity $S(r_i, r_j)$ for region merging can be written as following,

$$\begin{aligned} S(r_i, r_j) =& a_1 S_{color}(r_i, r_j) + a_2 S_{texture}(r_i, r_j) \\ &+ a_3 S_{size}(r_i, r_j) + a_4 S_{fill}(r_i, r_j) \end{aligned} \tag{1}$$

where a_1, a_2, a_3 and a_4 are the proportion of the four similarities. In our experiments, we set a_1, a_2, a_3, a_4 as 0.3, 0.4, 0.1, 0.2, respectively. Note that, we pay more emphasis to the similarity of color and texture. Detailed definitions of the four similarities can be referred to the selective search algorithm [41]. After region merging in the selective search algorithm, we can automatically extract a number of texture exemplar candidates with different scales, as shown in Fig. 2. Because it can automatically determine the patch size based on the criteria of $S(r_i, r_j)$ during the region merging, a pre-defined patch size is not required any more in our selection of texture exemplar candidates.

If the selected patch is a rectangle, we still need to resize it into a square before input to network because the input of our Trimmed T-CNN must be a square. Obviously, it may produce severe distortion or deformation if we linearly resize it into a square, especially for the thin and long rectangle. As we usually only obtain rectangles as the output for the original selective search algorithm, we can optimize it with a square cropping according the smaller edge of the extracted rectangles to guarantee that the output are all square image patches. On the other hand, we also filter out the redundant exemplar candidates with non-texture regions or outliers using a histogram matching algorithm. Specifically, we divide the candidate into four equal sub-regions, and compare the chi-square distance between the statistical histogram result of each sub-region and the candidate. As the histogram matching between sub-region and candidate can reflect statistical invariance, we can further filter out redundant exemplar candidates with image noises based on the average chi-square distance between sub-region and the candidate. In our experiments, we set the threshold of average chi-square distance as 0.6.

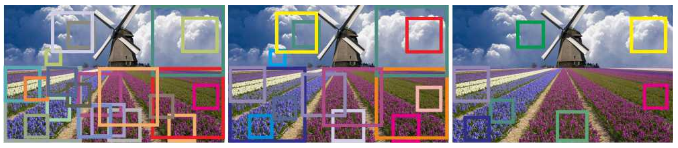


Fig. 2. Texture exemplar candidates extracted by selective search algorithm with different filtering parameters.

B. Standard Exemplar Dataset

To learn the features of the ideal texture exemplar, it is important to collect a texture dataset that includes as much as of distinguish and representative features of the desired texture exemplars. Currently, there exist several open texture datasets collected for texture classification. But most of them are used as texture recognition benchmarks datasets, such as DTD dataset [42], UMD dataset [43], and UIUC dataset [44]. On the other hand, there also have several material datasets for material recognition benchmarks, including Kylberg dataset [45], KTH-TIPS dataset [46], KTH-TIPS2 dataset [47], and FMD dataset [48]. However, most of above datasets are designed for texture classification or recognition, where different texture examples are grouped according to different texture material, such as grassland, sky, building and etc. As the number of different kinds of material in the world is countless, we cannot collect and label all of them to setup a standard dataset for

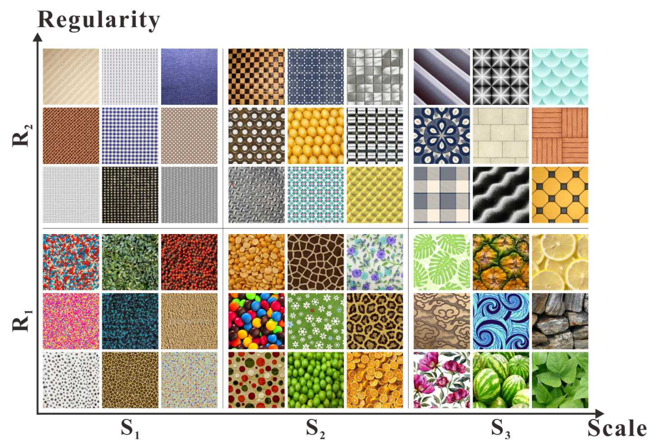


Fig. 3. Six divisions of our standard exemplar dataset. S_1, S_2, and S_3 represent the texture exemplar with small, median or large scale of objects, respectively. R_1 and R_2 denote the texture exemplar with random or regular distribution. All exemplars are with a resolution of 256×256.

learning different ideal texture exemplars. Moreover, ideal texture exemplars also should not be limited in a number kinds of material. To learn the intrinsic features of the texture exemplar, we do not divide the collected texture examples according to the material. Instead, we label the standard ideal texture exemplar in our standard dataset according to the texton scale and regularity in the texture patch.

As there exist too many kinds of material to be exhaustively collected for learning desired texture exemplars, we can collect a large number of ideal texture exemplars and divide them into several classes, each of which contains a desired texton scale and regularity for the ideal exemplar. In our experiments, we have invited the artists to manually crop thousands of ideal texture exemplars from the Internet, and treated them as the standard texture exemplar dataset in our following classification and recognition. To learn the intrinsic feature of the texture exemplar, we did not divide the collected dataset according to the material. Instead, we divide the standard dataset into six classes based on the texton scale and regularity in the texture patch. We use S_1, S_2, and S_3 to represent the texture exemplar with small, median or large scale of objects, respectively. To minimize inconsistency and human subjective feeling factor in the scale labelling, artists denote the texture scale according to the following steps. First, they crop the desired texture exemplars from the given nature images. Second, they are then asked to figure out the minimal repeated texton (epitome) in the desired texture exemplar by drawing a small square box. Finally, by calculating a ratio φ between the minimal texton area and the whole exemplar area, we can obtain a consistent scale labelling for the ideal texture exemplars based on the value of φ. In our experiments, if $0 <= \varphi < 0.08$, the scale is S_1. If the $0.08 <= \varphi < 0.2$, the scale is S_2. Otherwise, the scale is S_3. On the other hand , we use R_1 and R_2 to denote the texture exemplar with random or regular distribution. Thus, we have six classes of ideal texture exemplars with different scales and regularities, including S_1R_1, S_1R_2, S_2R_1, S_2R_2, S_3R_1, and S_3R_2. Typical exemplars are as shown in the Fig. 3. All exemplars are with a resolution of 256×256.

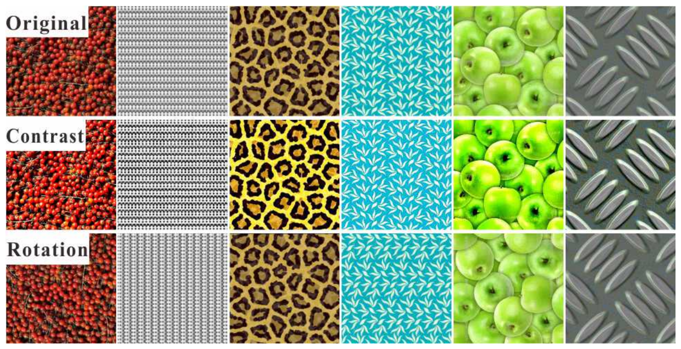


Fig. 4. Data augmentation by generating different illuminations and rotations on the exemplars.

In our standard exemplar dataset, we first selected 1260 best texture exemplars from the artist cropping, where each class contain 210 best texture exemplars. As there may exist different illuminations and rotations for the texture patches in the nature images, so we randomly generated 2 kinds of different illumination contrasts and 4 different rotations to generate more exemplars for T-CNN learning in the following steps. Finally, our standard texture exemplar dataset includes 10080 (1260×8) best texture exemplars and each class contain 1680 exemplars with different illuminations and rotations, as shown in the Fig. 4.

C. Trimmed T-CNN

Recently, CNN layers can be considered as filter banks to extract features and widely applied for object and material recognition [49], [50]. We found that filter banks can also efficiently extract useful texture features, which can be used in texture recognition and classification. Based on the network architecture named Texture CNN (T-CNN) [40], we develop a Trimmed T-CNN for ideal texture exemplar extraction in our system. Compared with the original T-CNN, our Trimmed T-CNN has a better efficiency performance based on less neural network parameters. As we only trim the redundant components in the original T-CNN, we can still achieve a similar accuracy performance. Note that, our Trimmed T-CNN is the first deep learning model in automatic texture extraction, as previous methods only emphasized on other texture synthesis applications and T-CNN is also only used for texture classification.

By sharing the weight of the convolution layers, T-CNN can learn appropriate texture exemplar features (such as edges, shapes, and et al.) through forward and backward propagation on the given standard exemplar dataset. So we can combine these learned texture feature descriptors within the network to classify new unknown image patches extracted from an input image in Fig. 2. Specifically, T-CNN architecture is derived from AlexNet [33]. For each feature map of the last convolution layer, T-CNN pools them by calculating the average of its rectified linear activation output and obtains a single value per feature map.

As shown in the Fig. 5, we employed T-CNN to be our filter banks in our texture exemplar classification and recognition. Firstly, all image patches were resized to 227×227 and put into the first convolutional layer (C1). As mentioned before, because our scale labelling is based on the ratio between the minimal texton area and the whole exemplar area, the linear resizing operation applied on the input exemplars does not change the scale labeling. After forward propagation of the former two convolutional layers, including activation layer and pooling layer (C1+P1, C2+P2), we can obtain texture descriptors for the fine detail (such as edges, key points and et al.). For the last convolutional layer (C3), we can capture larger scales of texture features (such as contours, shapes and et al.) which are more representative and distinguish for texture exemplar classification and recognition. So we can simply pool them by calculating the average value of its rectified linear activation output to obtain a single value per feature map, which is then cascaded by an energy layer for the following fully-connected layers. Similar to the fully connected layer in FCN for semantic segmentation, our energy layer is the global average pooling on the entire feature map, which is a convolution operation with exactly the same kernel size as the feature map. Finally, we gradually project the feature map of the fully-connected layers into six dimensional vector to identify the six classes ideal texture exemplars. Through forward propagation of all filter banks, we can also obtain our texture exemplar descriptors indicated with a number of parameters within the Trimmed T-CNN.

On the other hand, we also applied multi-layered Deconvolutional Network [51], [52] to visualize the feature of each CNN layer, as shown in the colorized image patches in Fig. 5. From visualization of the learned texture feature maps, we found many similar reconstructed images in each layer, indicating the redundancy in the extracted feature map for the convolutional layers. The major reason for the redundancy is that texture images usually appear with repeated patterns (textons), which is also named as the statistical invariance [53]. Especially in the deep convolutional layers, we can clearly observe that the extracted feature maps are very similar with each other. To achieve a more efficient and light-weight network, we trim the original T-CNN in both convolutional layers and fully connected layers, which also minimizes the interference of redundant feature maps. The original T-CNN has 5 convolutional layers (C1 to C5). Due the repeated statistical invariance in the texture patches, the feature maps in the last two convolutional layers (C4 and C5) are very hard to distinguish with each other. Therefore, we have trimmed C4 and C5 in our Trimmed T-CNN. Without the interference of redundant feature maps in C4 and C5, our Trimmed T-CNN extracts more distinguishable feature maps only relied on 3 convolutional layers, which also achieves better efficiency without scarifying accuracy performance of our network. Moreover, we also try to further trim more convolutional layers in our Trimmed T-CNN. However, the accuracy of our Trimmed T-CNN cannot be preserved if we employ less than 3 layers in our network, indicating that we will damage the feature learning ability of the network when trimming too many convolutional layers.

Although our experiments have verified that 3 layers are the minimum convolutional layers in our Trimmed T-CNN to guarantee the accuracy performance. We can still observe partially redundancy in the feature maps of C3 layer, as shown in the Fig. 5. Therefore, we also employ Han's method [54] to trim redundant connections in our fully connected layers. Specifically, our fully connected layer trimming is implemented by removing redundant hidden nodes from the second hidden layer (as shown in Fig. 5), where redundant hidden nodes are corresponding to the redundant connections for the partially redundancy in the feature maps of C3 layer. In the original T-CNN, hidden nodes in the two hidden layers are 4096×4096. Based on Han's method, we can train the network to learn which connections are important, so we can trim the redundant and unimportant connections by removing the hidden nodes in the second hidden layer. After pruning 3584 redundant connections based on the Han's method, our Trimmed T-CNN finally still remains 512 hidden nodes in the second hidden layer. Therefore, hidden nodes of the two hidden layers are 4096×512 in our Trimmed T-CNN. By gradually reducing the number of feature map in the fully connected layer, we can reduce the number of feature map in the second fully connected layer from 4096 to 512, and finally reduce it from 512 to 6 in the last fully connected layer. Our experiment results show that our Trimmed T-CNN achieves similar accuracy performance and has a better efficiency performance.

D. Scoring and Ranking

As the extracted image patches may contain similar region and with different scales, we need to design a texture exemplar scoring and ranking algorithm to obtain best texture exemplars and filter out the bad exemplars, as shown in Fig. 6. Note that, our scoring and ranking algorithms are not involved in the training process.

Given a natural image $I \in \mathbb{R}^{H \times W}$, where H, W are height and width of image. We can obtain plenty of rectangles R_i by selective search algorithm [41], where x, y, w, h are the top-left horizontal coordinate, top-left vertical coordinate, width and height of rectangle R_i respectively. We define its ideal texture exemplar score as following,

$$Score(R_i) = b_1 P_i + b_2 D_i + b_3 S_i \tag{2}$$

which takes the probability P_i, rectangle distribution D_i and rectangle size S_i into consideration. b_1, b_2, b_3 are the proportion of three features, where $b_1 + b_2 + b_3 = 1$. In our experiments, we set b_1, b_2, b_3 as 0.6, 0.2, 0.2 respectively. Each of feature is normalized from 0 to 1. More specifically, the probability factor P_i is written as,

$$P_i = max\{P_{i,k}\}, k = 1, 2, ..., 6 \tag{3}$$

which indicates the best probability for the k-th class in our Trimmed T-CNN output layer ($0 \leq P_i \leq 1.0$). Secondly, our rectangle distribution factor D_i is written as,

$$D_i = D_i(x, y) = 1 - \frac{d\{C(x, y), C_i(x, y)\}}{d_{max}} \tag{4}$$

where $C(x, y)$ is the mean central point coordinate of each rectangle in the same category. d_{max} is the max Euclidean

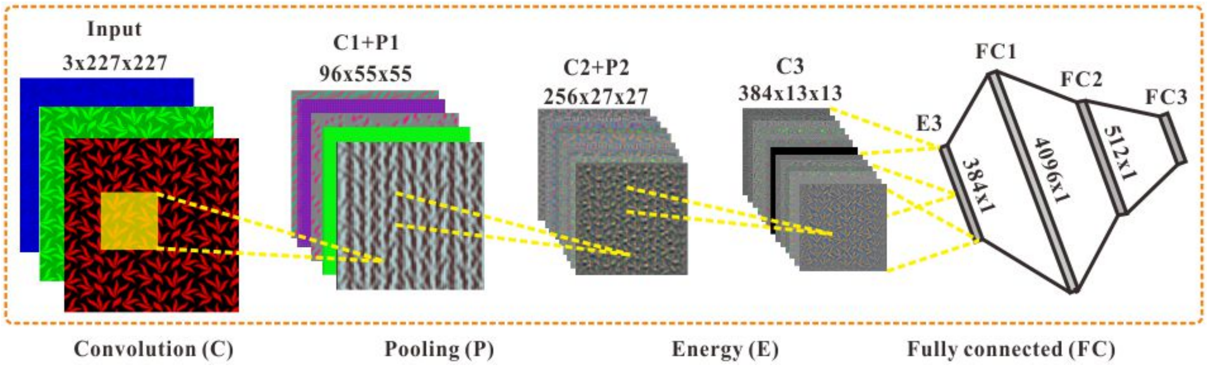


Fig. 5. Our Trimmed T-CNN model. To achieve a more efficient and light-weight network, we trim the original T-CNN in both convolutional layers and fully connected layers. Compared with the original T-CNN, our convolutional layers is reduced from 5 to 3, where the number of hidden nodes in second fully-connected layer is also pruned from 4096 to 512.

distance between $C(x, y)$ and the central point of each rectangle in the same category. $d\{C(x, y), C_i(x, y)\}$ is the Euclidean distance of $C(x, y)$ and central point of R_i. Obviously, $0 \leq D_i \leq 1.0$. When the distance between rectangle R_i and the central point of each category is larger, the value of D_i will get smaller. Finally, we define the relative rectangle size factor S_i as following,

$$S_i = \frac{w_i \times h_i}{Size(I)} \tag{5}$$

where w_i and h_i are the width and height of R_i. $Size(I)$ denotes the area of the image I. Obviously, $Size(I) = W \times H$, and $0 \leq S_i \leq 1.0$. As texture exemplar with larger size may contain more complex texture features, such as more complex illumination distribution or deformation distribution, so we rank the larger exemplars in front of smaller ones. At the meantime, we can also easily filter out the exemplar with too small image size based on S_i.

On the other hand, we should note that the weight of b_1 is larger than b_2 and b_3, so the output probability P_i in our Trimmed T-CNN is still the determining factor to obtain best texture exemplar. Therefore, we can still obtain a set of best texture exemplars with multiple different kinds of dominate texture contents based on above scoring and ranking scheme.

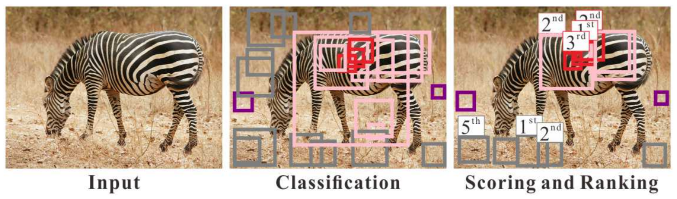


Fig. 6. Texture exemplar classification, scoring and ranking. Red, pink, violet and grey squares represent the extracted texture exemplars in the classes of S_3R_1, S_2R_2, S_1R_2, and S_1R_1, respectively.

IV. Results and Discussions

In our experiments, we implemented our method using Caffe on a single NVIDIA RTX 2080ti (11 GB RAM) in Ubuntu 16.04 LTS. For the network learning from scratch, we set the initial learning rate to be 0.001. In the fine-tuning stage after a rough convergence, we set learning rate to be 0.0001, where the following weight decay rate is 0.0005. For a faster convergence, we also employed the SGD optimizer in our training process. As our training set is relative small, which is not as huge as ImageNet, we set the batch size to be 32. Usually, our model can converge after 500 epochs in most of our training experiments.

A. Texture Exemplar Extraction

In our experiments, our Trimmed T-CNN is first trained with our collected standard exemplar dataset, which includes 10080 best texture exemplars and six classes exemplars with different scales, illuminations and rotations. For deep texture exemplar extraction, we have used hundreds of natural images to evaluate our exemplar extraction accuracy and efficiency. Note that, an ideal texture exemplar in each of the six classes can contain various material or colors, so our model does not discriminate textures with different kinds of material and colors. To test our Trimmed T-CNN framework in handling different kinds of nature images with different difficulties, we have divided our collect test cases into three levels.

The first difficult level is the simple cases that with purely flat cases, which do not include any other image noise or perspective contents. Typical examples for the level of cases are as shown in Fig. 7. From the results, we can see that our method successfully extracted multiple classes of flat texture exemplars, even where has only one kind of texture in the original natural images. For example, one may only crop one kind of texture exemplar for the carpet and pill cases. Our Trimmed T-CNN based deep learning method can extract 3 kinds texture exemplar for the carpet (S_1R_2, S_2R_2, S_3R_2) and pill cases (S_1R_1, S_2R_1, S_3R_1), with high score values, indicating the excellent quality of the exemplars. From the extracted exemplars, we also found that both random and regular cases with different scale exemplars can be handled well with our Trimmed T-CNN based deep learning framework. For the convenience of typography, we resize all the texture exemplars to the same size.

The second difficult level is the nearly flat cases, which have slight perspective and contain non-texture contents. Typical examples cases are as shown in Fig. 8. From the results, we can see that our method successfully extracted multiple classes of nearly flat exemplars too, even where has slight perspective and containing non-texture contents in the giraffe

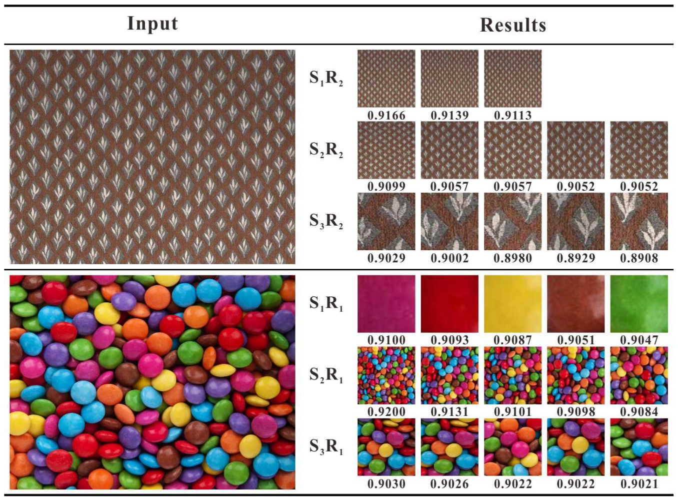


Fig. 7. Deep texture exemplar extraction for images with purely flat cases, which do not include any other noise or perspective regions.

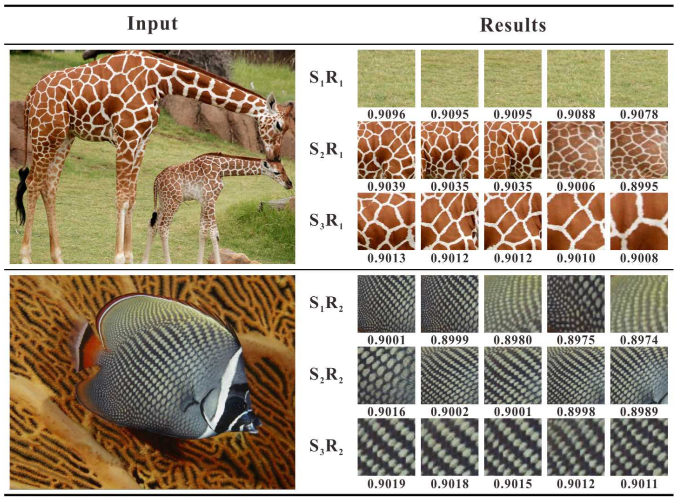


Fig. 8. Deep texture exemplar extraction for images with nearly flat cases, which contain slight perspective and non-texture regions.

and fish cases. The exemplars extracted for the giraffe(S_1R_1, S_2R_1, S_3R_1) show that our method can handle the images with multiple types of textures (turf texture and giraffe texture). On the other hand, the case of fish (S_1R_2, S_2R_2, S_3R_2) also demonstrates that our Trimmed T-CNN based deep learning framework can process well the images with random texture and various illumination distributions.

The third difficult level is the challenging cases, which have serious perspective and deformations, such as the cases shown in Fig. 9. From the results, we also clearly see that our method can still extract multiple classes of excellent exemplars with high scores, even where has serious perspective and deformations in the building and the terraced field. The exemplars extracted for the building (S_2R_1, S_2R_2, S_3R_1) and terraced field (S_1R_1, S_2R_1, S_3R_2) show that our method can output both random and regular types of exemplars with different scales for a single natural image. Note that, the artists may only crop one type of texture exemplar, and other types of excellent exemplars may be easily ignored.

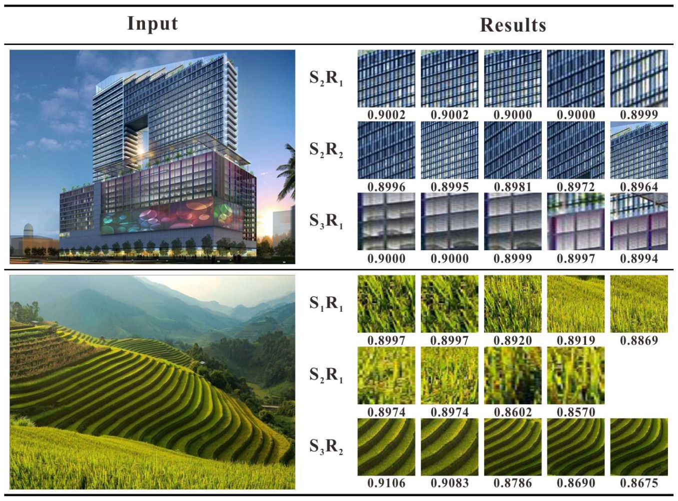


Fig. 9. Deep texture exemplar extraction for images with challenging cases, which have serious perspective and deformations. For the building example, the first row is R_1 because the color variation in the window is more random due to the perspective reflection. On the contrary, the color distribution is more uniform in the second row, so the exemplars are R_2.

B. Comparisons on Texture Exemplar Extraction

On the other hand, we also compare our method with three state-of-the-art methods, including Wu et al. [32], Dai et al. [26] and Lockerman et al. [28]. Typical results for comparisons of Wu's method, Dai's method and ours are as shown in Fig. 10. We can easily see that all three methods can extract the desired texture exemplars consisting to the dominative textures in the input images. From the three group results, Dai's method selects a number of image patches and measures its synthesizability, which can be treated as a descriptor and mostly reflect the quality of the extracted texture patches. Based on the global and local textureness descriptor, Wu's method can also obtain several desired texture exemplars. For a more clear scoring comparison, we normalized Wu's and Dai's method values into [0,1]. Without a training process, both Wu's method and Dai's method require a specific texture descriptor for texture exemplar extraction. Instead, our method can extract multiple types of texture exemplars, without requiring any specific feature detection and descriptors. From the comparison for the same texture exemplar, our method can achieve better performance by extracting better exemplars with much more uniform structure, illumination and deformation distributions, such as the desert (S_3R_1, S_1R_2, S_3R_2), zebra (S_1R_1, S_3R_1, S_3R_2) and palace (S_2R_1, S_1R_2, S_2R_2) exemplars.

Lockerman's method is also capable to extract texture patterns in multiple scales and levels, but their method usually requires multiple user input to specify the initial location and s-cale of the desired texture. They mainly employ a fast iteration method using diffusion manifolds to locate texture exemplars. In our comparisons, we first selected typical images from Lockerman's webpage, run Wu's method and our method on them for comparison with Lockerman's method. To guarantee a fair comparison, all of competitors were implemented and run on the same computing environments. Moreover, all other methods were also fine-tuned on our dataset to demonstrate their best performances when compared with our Trimmed

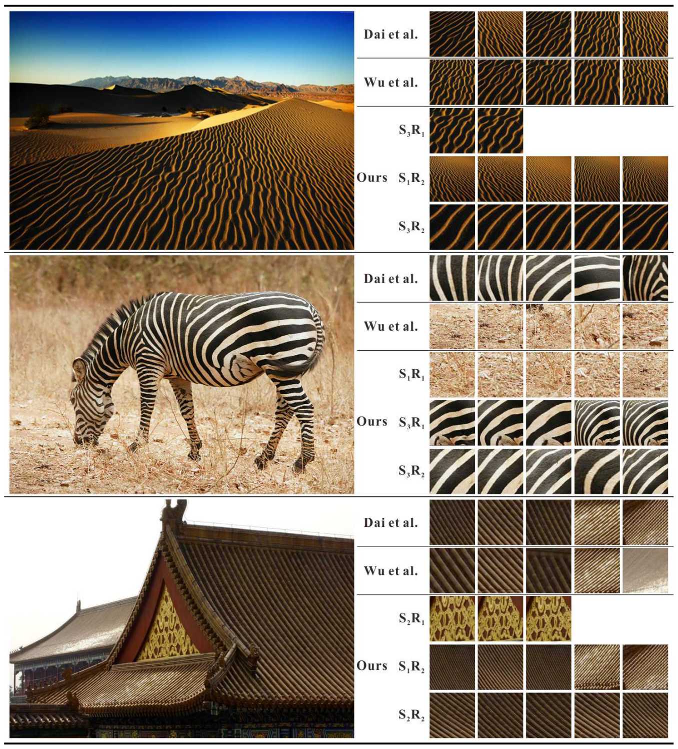


Fig. 10. Comparisons of Dai et al. [26], Wu et al. [32] and our method.

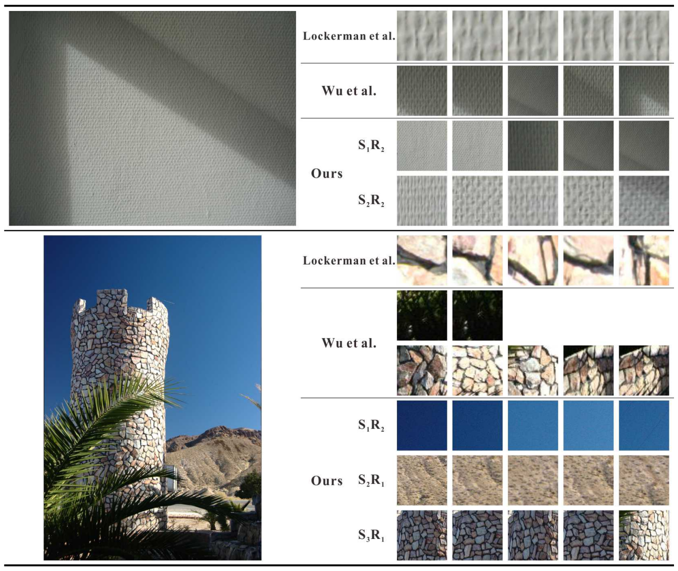


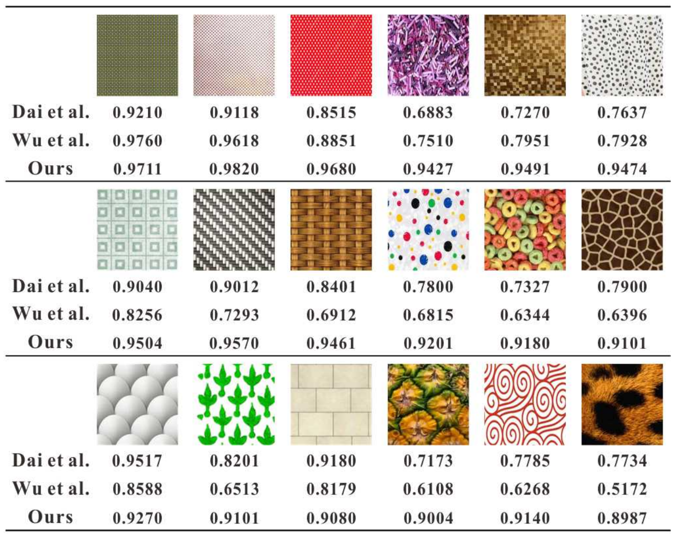
Fig. 11. Comparisons of Lockerman et al. [28], Wu et al. [32] and our method.

T-CNN. Typical comparison results are as shown in Fig. 11. From the results, we also observe that our method can extract multiple ideal texture exemplars, which have better qualities than Wu's method and Lockerman's method. The dagstuhl (S_1R_2, S_2R_2) and tower (S_1R_2, S_2R_1, S_3R_1) extracted exemplars demonstrate that our method successfully extracts several best meaningful exemplars with different scales, regularities and texture contents. Lockerman's method only focus on extracting small texels of the dominant texture, so exemplars with small sizes were always extracted, which usually cannot cover a meaningful exemplar for texture synthesis. More importantly, our method is an automatic exemplar extraction method, without requiring any user input to specify the initial location and scale of the desired texture.

C. Comparisons on Texture Exemplar Metrics

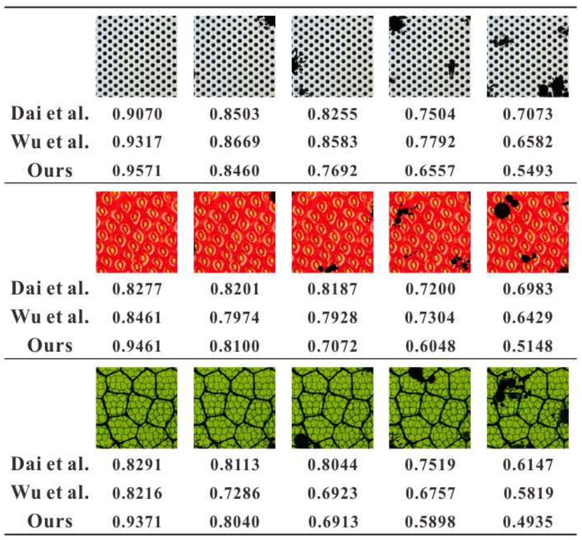
As we all know, it is very difficult to use a formula or sentence to define what a good exemplar is. In this paper, what is a good exemplar is defined by the standard ideal texture exemplar dataset collected by our invited artists. Our standard dataset also does not assume a high score for the exemplar with a better synthesizability in future texture synthesis, as we need to achieve multi-scale and multi-regularity texture exemplar extractions. If we over emphasize the synthesizability of the extracted exemplars, we may easily lose the ideal exemplars with larger scales, complex structures or image variations. To investigate the advantage of our method on texture exemplar extraction, we randomly selected a number of ideal cases from our dataset and compared the scores obtained with different texture exemplar methods, including Wu et al. [32], Dai et al. [26] and ours. For a fair scoring comparison, we normalized Wu's and Dai's metric values into [0,1]. So all three methods can be compared under the same range. Typical results are as shown in Fig. 12. From the scores among different ideal cases, we can see that Dai's synthesizability scores is unstable. It usually obtained higher scores on the cases with fewer scale changes or better regularity distributions. Dai's synthesizability metric would drop sharply when the texture scale changing growing or the regularity going down, such as the cases shown in the 4-6 cases on each row of Fig. 12. This also explains well why Dai's metric would skip several kinds of texture exemplars in the texture exemplar extraction, which may have excellent quality but with scale changing or low regularity for the texture content. Compared with Dai's synthesizability scores, Wu's metric obtained a better consistent performance by combining global and local textureness in the design of the texture exemplar metric. However, we also found that Wu's metric still cannot handle well the changing of scale and regularity for the texture exemplars, such as the low score obtained for an excellent texture exemplar of pineapple in the 3^{rd} row. Instead, our Trimmed T-CNN method can always obtain good scores for ideal texture exemplars with different scales or regularities, which also explains well why our method usually does not skip the excellent cases in the texture exemplar extraction.

On the other hand, we also measured the sensitivity on different degrees of occlusions or noises in the texture exemplars for the three competitors. As shown in Fig. 13, we synthesized different degrees of occlusions of the texture exemplars on the same case, and performed Wu et al. [32], Dai et al. [26] and our method to evaluate the quality of the texture exemplars. From the results, we can see that both Dai's synthesizability method and Wu's textureness method are not very sensitive with the occlusion or noise in the texture exemplars. Their scores drop slowly with the increasing of the occlusions or noises in the texture exemplars. In contrast, our Trimmed T-CNN method is very sensitive with the degrees of occlusions in the texture exemplars. From the three groups of texture



Dai et al.	0.9210	0.9118	0.8515	0.6883	0.7270	0.7637
Wu et al.	0.9760	0.9618	0.8851	0.7510	0.7951	0.7928
Ours	0.9711	0.9820	0.9680	0.9427	0.9491	0.9474
Dai et al.	0.9040	0.9012	0.8401	0.7800	0.7327	0.7900
Wu et al.	0.8256	0.7293	0.6912	0.6815	0.6344	0.6396
Ours	0.9504	0.9570	0.9461	0.9201	0.9180	0.9101
Dai et al.	0.9517	0.8201	0.9180	0.7173	0.7785	0.7734
Wu et al.	0.8588	0.6513	0.8179	0.6108	0.6268	0.5172
Ours	0.9270	0.9101	0.9080	0.9004	0.9140	0.8987

Fig. 12. Comparisons of texture exemplar method on different ideal examples for Dai et al. [26], Wu et al. [32] and our method.



Dai et al.	0.9070	0.8503	0.8255	0.7504	0.7073
Wu et al.	0.9317	0.8669	0.8583	0.7792	0.6582
Ours	0.9571	0.8460	0.7692	0.6557	0.5493
Dai et al.	0.8277	0.8201	0.8187	0.7200	0.6983
Wu et al.	0.8461	0.7974	0.7928	0.7304	0.6429
Ours	0.9461	0.8100	0.7072	0.6048	0.5148
Dai et al.	0.8291	0.8113	0.8044	0.7519	0.6147
Wu et al.	0.8216	0.7286	0.6923	0.6757	0.5819
Ours	0.9371	0.8040	0.6913	0.5898	0.4935

Fig. 13. Comparisons of texture exemplar method on the cases with different qualities for Dai et al. [26], Wu et al. [32] and our method.

TABLE I
ACCURACY COMPARISON AMONG DIFFERENT DEEP CNN ARCHITECTURES ON OUR STANDARD EXEMPLAR DATASET.

CNN model	AlexNet	VGG-M	Deep-TEN	FV-CNN	T-CNN	Trimmed T-CNN
Accuracy (%)	92.60	93.82	94.41	94.35	94.32	94.44
Weights (MB)	238.14	310.12	244.81	310.49	76.62	19.16

TABLE II
ACCURACY COMPARISON AMONG DIFFERENT DEEP CNN ARCHITECTURES ON SIMPLE EXEMPLAR DATASET ONLY INCLUDING PURELY FLAT CASES, WHERE DO NOT HAVE ANY OTHER NOISE OR PERSPECTIVE REGIONS.

CNN model	AlexNet	VGG-M	Deep-TEN	FV-CNN	T-CNN	Trimmed T-CNN
Accuracy (%)	93.81	94.93	95.41	95.33	95.32	95.83
Weights (MB)	238.14	310.12	244.81	310.49	76.62	19.16

TABLE III
ACCURACY COMPARISON AMONG DIFFERENT DEEP CNN ARCHITECTURES ON MEDIUM DIFFICULTY EXEMPLAR DATASET, WHICH INCLUDES SLIGHT PERSPECTIVE AND CONTAIN NON-TEXTURE REGIONS.

CNN model	AlexNet	VGG-M	Deep-TEN	FV-CNN	T-CNN	Trimmed T-CNN
Accuracy (%)	92.51	93.77	94.44	94.10	94.12	94.53
Weights (MB)	238.14	310.12	244.81	310.49	76.62	19.16

TABLE IV
ACCURACY COMPARISON AMONG DIFFERENT DEEP CNN ARCHITECTURES ON CHALLENGING EXEMPLAR DATASET, WHICH HAVE SERIOUS PERSPECTIVE AND DEFORMATIONS.

CNN model	AlexNet	VGG-M	Deep-TEN	FV-CNN	T-CNN	Trimmed T-CNN
Accuracy (%)	92.22	93.21	93.47	93.41	93.11	93.42
Weights (MB)	238.14	310.12	244.81	310.49	76.62	19.16

exemplars in Fig. 13, we can see that our method always obtained a proportional dropping score with the increasing of the occlusions or noises. Exactly, one may suggest to simply adapt a threshold to filter out the exemplars with heavy noises or occlusions, but we still require a better algorithm which is more sensitive to the image noises and occlusions, especially for the small occlusions and slight noises. The high sensitivity on occlusions or noises for our method improves the ability in filtering out the interference of defective cases, which also improves our ability in identifying multiple kinds of ideal texture exemplars in the texture exemplar extraction.

D. Comparisons with Different Deep CNN Architectures

In addition, we also implemented different deep CNN architectures for exemplar classification evaluations on our standard exemplar dataset, including AlexNet [33], VGG-M [55],Deep-TEN [56], FV-CNN [35] and T-CNN [40]. In our experiments, 80% exemplars in our dataset are used for the training set, and 20% are used for the test set. We run three hundred ideal exemplars with known classification ground truth to evaluate them in our standard exemplar dataset and calculate an average accuracy for all competitors. The accuracy and number of weights (in MB) for different architectures are as shown in Table I-IV. From the statistical results, we can see that our CNN architecture generally outperforms all competitors in terms of accuracy in texture exemplar classification. As both Deep-TEN and FV-CNN are designed for texture classification, they are more focused on discrimination for hundreds kinds of texture recognitions, such as grassland, building and etc. In contrast, we do not emphasize hundreds kinds of texture recognitions in our Trimmed T-CNN, and only focus on the extraction of the six kinds of ideal texture exemplars with different scales and regularities. Therefore, Deep-TEN and FV-CNN usually have much more complex and deeper architectures which are unnecessary for our experiments. As they are not designed for automatic texture exemplar extractions, their performances are

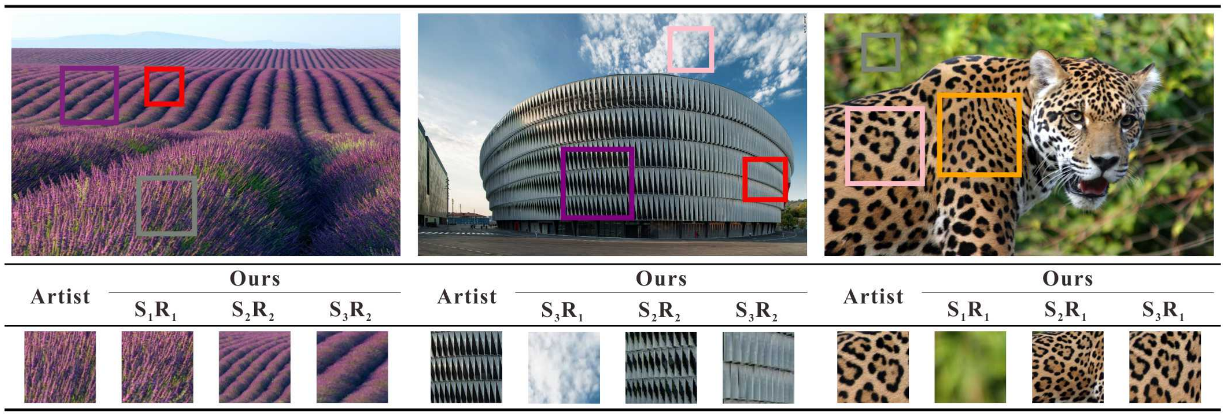


Fig. 14. Comparison between our method and the artists.

not as good as ours. More importantly, our Trimmed T-CNN has only 24% parameters of the original T-CNN [40], and no more than 10% weights in parameters compared with the other state-of-the-art architectures.

E. User Study and Applications

Furthermore, we also compared our method with the manually texture exemplar extraction by the artists. In our experiments, we first invited three artists to manually label texture exemplars. To minimize the bias, the best texture exemplar is voted out from the three labeled texture patches by the three artists, as shown in Fig. 14. So the manually cropped exemplars can be considered as the ground-truth for exemplar extraction. We can compare them with our Trimmed T-CNN learning method. From the cases shown in Fig. 14, our results are very close to the ground-truth of artists for the same kind of texture exemplar. Also, our method can provide much more diverse exemplars with different scales, regularities, illumination and deformation distributions for us to choose in future applications, as shown in the lavender (S_1R_1, S_2R_2, S_3R_2), cylinder building (S_3R_1, S_2R_2, S_3R_2) and tiger (S_1R_1, S_2R_1, S_3R_1).

Similarly, we also perform the user study on the different texture exemplar extraction methods. For the user study comparison, we also choose three state-of-the-art methods, including Wu et al. [32], Dai et al. [26] and Lockerman et al. [28]. We randomly select three groups natural images with three different difficulty levels, each group including 100 images for texture exemplar extraction. We run our method, Wu et al.'s method, Dai et al.'s method and Lockerman et al.'s method on the three groups images and got the extracted texture exemplar results for each competitor. We then asked the artists to choose the satisfactory exemplars, and collected the statistical results. The number of satisfactory exemplars are plotted as a function of the total number of test images, as shown in Fig. 15, Fig. 16 and Fig. 18. From the statistical results, we found that the satisfactory performances for each competitor declining with the difficulty increasing for the three different datasets. Also, we still can see that our Trimmed T-CNN architecture generally outperforms all competitors, with more satisfy exemplars selected for the users. More importantly, the three state-of-the-art methods are feature-based methods, which require extracting specific image features or pre-defining the initial texture location before texture exemplar extraction. Instead, our Trimmed T-CNN methods are learning-based methods and fully automatic, which do not require any user input to specify the initial image feature, location and scale of the desired texture.

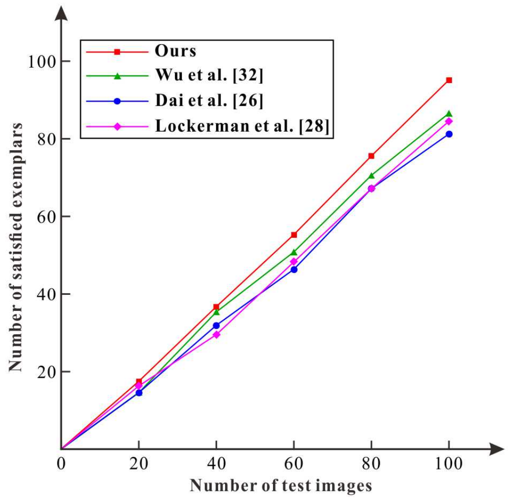


Fig. 15. Statistical comparison of user study among different methods on simple exemplar dataset only including purely flat cases, where do not have any other noise or perspective regions.

To further evaluate the synthesizability of the extracted texture exemplars, we generated graph-cut textures with extensive resolutions for the applications of texture synthesis and texture replacements, as shown in Fig. 17. From the synthesis and replacement results, we can see that both the two kinds of applications (texture synthesis and texture replacements) were well done based on the texture exemplars extracted using our Trimmed T-CNN method.

Finally, we also performed a time statistics to evaluate the speed of our method in extracting texture exemplars. We collected a number of natural images with different resolutions as the input for time statistics. They were resized into three different sizes, including 512×512, 1024×1024 and 2048×2048. All resized images were put into our Trimmed T-

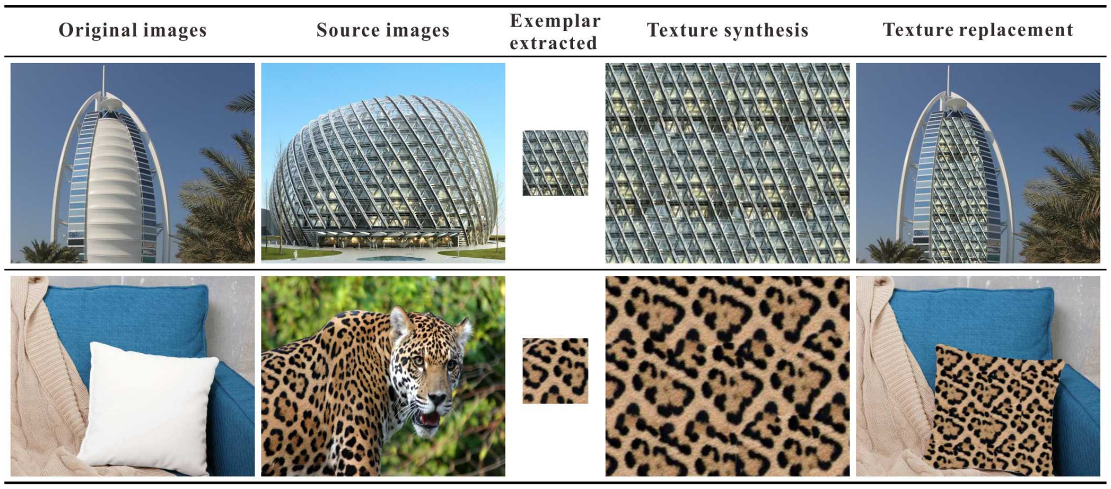


Fig. 17. Applications of texture synthesis and replacements using our extracted texture exemplars.

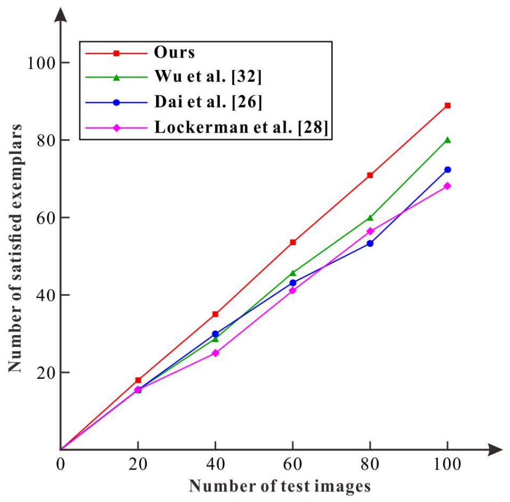


Fig. 16. Statistical comparison of user study among different methods on medium difficulty exemplar dataset, which includes slight perspective and contain non-texture regions.

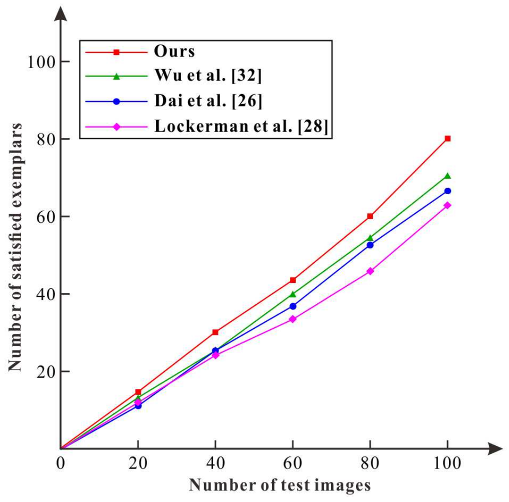


Fig. 18. Statistical comparison of user study among different methods on challenging exemplar dataset, which have serious perspective and deformations.

TABLE V
TIMING STATISTICS FOR OUR TRIMMED T-CNN METHOD ON THE INPUT IMAGE WITH DIFFERENT RESOLUTIONS.

Resolution	Time in Second			
	SS	Classification	SR	Total
512×512	1.1632	3.3247	0.0001	4.4880
1024×1024	1.8171	3.7702	0.0001	5.5874
2048×2048	2.5066	4.4442	0.0002	6.9510

∗ Selective Search (SS) Scoring and Ranking (SR)

CNN learning system, the time cost in all steps for extracting texture exemplars were collected. To minimize the error for a single test, we repeated ten times for the timing collection and calculated the average timing for each step. The results are as shown in the Table V. From the results, we can see that our timing statistics mainly includes three steps for automatic texture exemplar extraction, including selective search(SS), classification, scoring and ranking(SR). More specifically, most of the time is cost on the selective searching step and classification step. It generally cost 4-6 seconds to extract the final exemplar results for an image with about tens of millions of pixels. Note that, there is without any GPU acceleration for our current implementation.

F. Discussion and Limitations

Moreover, we also have tested our method on the Dai et al.'s dataset, as well as DTD and UIUC datasets. Because above datasets are labelled based on different kinds of material, which usually contain tens of classes, our Trimmed T-CNN cannot obtain high accuracy for material recognition when compared with the complex network, such as the work recently published for material recognition [49], [50]. However, above complex network will be overfitting on our collected dataset, which only includes six classes according to different scales

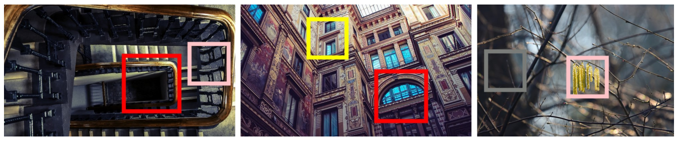


Fig. 19. Failure cases. Our method still has limitations when the images have irregular shape and color distribution, or too complex scale, deformation, and lighting variations. Red, pink, yellow and grey squares represent the extracted poor texture exemplars in the classes of $\mathrm{S_3R_1}$, $\mathrm{S_2R_2}$, $\mathrm{S_3R_2}$, and $\mathrm{S_1R_1}$, respectively.

and regularities. As our Trimmed T-CNN is a relative shallow and light weight network, our experiments have verified that the trained Trimmed T-CNN is not overfitting on our collected dataset, as shown in Table I-IV.

On the other hand, our method still has some limitations. If the input images have very irregular shape and color distribution, or too complex scale, deformation, and lighting variations, our method will be failed, as shown in Fig. 19. In addition, our method also cannot be directly applied for dynamic texture exemplar extraction, where the dynamic texture sample often contains a sequence of images. Based on the previous dynamic texture framework [57], we may develop a deeper CNN network to learn the synthesizability of dynamic texture samples from videos in the future work.

V. Conclusion

In this paper, we presented an automatic texture exemplar extraction based on Trimmed T-CNN. To the best of our knowledge, our Trimmed T-CNN is the first deep learning framework for automatic texture extraction. To learn excellent texture exemplar descriptors, we also have designed the first standard ideal exemplar dataset for deep texture exemplar extraction, which contains thousands of desired texture exemplars. Specifically, Trimmed T-CNN were implemented to be our filter banks for texture exemplar classification and recognition. By training our Trimmed T-CNN with our standard ideal exemplar dataset, we can achieve texture exemplar extraction based on classification and recognition. We also implemented an improved selective search algorithm to extract the potential texture exemplar patches, where all candidates can be efficiently put into our Trimmed T-CNN for learning ideal texture exemplars. Finally, we can obtain optimal texture exemplars with a scoring and ranking scheme. Our texture exemplar extraction system is evaluated with numerous kinds of textures. We also compared our Trimmed T-CNN with state-of-the-art deep CNN architectures (AlexNet, VGG-M, Deep-TEN and FV-CNN). User study and texture replacement applications based on the extracted exemplars were also conducted and convincing results demonstrated its effectiveness. For further study, it will be interesting to extend our method to extract dynamic texture exemplars [57] from videos.

References

[1] L. Raad, A. Davy, A. Desolneux, and J.-M. Morel, "A survey of exemplar-based texture synthesis," *arXiv preprint arXiv:1707.07184*, 2017.

[2] S. Zhou, J. Wang, R. Shi, Q. Hou, Y. Gong, and N. Zheng, "Large margin learning in set-to-set similarity comparison for person reidentification," *IEEE Transactions on Multimedia*, vol. 20, no. 3, pp. 593–604, 2017.

[3] T. Yu, L. Wang, C. Da, H. Gu, S. Xiang, and C. Pan, "Weakly semantic guided action recognition," *IEEE Transactions on Multimedia*, vol. 21, no. 10, pp. 2504–2517, 2019.

[4] K. Gu, Z. Xia, J. Qiao, and W. Lin, "Deep dual-channel neural network for image-based smoke detection," *IEEE Transactions on Multimedia*, 2019.

[5] H. Xiao, J. Feng, Y. Wei, M. Zhang, and S. Yan, "Deep salient object detection with dense connections and distraction diagnosis," *IEEE Transactions on Multimedia*, vol. 20, no. 12, pp. 3239–3251, 2018.

[6] F. Chen, R. Ji, J. Su, D. Cao, and Y. Gao, "Predicting microblog sentiments via weakly supervised multimodal deep learning," *IEEE Transactions on Multimedia*, vol. 20, no. 4, pp. 997–1007, 2017.

[7] J. Xie, G. Dai, and Y. Fang, "Deep multimetric learning for shape-based 3d model retrieval," *IEEE Transactions on Multimedia*, vol. 19, no. 11, pp. 2463–2474, 2017.

[8] J. Choi, H. Cho, J. Song, and S. M. Yoon, "Sketchhelper: Real-time stroke guidance for freehand sketch retrieval," *IEEE Transactions on Multimedia*, vol. 21, no. 8, pp. 2083–2092, 2019.

[9] X. Yang, H. Mei, J. Zhang, K. Xu, B. Yin, Q. Zhang, and X. Wei, "Drfn: Deep recurrent fusion network for single-image super-resolution with large factors," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 328–337, 2018.

[10] J. Lin, L.-Y. Duan, S. Wang, Y. Bai, Y. Lou, V. Chandrasekhar, T. Huang, A. Kot, and W. Gao, "Hnip: Compact deep invariant representations for video matching, localization, and retrieval," *IEEE Transactions on Multimedia*, vol. 19, no. 9, pp. 1968–1983, 2017.

[11] J. Wang, W. Wang, and W. Gao, "Multiscale deep alternative neural network for large-scale video classification," *IEEE Transactions on Multimedia*, vol. 20, no. 10, pp. 2578–2592, 2018.

[12] A. Lagae, S. Lefebvre, R. Cook, T. DeRose, G. Drettakis, D. S. Ebert, J. P. Lewis, K. Perlin, and M. Zwicker, "A survey of procedural noise functions," in *Computer Graphics Forum*, vol. 29, no. 8. Wiley Online Library, 2010, pp. 2579–2600.

[13] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. S. Lempitsky, "Texture networks: Feed-forward synthesis of textures and stylized images." in *ICML*, vol. 1, no. 2, 2016, p. 4.

[14] N. Jetchev, U. Bergmann, and R. Vollgraf, "Texture synthesis with spatial generative adversarial networks," *arXiv preprint arXiv:1611.08207*, 2016.

[15] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2. IEEE, 1999, pp. 1033–1038.

[16] A. A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 2001, pp. 341–346.

[17] C. Li and M. Wand, "Combining markov random fields and convolutional neural networks for image synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2479–2486.

[18] L. Raad, A. Desolneux, and J.-M. Morel, "A conditional multiscale locally gaussian texture synthesis algorithm," *Journal of Mathematical Imaging and Vision*, vol. 56, no. 2, pp. 260–279, 2016.

[19] L. A. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis and the controlled generation of natural stimuli using convolutional neural networks," in *Bernstein Conference 2015*, 2015, pp. 219–219.

[20] O. Sendik and D. Cohen-Or, "Deep correlations for texture synthesis," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 5, p. 161, 2017.

[21] U. Bergmann, N. Jetchev, and R. Vollgraf, "Learning texture manifolds with the periodic spatial gan," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 469–477.

[22] M. Elad and P. Milanfar, "Style transfer via texture synthesis," *IEEE Transactions on Image Processing*, vol. 26, no. 5, pp. 2338–2351, 2017.

[23] M.-C. Yeh and S. Tang, "Improved style transfer by respecting inter-layer correlations," *arXiv preprint arXiv:1801.01933*, 2018.

[24] L.-Y. Wei, J. Han, K. Zhou, H. Bao, B. Guo, and H.-Y. Shum, "Inverse texture synthesis," in *AcM Transactions on Graphics (TOG)*, vol. 27, no. 3. ACM, 2008, p. 52.

[25] W. Wang and M. Hua, "Extracting dominant textures in real time with multi-scale hue-saturation-intensity histograms," *IEEE Transactions on Image Processing*, vol. 22, no. 11, pp. 4237–4248, 2013.

[26] D. Dai, H. Riemenschneider, and L. Van Gool, "The synthesizability of texture examples," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3027–3034.

[27] Y. D. Lockerman, S. Xue, J. Dorsey, and H. Rushmeier, "Creating texture exemplars from unconstrained images," in *2013 International Conference on Computer-Aided Design and Computer Graphics*. IEEE, 2013, pp. 397–398.
[28] Y. Lockerman, H. Rushmeier, and J. Dorsey, "Systems and methods for creating texture exemplars," Apr. 14 2015, uS Patent 9,007,373.
[29] Y. D. Lockerman, B. Sauvage, R. Allègre, J.-M. Dischler, J. Dorsey, and H. E. Rushmeier, "Multi-scale label-map extraction for texture synthesis." *ACM Trans. Graph.*, vol. 35, no. 4, pp. 140–1, 2016.
[30] J. Lu, J. Dorsey, and H. Rushmeier, "Dominant texture and diffusion distance manifolds," in *Computer Graphics Forum*, vol. 28, no. 2. Wiley Online Library, 2009, pp. 667–676.
[31] J. Moritz, S. James, T. S. Haines, T. Ritschel, and T. Weyrich, "Texture stationarization: Turning photos into tileable textures," in *Computer Graphics Forum*, vol. 36, no. 2. Wiley Online Library, 2017, pp. 177–188.
[32] H. Wu, X. Lyu, and Z. Wen, "Automatic texture exemplar extraction based on global and local textureness measures," *Computational Visual Media*, vol. 4, no. 2, pp. 173–184, 2018.
[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
[34] M. Cimpoi, S. Maji, I. Kokkinos, and A. Vedaldi, "Deep filter banks for texture recognition, description, and segmentation," *International Journal of Computer Vision*, vol. 118, no. 1, pp. 65–94, 2016.
[35] M. Cimpoi, S. Maji, and A. Vedaldi, "Deep filter banks for texture recognition and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3828–3836.
[36] T.-Y. Lin, A. RoyChowdhury, and S. Maji, "Bilinear cnn models for fine-grained visual recognition," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1449–1457.
[37] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," *arXiv preprint arXiv:1809.02165*, 2018.
[38] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.
[39] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.
[40] V. Andrearczyk and P. F. Whelan, "Using filter banks in convolutional neural networks for texture classification," *Pattern Recognition Letters*, vol. 84, pp. 63–69, 2016.
[41] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.
[42] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, "Describing textures in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3606–3613.
[43] Y. Xu, H. Ji, and C. Fermüller, "Viewpoint invariant texture description using fractal analysis," *International Journal of Computer Vision*, vol. 83, no. 1, pp. 85–100, 2009.
[44] S. Lazebnik, C. Schmid, and J. Ponce, "A sparse texture representation using local affine regions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1265–1278, 2005.
[45] G. Kylberg, *Kylberg Texture Dataset v. 1.0*. Centre for Image Analysis, Swedish University of Agricultural Sciences and , 2011.
[46] E. Hayman, B. Caputo, M. Fritz, and J.-O. Eklundh, "On the significance of real-world conditions for material classification," in *European conference on computer vision*. Springer, 2004, pp. 253–266.
[47] B. Caputo, E. Hayman, and P. Mallikarjuna, "Class-specific material categorisation," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 2. IEEE, 2005, pp. 1597–1604.
[48] L. Sharan, R. Rosenholtz, and E. Adelson, "Material perception: What can you see in a brief glance?" *Journal of Vision*, vol. 9, no. 8, pp. 784–784, 2009.
[49] S. Bell, P. Upchurch, N. Snavely, and K. Bala, "Material recognition in the wild with the materials in context database," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3479–3487.
[50] J. Xue, H. Zhang, K. Dana, and K. Nishino, "Differential angular imaging for material recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 764–773.
[51] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.
[52] M. D. Zeiler, G. W. Taylor, R. Fergus *et al.*, "Adaptive deconvolutional networks for mid and high level feature learning." in *ICCV*, vol. 1, no. 2, 2011, p. 6.
[53] X. Liu, L. Jiang, T.-T. Wong, and C.-W. Fu, "Statistical invariance for texture synthesis," *IEEE transactions on visualization and computer graphics*, vol. 18, no. 11, pp. 1836–1848, 2012.
[54] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems*, 2015, pp. 1135–1143.
[55] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," *arXiv preprint arXiv:1405.3531*, 2014.
[56] H. Zhang, J. Xue, and K. Dana, "Deep ten: Texture encoding network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 708–717.
[57] F. Yang, G.-S. Xia, D. Dai, and L. Zhang, "Learning the synthesizability of dynamic texture samples," *IEEE Transactions on Image Processing*, vol. 28, no. 5, pp. 2502–2517, 2018.

Huisi Wu is currently an Associate Professor in the College of Computer Science and Software Engineering, Shenzhen University. He received his B.Sc. and M.Sc. degrees in computer science from Xi'an Jiaotong University in 2004 and 2007, respectively. He obtained his Ph.D. degree in computer science from the Chinese University of Hong Kong in 2011. His research interests are in computer graphics, image processing, and medical imaging.

Wei Yan received his B.S. degree in computer science and technology from Hunan Institute of Science and Technology in 2017. Currently he is studying at Shenzhen University for his master degree. His research interests include computer vision, texture analysis and pattern recognition.

Ping Li (M'14) received the Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong, Shatin, Hong Kong.

He is currently a Research Assistant Professor with The Hong Kong Polytechnic University, Kowloon, Hong Kong. His current research interests include image/video stylization, artistic rendering and synthesis, and creative media. He has one image/video processing national invention patent, and has excellent research project reported worldwide by *ACM TechNews*.

Zhenkun Wen received his M.Sc.degree in science and technology from Tsinghua University in 1999. He is currently a Professor of computing and software, and director of the Science and Technology Department of Shenzhen University. His research interests are in computer graphics and video information security.