Deep Neural Representation Guided Face Sketch Synthesis

Bin Sheng, Ping Li, Chenhao Gao, and Kwan-Liu Ma, *Fellow, IEEE*

Abstract—Face sketch synthesis shows great applications in a lot of fields such as online entertainment and suspects identification. Existing face sketch synthesis methods learn the patch-wise sketch style from the training dataset containing photo-sketch pairs. These methods manipulate the whole process directly in the field of RGB space, which unavoidably results in unsmooth noises at patch boundaries. If denoising methods are used, the sketch edges would be blurred and face structures could not be restored. Recent researches of feature maps, which are the outputs of a certain neural network layer, have achieved great success in texture synthesis and artistic image generation. In this paper, we reformulate the face sketch synthesis problem into a neural network feature maps based optimization task. Our results accurately capture the sketch drawing style and make full use of the whole stylistic information hidden in the training dataset. Unlike former feature map based methods, we utilize the Enhanced 3D PatchMatch and cross-layer cost aggregation methods to obtain the target feature maps for the final results. Multiple experiments have shown that our approach imitates hand-drawn sketch style vividly, and has high-quality visual effects on CUHK, AR, XM2VTS and CUFSF face sketch datasets.

Index Terms—Non-photorealistic rendering, face sketch synthesis, convolutional neural network (CNN), style transformation

1 Introduction

Face sketch portrait has been proved to show a wide range of usages in lots of fields, e.g., instant message communications, and suspects identification. Especially in the process of criminal investigation, public security organization needs to draw a sketch of suspects according to the description of witnesses. If a sketch dataset is built, the suspects could be quickly identified by matching the sketches with the dataset [1]. Generally, this is a non-photorealistic rendering (NPR) problem [2], [3], [4], [5], [6]. The key to a successful NPR design lies in how to express stylistic textures of target style, while fully preserving contents information of original images. Many researches have made a great attempt in such fields [7], [8], [9], [10]. In all subfields of non-photorealistic art, sketch might be a strongly attracting artistic form. A sketch is defined as a kind of drawing using purely line strokes without any color. It is an ancient art form and be carried forward by famous artists like Leonardo da Vinci and Michelangelo during the Renaissance. Famous sketch drawings include "Self Portrait" by da Vinci and "Libyan Sibyl" by Michelangelo. Researches on how an image transfer into a sketch would help us understand the intrinsic essence of art.

- *B. Sheng and C. Gao are with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: shengbin@sjtu.edu.cn.*
- *P. Li is with the Faculty of Information Technology, Macau University of Science and Technology, Macau 999078, China. E-mail: pli@must.edu.mo.*
- *K.-L. Ma is with the Department of Computer Science, University of California, Davis, CA 95616. E-mail: ma@cs.ucdavis.edu.*

Manuscript received 30 Dec. 2017; revised 31 July 2018; accepted 14 Aug. 2018. Date of publication 20 Aug. 2018; date of current version 26 Oct. 2019. (Corresponding author: Bin Sheng.)
Recommended for acceptance by Y. Yu.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TVCG.2018.2866090

However, the photo-to-sketch conversion is still a challenging problem in rendering [12] and synthesis [13], [14], [15]. Intrinsically, the problem seems like a reconstruction which needs both middle-level information (the object contour such as nose, mouse, and eyes) from the photo, and low-level texture information (the hair texture and the shadow brush near chins) from the sketch. Existing state-of-the-art face sketch synthesis methods are mainly exemplar-based methods. Such methods [16], [17], [18], [19], [20], [21] learn the whole texture and face contour information from a dataset consisting of several pairs of sketch and face photo, and could generate the synthesized sketch to various face details. Nevertheless, exemplar-based methods are mostly based on patches in raw RGB space [17], [18], [19], [21], and based on the assumption that if two photo patches are similar, their sketch patches should also be similar. This assumption brings a contradictory drawback: if the selected patch size is big, the appearance of final sketch may not resemble the target photo since two non-linear transformations have been executed (one is from target photo to dataset photo, the other is from dataset photo to dataset sketch); if the selected patch size is small, the final sketch could be noisy since there is no extra space to eliminate border effects.

In recent years, with the development of GPUs, the convolutional neural network (CNN) based models have become a powerful solution to a wide range of tasks [22], [23], [24], [25]. Compared with traditional models, CNN-based models have a more potent features extraction capacity, which stores the image information inside feature maps over the whole layers. Fully convolutional network (FCN), which predicts pixel-wise object, is also qualified for the end-to-end image transformation [23], [26], [27]. In the face sketch synthesis areas, Zhang et al. [27] applied an end-to-end FCN to directly transform a face photo into a face

Fig. 1. Improvements for style transformation [11] with our proposed approach for face sketch synthesis. (a) Target photo. (b) Style image drawn by artist and serves as style image in [11]. (c) Synthesis sketch by [11]. (d) Synthesis sketch by our proposed approach.

sketch. However, the results from their work have blurred edges and result in fuzzy visual perception, which is far from a hand-drawn sketch quality. Besides FCN, another image generation model has been introduced [11], [28], where the final output images are treated as "weights" to be optimized in a traditional convolutional network framework, while a pre-trained network model such as VGG [29] is responsible for features extraction. For a lightweight mission, making full use of an existing trained network is confirmed to be more efficient and convenient.

Our approach is a combination of style transfer model [11] and patch-based model. In the first stage, we search for patches in our off-line dataset to synthesize target feature maps with our Enhanced 3D PatchMatch, which is a strengthened variation of original PatchMatch [30]. In the second stage, a cross-layer aggregated cost is utilized as the metric for multi-scale consistency, which is proved to make our results express more face sketch details (see Fig. 1). Our work makes the following three main contributions:

- Combining traditional patch-based image synthesis methods with the deep neural style transfer model;
- Using an Enhance 3D PatchMatch and cross-layer cost aggregation to select best patch, which is proved to be effective and could restore face sketch details;
- Better visual conceptions and detailed structures than previous methods on CUHK and AR datasets. User studies and statistics testified our superiority.

2 Related Work

2.1 Previous Sketch Synthesis Methods

Exemplar-based face sketch synthesis originated from face recognition. To overcome the modality gaps between photo and sketch [31], [32], Tang and Wang [16] treated each face sketch image as a reconstruction from eigenface in PCA representation. This work also notices that a face photo and its corresponding sketch has barely linear mapping relation due to the difference existing in both textures and shapes. Locally linear embedding [33] based method found that the nonlinear relationship between photo and sketch could be formulated as manifolds in different image spaces. Since this work, the following sketch synthesises start to focus on local patch learning rather than the global face learning. Wang and Tang [17] further used a multi-scale Markov Random Fields (MRF) model to select best patches and utilizes the patches to mosaic the final sketch. The work enlightens many follow-up researches. Zhou et al. [18] introduced Markov Weight Fields (MWF) to synthesize patches that do not exist in the training set. Their work also changes the original NP-hard MRF problem into a standard quadratic programming problem. Song et al. [19] improved the work of [33] by applying a brand-new sketch denoising method called Spatial Sketch Denoising (SSD). The advantage of this work is the real-time performance on GPU.

Some new exemplar-based sketch synthesis methods [34], [35], [36], [37] have sprung up and achieved good performance. [34] exquisitely decomposed original patch distribution in [17] into two parts and built two models to estimate them. Their key contribution is combining the MRF-based method [17] and MWF-based method [18] and applying into neighbor selection model and weight computation model. [36] focused on fast sketch synthesis and could generate a sketch within 1.5 seconds. Normally an image patch has very high dimension, e.g., a $H \times W$ patch would be represented by a vector of length WH. They applied PCA to reduce such dimension and thus speeded up distance computation greatly. The work effectively used random neighbor patches to reconstruct centralized target patch. The projection matrix is computed by random sampled neighbor patches. The approach is basically an unparameterized model with no need to learn global parameters.

The above methods focus mainly on the difference between raw image and sketch spaces. According to the manifold theory, the photo-sketch pair is considered as a transformation with same contents or geometry structures. This assumption has some rationality, but sketches drawn by artists have shown that sketch faces has more exaggeration than photo faces. If the sketch needs to resemble vividly with the original photo, the contents information should be learned directly from the original photo while the local strike containing texture information should be learned from the sketch dataset. A neural style transformation method [11] has achieved marvelous effects. With a pre-trained neural network, the contents and style information could be extracted richly in each layer's feature maps. In this way, Gram matrix of each layer's feature map provides a metric of image's overall style. Instead of using Gram matrix, [28] combines traditional MRF model with the style transfer model [11] to synthesize an artistic image, which is more visually consistent with original style image.

2.2 Dig Texture Information in Neural Representations

From [11], we could confirm that, in a specific network for classification such as VGG19 [29], higher layers (such as *conv*4_2) contain almost all object information for input image. In [11] and [28], the synthesis results show that, by using only one higher layer feature map as content loss, the overall structure could be restored perfectly. Thus, the key point lies in texture part. [11] uses Gram matrix as the style measurement. To compute Gram matrix for each layer, the method flattens the feature map along both height and width dimensions, while reserves the channel dimension. From probability theory, we could find that the diagonal terms of Gram matrix are *variance* of particular channels of feature maps, and the off-diagonal terms are *covariance* between different channels. If we consider the feature maps of the output image as random variables, the purpose of minimizing such style loss is to make the output image "like" the style image in an overall view, rather than exactly

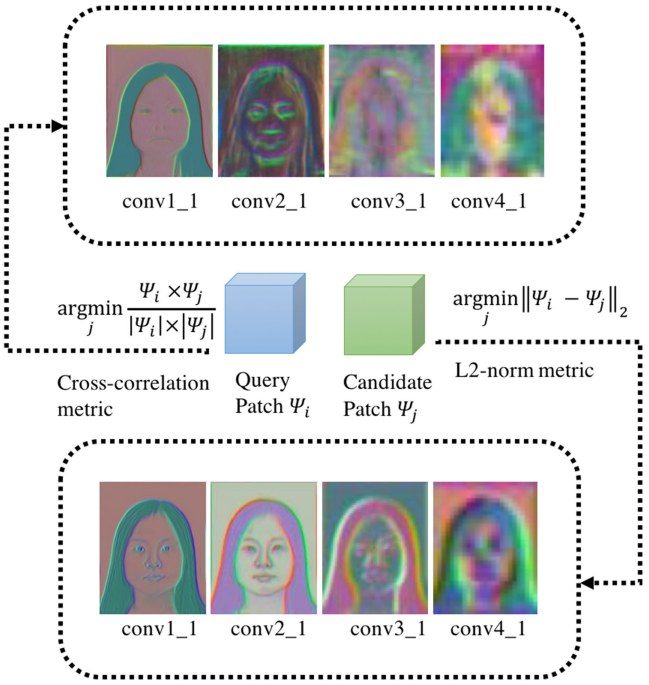


Fig. 2. L2-norm metric and cross-correlation metric comparison in patch matching. Columns of both upper and lower part show target style feature maps extracted from different layers with same input photo. The used network is VGG19 [29], and all feature maps are embedded in three dimensional PCA representation. The size of patches is 3×3. Lower part shows global search with L2-norm. The upper part shows searching results using cross-correlation metric, which is the metric used in [28]. We could find that feature maps selected by L2-norm preserve style information better than those selected by cross-correlation.

imitating the style image. [28] improves such metric by using cross-correlation metric in patch selection. For a specific neural patch, this method first selects nearest patch by normalized cross-correlation. After that, a standard euclidean norm loss are summed over all patches. Compared to [11], the target feature maps are made up of neural patches selected from style image, which makes the synthesis share more similar texture as style image. However, the neural patches in [28] could only be searched within one feature map, which reduces the diversity of texture richness. In our method, neural patches are searched along a training set so that they could reflect sketch textures more accurately.

Since our approach is based on patch matching, with respect to comparing metric between patches, we compare regular L2-norm and normalized cross-correlation used in [28]. In Fig. 2, we could find that normalized cross-correlation is not suitable for a subtle synthesis task like face sketch synthesis. In the upper part of Fig. 2, feature maps generated by cross-correlation metric truly choose some "similar" patches with respect to the input photo, but are much less accurate compared to L2-norm metric. We could see that photo background regions are matched to face regions in sketch dataset, which results in an odd global visual perception. Thus, we choose to use L2-norm metric in our synthesis approach. Another important difference in our method lies in that we abandon the content loss used in [11] and [28]. Content loss is aimed to ensure the output image has general same object shape with the input image. Since our approach uses neural patches to piece together feature maps in dominant neural layers, face structures can be restored via those feature maps. We utilize all training photos in dataset to find the best match, then apply training sketches in dataset for the synthesis of target feature maps. Since the sketch image feature maps contain abundant information from low level to high level after convolution, they are responsible for the final target feature maps generation.

3 Approach Overview

Our overall framework is shown in Fig. 3. As explained before, using neural style transformation into face sketch synthesis directly like [11] does has a fatal drawback: only one sketch in dataset could offer style information, while ample texture information inside whole dataset cannot be used. In Fig. 1, we can see that the result by [11] has an overall sketch style like input style image, but large parts of the face textures are smeared. This is because the Gram matrix used in [11] could not reflex position information for face organs. We could say that using Gram matrix is not suitable for our tasks. In neural representations, position information is fully preserved since convolutional filters do not change the overall space structures. To synthesize a face sketch with detailed texture information, the training sketch database should be fully used. Our main contributions lie in integrating neural representations with patch-based synthesis methods. We search for the best feature map patches for output image in several network layers. We utilize our Enhanced 3D PatchMatch to do the global search in the whole face photo feature maps to find the closest region which resembles input face patches. This is our first stage searching. Then, we search in such *object region* via a cross-layer cost aggregation to find the most closest neural patch still in face photo training space. This is our second stage searching. Finally, we perform patch voting to synthesize target feature maps using neural patches from face sketch training dataset. Through such two stage searching, the nearest feature map patches with texture similarity could be selected. Our approach has achieved high-quality visual effects on CUHK, AR, XM2VTS and CUFSF datasets.

4 Face Sketch Synthesis

4.1 Neural Representation Guided Synthesis

In our overall framework (Fig. 3), assuming that a input target photo image $x_c \in \mathbb{R}^{H \times W}$ is given, and a set of feature maps $M_l(x_c) \in \mathbb{R}^{H_l \times W_l \times C_l}$, $l = \{conv1\_1, \ldots, fc7\}$ for each layer is obtained. We then apply layers $L = \{conv1\_1, conv2\_1, conv3\_1, conv4\_1\}$ as feature map constraint for our final output x. Our goal sketch image is $x^* \in \mathbb{R}^{H \times W}$, and it is optimized as:

$$x^* = \underset{x}{\text{argmin}}\, E_T(x, x_c) + \alpha E_S(x), \tag{1}$$

where, α is a coefficient to control overall style contribution. On right side of Eq. (1), $E_T(x, x_c)$ stands for texture feature map loss, which computes the distance between feature maps of output image and objective feature maps in specific layers L. The target texture feature maps are blended by neural patches searched from dataset according to x_c. $E_S(x)$ is the global style loss which serves as a constraint to make the whole pixel distribution resemble dataset sketches. Our sketch optimization process is similar to the training of a

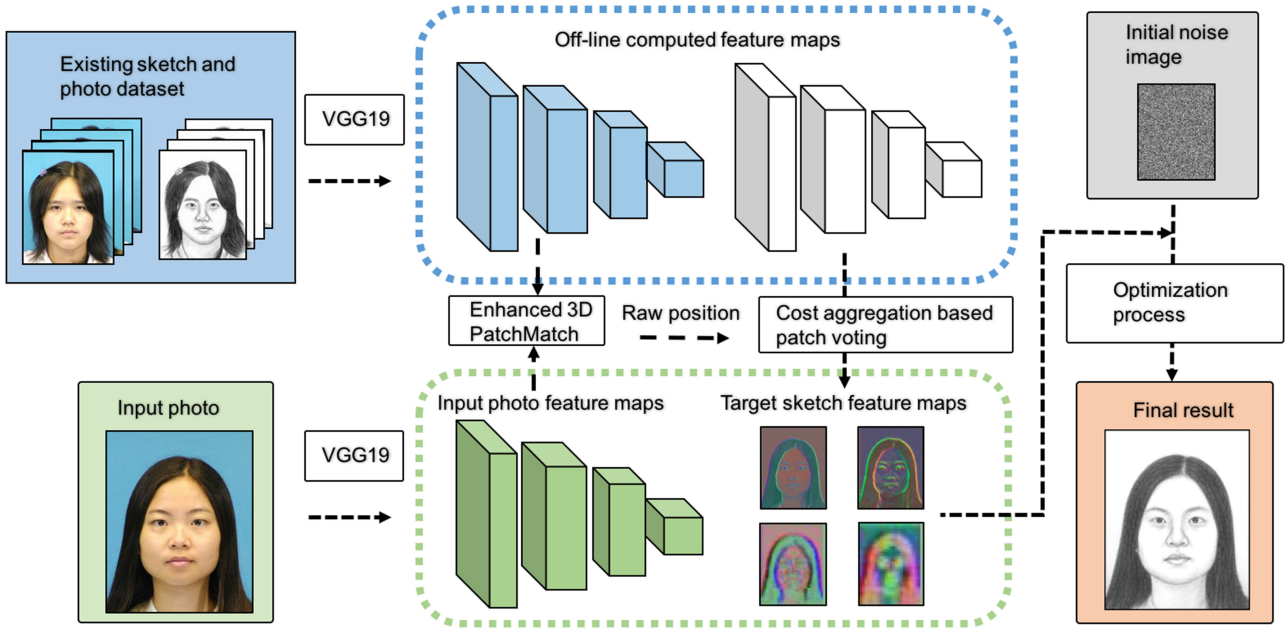


Fig. 3. The overall framework of our neural representation guided face sketch synthesis approach. In the first stage of searching, we first match input photo neural patch with off-line computed photo dataset feature maps using Enhanced 3D PatchMatch. The raw positions are called object regions. We perform a cross-layer cost aggregation in such object regions, and neural patch for each position is selected from training sketch dataset. Patches around central position perform patch voting to generate target feature maps. Finally, a random noise image is fed into the VGG19 network, and gradient descent method is applied to optimize each pixel for the output face sketch.

neural network. The random noise image x is similar to the random initialized weights of a neural network. Eq. (1) gives out the overall optimization target. Once the texture feature maps are computed by $E_T(x, x_c)$ and style feature maps are computed by $E_S(x)$, the final output face sketch x^* can be optimized from a random noise image x.

Removal of Content Loss Contribution. In the optimization object of [11] and [28], *content loss* is added to reconstruct the content of input photo. Suppose that x_c denotes input face photo and x denotes final result. $M_l(x_c)$ and $M_l(x)$ are feature maps for x_c and x in layer l. Content loss is usually the high layer loss between $M_l(x_c)$ and $M_l(x)$, and can be defined as:

$$E_C(x, x_c) = \sum_{l \in L'} \frac{1}{2} \|M_l(x) - M_l(x_c)\|^2, \tag{2}$$

where, $L' = \{conv4\_2\}$. In our case, the background of sketch is pure white, while the background of the face photo is blue curtain. If we bring in the content loss into Eq. (1), the background of the final result will be impure and the tone will totally deviate from the sketch style. Bringing in content loss would also make the final synthesis result resemble more face photo rather than face sketch. A simple comparison between the existence of content loss contribution is shown in Fig. 4. Therefore, we do not use the content loss in our optimization framework.

Target Texture Feature Maps. The texture feature map loss $E_T(x, x_c)$ is defined as follows:

$$E_T(x, x_c) = \sum_{l \in L} \frac{1}{2} \|M_l(x) - \Phi_l(x_c)\|^2, \tag{3}$$

where, target feature map $\Phi_l(x_c)$ has the exact same size and channels with $M_l(x_c)$, but is synthesized from the feature maps of training sketches. Function Φ selects best patches from the training face photo feature maps according to the input photo feature maps, and then uses the corresponding training sketch feature map patches to synthesize the target feature maps, which is the core of our approach. Suppose we have an existing sketch and face photo dataset $\{P_1, S_1\}$, $\{P_2, S_2\}, \ldots, \{P_N, S_N\} \in \{\mathbb{R}^{H \times W}, \mathbb{R}^{H \times W}\}$, and their feature maps are all computed off-line and stored for use. In each

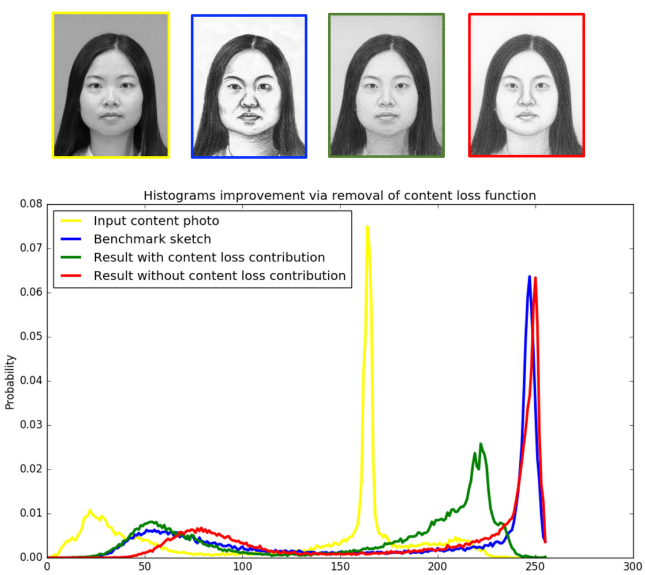


Fig. 4. Comparison of effects via removal of content loss contribution in Eq. (2). The upper row shows the input photo, the benchmark sketch, the synthesized result with content loss and the result without content loss, respectively. The lower diagram shows the histogram of the four images. From the histogram below, we can find that the result with content loss has close tone distribution with gray scaled input photo. By removing such loss, the synthesis result's tone distribution improves much better and has almost the same style with the benchmark.

pair, P stands for a face photo and S stands for the sketch of the photo drawn by artist. For a certain patch size, the feature maps are segmented into m overlapping patches $\phi(k), k = 1, 2, \ldots, m$. For a specified patch in a certain layer l, we need to match the input photo feature map with the training face photo feature maps of $P_1, P_2, \ldots, P_N$, and then use neural patches from sketch feature maps of $S_1, S_2, \ldots, S_N$ to perform patch voting. Such tasks could be converted into a *Nearest Neighbor Field (NNF)* problem. We would discuss our approach in Section 4.2. After we have selected the best match, the target texture feature maps are constructed by *patch voting*, which averages the intensity in one pixel from all patches covering the pixel.

Obviously, matching process along the whole image feature space is rather time-consuming. There are a lot of methods to speed up such process, and PatchMatch [30] is a randomized method for fast nearest neighbor patch matching. The speedup lies in that PatchMatch makes full use of natural image consistency. Neural feature maps conserve the space structure of original image and just extend from one or three channels of natural image to multiple channels. Thus, we revise the original PatchMatch into a third dimension, and introduce our Enhanced 3D PatchMatch approach. We use such approach to search in the first stage. The raw position chose in the first stage is called *object region*. Since we need to construct four level feature maps, the inter-layer consistency is another key point to be considered. By the enlightenment of *Cost Aggregation Algorithm* [38], we utilize cost aggregation to achieve more stable results.

Style Loss Contribution. For each artistic style, a sketch has its own tone distribution. In [39], the authors conclude that in general the tone distribution of a sketch is composed of three tone layers, i.e., dark, mild-tone, and bright layers. Each of them can be generalized by a certain distribution, and the whole sketch tone is the sum of such three layers. In our model, since the output is a sketch rather than a natural image, we could not use the widely used *total variation* defined in [40] as a prior to smooth the results. Using the Gram matrix in [11] to give an overall visual enhancement is our strategy. Noticing that the Gram matrix contains no position information and just serves for a component for overall style, we hence use the mean Gram matrix $\overline{G_l}$ of the whole dataset as such prior:

$$E_S(x) = \frac{1}{4H_l^2 W_l^2 C_l^2} \|M_l'(x)^T M_l'(x) - \overline{G_l}\|^2, \; l \in L, \quad (4)$$

where, $M_l'(x) \in \mathbb{R}^{H_l W_l \times C_l}$ is a reshape form of $M_l(x)$. The mean Gram matrix represents the mean of each Gram matrix computed from the sketch dataset:

$$\overline{G_l} = \frac{1}{N} \sum_{i=1}^{N} M_l'(S_i)^T M_l'(S_i), \quad (5)$$

where, $M_l'(x) \in \mathbb{R}^{H_l W_l \times C_l}$ is also a flattened form of $M_l(x)$. By such style loss, pixel intensity distribution is well constrained and whole pixel tone resembles true sketch nicely.

4.2 Target Feature Maps Acquisition via Enhanced 3D PatchMatch and Cross-Layer Cost Aggregation

Target feature maps $\Phi_l(x_c)$, $l \in L$ in layers $L = \{conv1\_1, conv2\_1, conv3\_1, conv4\_1\}$ serve as the constraint in our optimization process, which significantly decide the synthesis quality of the final results. Our approach is based on two assumptions:

- To preserve simplicity of sketch textures, feature maps in selected layers must be composed of patches from feature maps of sketch dataset images rather than photo dataset images.
- To restore inter-layer consistency, cross-layer cost aggregation around best selected patch should be done.

The simplicity of sketch texture is vital for our style transfer framework. We select patches only from feature maps of sketch dataset images to preserve such simplicity. In [11] and [28], high layer feature maps of input images are used in the optimization processes, which are called content loss and have been mentioned in the above subsections. In our work, feature maps of content image are only used to serve as a reference for patch searching, rather than directly serve as a loss term in the optimization process. For dense-correspondence problem, one of the most efficient method is PatchMatch [30], which is designed to quickly find approximate nearest neighbors. Although feature maps have much larger channel dimension than natural images, yet they still preserve the spatial information of original image, which is vital for our extension to the original PatchMatch method.

In our training set, since there is a pair of photo and sketch, we should use them fully by another assumption:

- The face photo and sketch in a pair describe the same person, thus the same position neural patch from feature maps of face photo and face sketch should be viewed as a nonlinear mapping.

Based on such assumption, we first search patches in face photo feature map space to determine the location, then we use same position neural patches from face sketch feature maps to synthesize the target feature maps.

Enhanced 3D PatchMatch to Find the Best Matching Position. For a target face photo x_c, we first perform PatchMatch between the input face photo feature maps $M_l(x_c)$ and the training face photo feature maps $M_l(P_{n'})$, $n' = 1, 2, \ldots, N$, $l \in L$. For a certain position $k \in \mathbb{R}^2$ in $M_l(x_c)$, we could define a *Nearest Neighbor Field* $NNF^l(k)$ after our Enhanced 3D PatchMatch search. Here, the $NNF^l(k)$ is a three dimensional tuple (n', i'), in which $n' \in \mathbb{R}$ defines which training face photo feature maps, and $i' \in \mathbb{R}^2$ defines the position. To compare the neural patch distance, we define the following metric:

$$C^l_{P_{n'}}(i', k) = \frac{\|\phi_{M_l(P_{n'})}(i') - \phi_{M_l(x_c)}(k)\|}{\|M_l(x_c)\|}, \; l \in L, \quad (6)$$

where, $\phi_{M_l(P_{n'})}(i')$ and $\phi_{M_l(x_c)}(k)$ are neural patches centering pixel i' and k, respectively. $\|M_l(x_c)\|$ is an invariant for specific target face photo, and can be used as a normalization term. The cost defined in Eq. (6) is used not only as metric in Enhanced 3D PatchMatch but also for cross-layer cost

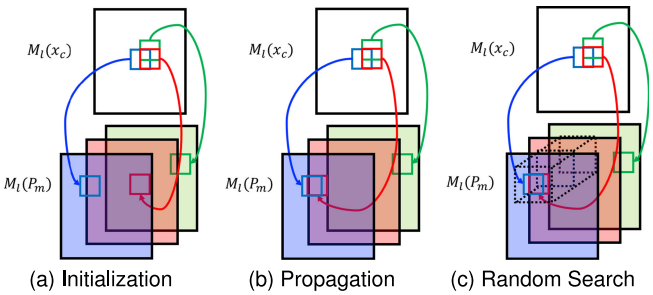


Fig. 5. Illustration of our Enhanced 3D PatchMatch. In the first initialization stage, we allocate random values to the whole Nearest Neighbor Field (NNF). Then in the propagation stage, with a line-scanning way, the NNF of position labeled by red square in this image is propagated by its left and upper position. The third step is the random search, in which the NNF of position for red square is optimized within a three dimensional cube centering at its current position.

comparison. The Nearest Neighbor Field is a function NNF^l: $k \rightarrow (n', i')$ that records the best match patch position for k through the whole (n', i'). Suppose n indicates which face photo feature map in dataset is selected, and i indicates which patch in such face photo feature map is selected, then the best patch $\phi_{M_l(P_n)}(i)$ can be selected as:

$$(n, i) = \underset{(n', i')}{\operatorname{argmin}} C^l_{P_{n'}}(i', k), \tag{7}$$

where, the (n, i) is the optimum value of (n', i'). To search for the Nearest Neighbor Field of each position, we adopt the so called *Enhanced 3D PatchMatch* algorithm, which is described in Algorithm 1. Traditional PatchMatch defined in [30] matches patches in a 2D plane and makes full use of the space consistency of natural image. In our tasks, an additional dimension n' is introduced. To maintain the intrinsic principle of original PatchMatch algorithm, we revise the original third step of random search into different training face photo feature maps, which is the third dimension besides the two space dimensions.

Fig. 5 shows our Enhanced 3D PatchMatch. Around the patch $\phi_{M_l(P_n)}(i)$, the neighborhood $Nei(n, i)$ is called *object region*, which is considered as the most matching area for k in $M_l(P_{n'})$. But we all know that the best match in feature maps of face photo dataset does not mean it is the best match in feature maps of corresponding sketch dataset. With above considerations, we search in $Nei(n, i)$ of P_n to find the best match of sketch neural patches. We have collected statistical information on the matched patches of all the tested input face photos in different layers for the CUHK and AR datasets. Table 1 shows the percentage of patches matched in different layers. We can see from Table 1 that the neuro feature maps at layers $conv2\_1$ and $conv3\_1$ offers better matching results. The lower layer $conv1\_1$ seems to have lower performance compared to the layers $conv2\_1$ and $conv3\_1$ as it is too sensitive to visual variations. Although the upper layer $conv4\_1$ has low performance in matching, yet it still could establish certain level of correspondence (around 15 percent matched patches) between query patches and candidate patches. Since the top layer $conv5\_1$ (resolution: 13×16) loses too much discriminative information, we do not apply this layer in our approach. Our findings are also in accordance with [28], [41], which show that middle layers are more suitable for recognition purposes as they offer better discriminative performance while being more robust to noises. Experiments shows that our Enhanced 3D PatchMatch could drastically speed up matching process compared to traditional PatchMatch [30] and brutal force search. The comparison results are shown in Fig. 6. All the four methods are implemented in naive Python code without any parallel schemes. From Fig. 6, we could find that our Enhanced 3D PatchMatch acquires fastest speed than the other searching schemes and makes a good balance between synthesis speed and effects.

TABLE 1
Percentage of Patches Matched in Different Layers

Layers	$conv1\_1$	$conv2\_1$	$conv3\_1$	$conv4\_1$
Resolution	200×250	100×125	50×63	25×32
CUHK (%)	26.3%	30.4%	28.7%	14.6%
AR (%)	25.6%	29.5%	29.3%	15.6%

Algorithm 1. Enhanced 3D PatchMatch to Select Best Neural Patch

Input: Target face photo x_c, training photo feature maps $M_l(P_{n'}), n' = 1, 2, \ldots, N, l \in L$
Output: Nearest Neighbor Field NNF^l: $k \rightarrow (n, i)$

for $l \in L$ **do**
  Initialization:
  for position k in $M_l(x_c)$ **do**
    Allocate $NNF^l(k_x, k_y)$ with random value (n', i'_x, i'_y), $n' \in [1, N]$, $i'_x \in [1, W^l]$, $i'_y \in [1, H^l]$;
  end for
  for $iteration \in [1, 5]$ **do**
    for position k in $M_l(x_c)$ **do**
      Propagation:
      Suppose that $(t, q_x, q_y) = NNF^l(k_x - 1, k_y)$;
      if $C^l_{P_t}((q_x + 1, q_y), k) < C^l_{P_{n'}}((i'_x, i'_y), k)$ **then**
        $n' = t, i'_x = q_x + 1, i'_y = q_y$;
      end if
      Suppose that $(t, q_x, q_y) = NNF^l(k_x, k_y - 1)$;
      if $C^l_{P_t}((q_x, q_y + 1), k) < C^l_{P_{n'}}((i'_x, i'_y), k)$ **then**
        $n' = t, i'_x = q_x, i'_y = q_y + 1$;
      end if
      Random Search:
      $radius = min(H^l, W^l)$;
      while $radius > 0$ **do**
        Allocate (t, q_x, q_y) with random value, $t \in [1, N]$, $q_x \in [1, radius]$, $q_y \in [1, radius]$;
        if $C^l_{P_t}((q_x, q_y), k) < C^l_{P_{n'}}((i'_x, i'_y), k)$ **then**
          $n' = t, i'_x = q_x, i'_y = q_y$;
        end if
        $radius = radius/2$;
      end while
    end for
  end for
end for

Cross-Layer Cost Aggregation in Object Region. Our cross-layer cost aggregation is visualized and shown in Fig. 7. More details are explained in Algorithm 2. Unlike the photo feature map, sketch feature map has different encoding method with input photo due to the style difference, especially in the lower layers. To find the best match, we use the

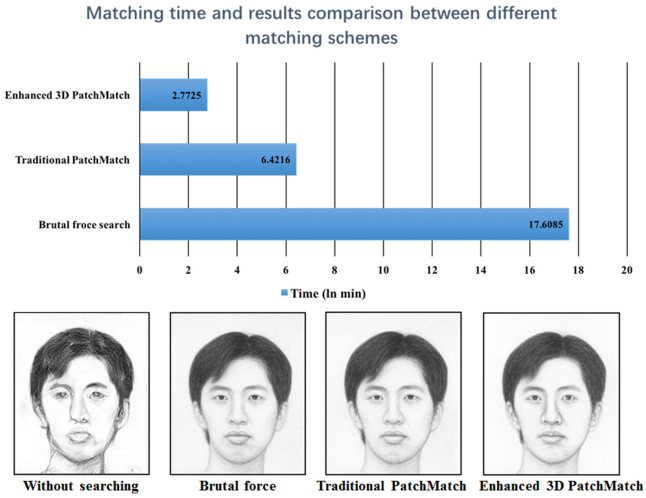


Fig. 6. Matching time and results comparison between different searching schemes. All four methods are implemented in naive Python code without any parallel schemes. The above diagram shows the matching time of our Enhanced 3D PatchMatch, traditional PatchMatch [30], and brutal force search. Note that the x-axis of the diagram used natural logarithm of minutes (ln min) to rescale the ratio of different methods. The lower four sub-images show synthesized examples of a same person. We could find that: without searching process (whole training sketch patches used) cannot compare the matching time, and the results are inaccurate; PatchMatch and brutal force search did improve some local details than Enhanced 3D PatchMatch, but the two methods consume too much time and are unpractical for quick sketch generation tasks.

cross-layer cost aggregation method used in [38] to consider the consistency of different layers. In Eq. (6), we have defined neural patch cost $C^l_{P_{n'}}(i',k)$ in layer l for position k, we can extend it to different layers. Our cross-layer cost aggregation is based on the assumption: averaged cost computed from current and upper layer would make the selected patch more accurately. Suppose that the vector

$$a = [C^{conv1\_1}_{P_n}(i,k), C^{conv2\_1}_{P_n}(i,k), C^{conv3\_1}_{P_n}(i,k), C^{conv4\_1}_{P_n}(i,k)]^T, \tag{8}$$

denotes the original normalized cost at each layer in position i. Note that i is exactly the center of the object region in the first searching stage. Then, we add a generalized Tikhonov regularizer in the following optimization objective, which is aimed to average cost between the current and upper layers:

$$\hat{a} = \underset{\{z^l\}_{l\in L}}{\operatorname{argmin}} \left(\sum_{l\in L} \|z^l - C^l_{P_n}(i,k)\|^2 + \lambda \sum_{l\in L} \|z^l - z^{l_{upper}}\|^2 \right), \tag{9}$$

where, l_{upper} is the upper layer of layer l. For layer $conv1\_1$, l_{upper} denotes $conv2\_1$. λ is a coefficient to control consistency between layers, the larger λ is, the more consistency between the two layers is considered in cost aggregation. Eq. (9) is a convex optimization and has analytical solution. Suppose that O is the objective function in Eq. (9), we set the partial derivative $\frac{\partial O}{\partial z^l} = 0$ to derive:

$$(1+2\lambda)z^l - \lambda z^{l_{upper}} - \lambda z^{l_{lower}} = C^l_{P_n}(i,k), l \in L, \tag{10}$$

where, $z^{l_{lower}}$ stands for the lower layer of l, e.g., the lower layer of $conv2\_1$ is $conv1\_1$. From Eq. (10), we can get similar equations for $l = conv1\_1$ to $l = conv4\_1$. We hence have 4 linear equations in total, which can be expressed clearly as: $S\hat{a} = a$. In our case, the aggregated cost vector $\hat{a} = [\hat{C}^{conv1\_1}_{P_n}(i,k), \hat{C}^{conv2\_1}_{P_n}(i,k), \hat{C}^{conv3\_1}_{P_n}(i,k), \hat{C}^{conv4\_1}_{P_n}(i,k)]^T$, and the matrix S is a 4×4 tridiagonal constant matrix, which could be easily derived from Eq. (10). S is tridiagonal, the inverse S^{-1} exists. Thus, we could solve $\hat{a}$ as:

$$\hat{a} = S^{-1}a, \tag{11}$$

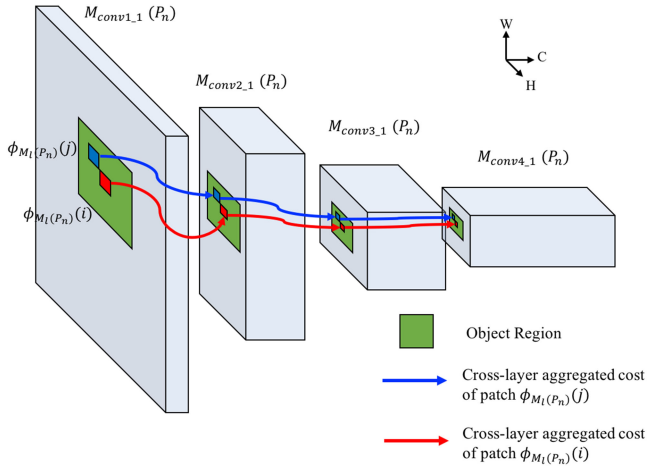


Fig. 7. Visualization of our cross-layer cost aggregation approach. Patch $\phi_{M_l(P_n)}(i)$ is selected by Enhanced 3D PatchMatch and denotes the patch in location i of nth candidate feature map in the face photo dataset. The green region denotes the object region, where we do cross-layer cost aggregation for each pixel around the neighborhood of patch $\phi_{M_l(P_n)}(i)$. After that, the patch with minimum aggregated cost is selected for patch voting of target feature maps in the layer.

Algorithm 2. Feature Maps Acquisition via Cross-Layer Cost Aggregation in Object Region

Input: Target face photo x_c, training photo feature maps and sketch feature maps $\{M_l(P_{n'}), M_l(S_{n'})\}$, $n' = 1, 2, \ldots, N$, $l \in L$, $NNF^l: k \rightarrow (n,i)$
Output: Target texture feature maps $\Phi_l(x_c)$, $l \in L$

```
for l ∈ L do
  for position k in M_l(x_c) do
    Select the nearest patch φ_{M_l(P_n)}(i) according to the input Nearest Neighbor Field NNF^l(k);
    for j ∈ Nei(n,i) do
      Compute aggregated cost â using Eq. (11);
      Select the best j with minimum aggregated cost Ĉ^l_{P_n}(j,k), training sketch neural patch φ_{M_l(S_n)}(j_best) is selected for position k and used finally for patch voting;
    end for
  end for
  for position k in M_l(x_c) do
    Do patch voting to get Φ_l(x_c): average all selected neural patches φ_{M_l(S_n)}(j_best) arounding location k;
  end for
end for
```

In the object region around central pixel i, the cross-layer cost aggregation is done for each pixel j in such region, we then select the best j according to their aggregated cost for the patch voting:

$$j_{best} = \underset{j\in Nei(n,i)}{\operatorname{argmin}} \hat{C}^l_{P_n}(j,k), \tag{12}$$

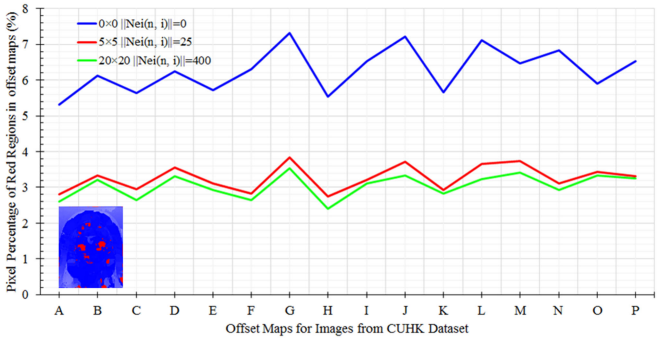


Fig. 8. Effects of cross-layer cost aggregation. The down-left sub-figure shows an example of $conv1\_1$ offset map defined in Eq. (13) for a certain input face. Note that, when the distance between selected neural patch and neural patch of benchmark sketch feature map in the position exceeds a certain threshold, the position is marked as red. We could see that the pixel percentage of red regions decreases with cross-layer aggregation being added (red and green lines). Results without cross-layer aggregation (0×0) have worse effects than those with it. But bigger searching size (20×20 compared to 5×5) has no obvious improvement.

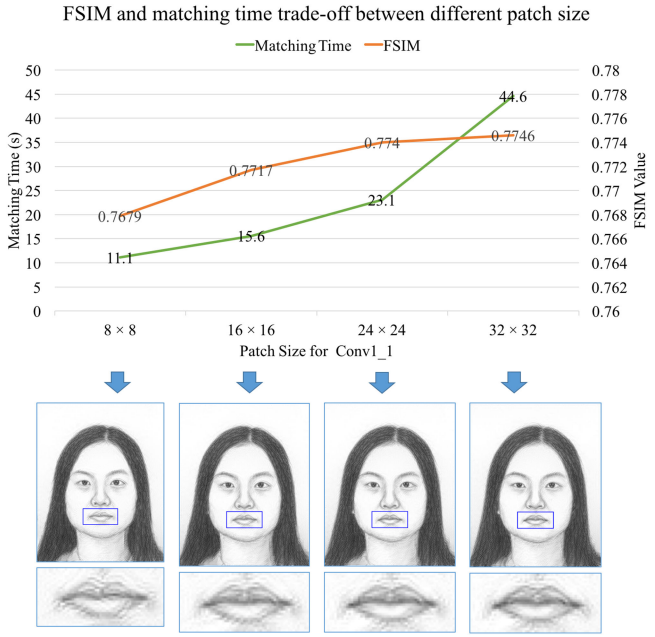


Fig. 9. FSIM value and matching time trade-off between different patch size. Note that here the patch size denotes the patch size of layer $conv1\_1$. The upper diagram shows the increment of the FSIM value and matching time with respect to different patch size. The lower part shows the detailed synthesis result with respect to different patch size. Since the FSIM value increases slower when patch size approaches 24×24, we choose 24×24 as our final configuration.

After we have selected j_{best} for each position for target feature map $\Phi_l(x_c)$, we use the *training sketch feature maps* to do patch voting. Note that all the above selecting algorithms are based on *training photo feature maps*. But since we assume the two kinds of feature maps have one-to-one correspondence, we use neural patches in position j_{best} from $M_l(S_n)$ to synthesize our final target feature maps.

4.3 Implementation Details

Apparently, the optimization method plays a huge role in the synthesis speed. We set the whole iteration number for a single input photo as 512 so as to make sure the output converges to a visually acceptable state. The optimization method used here is L-BFGS-B [42]. α in Eq. (1) is set to 100 so as to make the overall style of synthesis result resemble true sketch. λ in Eqs. (9) and (10) is set to 0.3.

4.4 Algorithm Verification and Parameter Setting

Effects of Cross-Layer Cost Aggregation. Here, we display a simple experiment to show the effects of our cross-layer cost aggregation (see Fig. 8). $Nei(n, i)$ is the set of neighborhood positions in object region. $|Nei(n, i)|$ is the cardinality of set $Nei(n, i)$. We change the size of $Nei(n, i)$ to get final neural patch $\phi_{M_l(S_n)}(j_{best})$ for position k. Suppose that $j_{best} - k = (off_x, off_y)$ is the offset from k to j_{best}. Then, we define the following offset distance as:

$$d = \sqrt{off_x^2 + off_y^2}. \tag{13}$$

In our offset map, the larger offset distance is, the lighter the blue component is for specific position (see the down-left sub-figure in Fig. 8). We compare the distance between selected neural patch and benchmark sketch neural patch in this position. If the distance exceeds a certain threshold, the position is marked with pure red. In Fig. 8, we can see clearly that, when cross-layer cost aggregation is used, the selected neural patch resembles benchmark sketch neural patch more. But using a larger size of object region has no obvious improvement. The reason lies in that our Enhanced 3D PatchMatch has already selected a relevantly accurate neural patch. The second stage searching around object region could be viewed as a kind of fine-tuning. In this way, larger size of object region could not improve synthesis results too much, however, it would be a waste of synthesis time. Thus, we set the window size of such regions to 5×5.

Patch Size in Our Enhanced 3D PatchMatch Algorithm. In our sketch synthesis framework, the most import parameter is the patch size for layers in L. Here, we perform a simple experiment to show the reason of patch size selection. We denote the sketch drawn by artist as benchmark sketches. To evaluate the effects of different patch size, we compute the FSIM value [43] between benchmark sketch and synthesized sketch. Compared to SSIM [44], FSIM emphasizes more on local structure details and is more qualified for the assessment between benchmark sketch and synthesized sketch. From Fig. 9, we can see clearly that, with a larger patch size, the FSIM value between the benchmark sketch and the synthesized sketch is closer. Considering that using a larger patch size also means larger computation load, we balance between speed and quality and choose the patch size for layer $conv1\_1$ as 24×24. We set the patch size for layers in L as conv1_1: 24×24, conv2_1: 12×12, conv3_1: 6×6, conv4_1: 3×3. Apparently, in our cross-layer cost aggregation, the channels differ from layer to layer. And by above patch size setting, the raw cost defined in Eq. (6) for different layers has the same ratio relative to the whole feature map: (conv1_1 : conv2_1 : conv3_1 : conv4_1) = $(\frac{24\times24}{W\times H} : \frac{12\times12}{\frac{1}{2}W\times\frac{1}{2}H} : \frac{6\times6}{\frac{1}{4}W\times\frac{1}{4}H} : \frac{3\times3}{\frac{1}{8}W\times\frac{1}{8}H})$= (1 : 1 : 1 : 1). This setting makes the whole cost aggregation available.

Optimization Strategies Comparison. The optimization strategies affect the synthesis speed. We compare the total loss decedent with respect to iterations for different

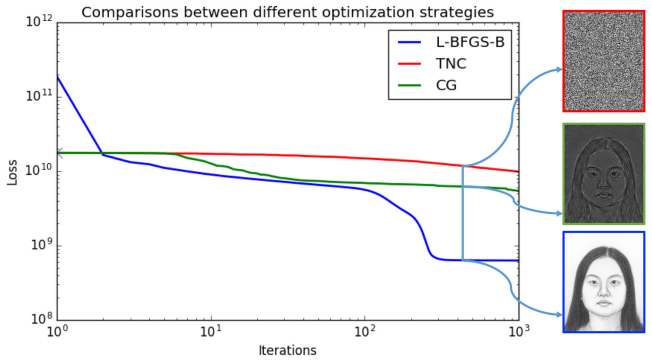


Fig. 10. Loss descendant comparisons between different optimization strategies. Target feature maps are fixed and an initial noise image are fed into VGG19 network. The diagram gives the overall loss defined in Eq. (1) according to different iterations. The right part of this diagram gives midterm synthesis results after same iterations given different optimization strategy. We could find that L-BFGS-B [42] prevails CG [45] and TNC [45], which is the reason why we choose such method.

strategies. The result is shown in Fig. 10. Given same target feature maps in the four main layers and an initial noise image, we take down the overall loss of Eq. (1) with increment of iterations. From Fig. 10, we could see that the CG method [45] and L-BFGS-B method [42] have obvious convergence processes, while the TNC method [45] cannot converge within 1000 iterations. Based on the experiments, we choose L-BFGS-B as our optimization strategy.

5 Experimental Results

5.1 Experiments Configuration

The experiments of our proposed approach are executed mainly on two face datasets: CUHK Face Sketch Databaseand AR Database. We set two kinds of comparison, which are qualitative and quantitative verifications. For the inter-database verification of CUHK database we use the default 88 faces for training and extract their feature maps off-line. The rest 100 faces are used for testing. The AR database has 123 pairs of face and sketch, and we take the leave-one-out strategy to test our synthesis results. Since the benchmark sketches from the CUHK Face Sketch Database are cropped into 250×200, we use this size for all verification.

Methods Chosen for Comparison For face sketch synthesis tasks, the state-of-the-art researches are mainly in three frameworks, based on Markov Random Field [17], Markov Weight Field [18], and Spatial Sketch Denoising [19]. MRF-based sketch synthesis uses a multi-scale graph model and converts synthesis task into solving a Markov Random Field. This work enlightens several following researches, including MWF method. The MWF method synthesizes a patch via k patches selected from dataset and optimize the weights for the k patches. The SSD-based method uses the same k patches to synthesize a patch, but uses a novel sketch denoising method to smooth results. In general, MRF-based method has largest noise due to the direct use of patches. The SSD-based method has lowest noise but loses the sharp edges of pencil stroke, which makes results deviate from sketch style. The MWF-based method maintains a balance between the other two methods and achieves effective results for different samples.

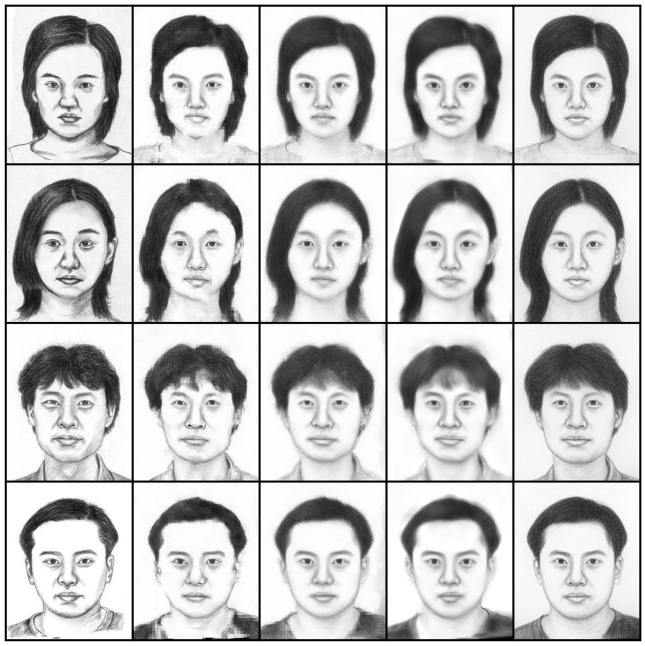


Fig. 11. Face sketch synthesis results using different methods on the CUHK face sketch database. The first column shows benchmark sketches, which is drawn by artist according to face photos. The second column shows synthesis results by MRF [17]. The third column shows results by MWF [18]. Fourth column shows results by SSD [19]. The last column is the results produced by our approach.

5.2 Qualitative Comparisons

5.2.1 *Synthesis Results of CUHK Database*

Fig. 11 shows the comparison for CUHK database. The four columns are random synthesis samples on CUHK face test dataset. From the first column, we can conclude that the artist are inclined to use hard strokes to draw chins in order to highlight the outline of the face. In the neck regions around face, the artist likes to use shadows to express light relationship. From the comparisons, we find that our results have the sharpest chins and best shadow effects around chins, which gives them stereo perceptions. The MRF-based results are the noisiest ones because such method only uses small portions of patch to do belief propagation and segments boundaries between patches by graph cut. The segmentation could not confirm smooth boundaries. In the fourth row second column, the collar regions of the face contains obvious inconsistency between patches. MWF-based method improves such inconsistency between patches greatly by using several patches instead of just one patch to do inference. Through different weights, such method could create patches that even do not exist in dataset. In general, the MWF-based method is visually superior to the MRF-based method, but patch blending to a certain extent also brings the problem of blurring. The SSD-based method has the most blurred synthesis. Lots of detailed information loses, and the whole sketch cannot present pencil-like stroke as artist does in the first row. Compared to these methods, our approach achieves a balance between detailed pencil stroke and moderate blurring. In the third subject of the comparison, the forehead hair is sparse and its color merges with forehead skin. The other three methods could only simulate such color merging by blurring (SSD) or inconsistency (MRF and MWF). Our result

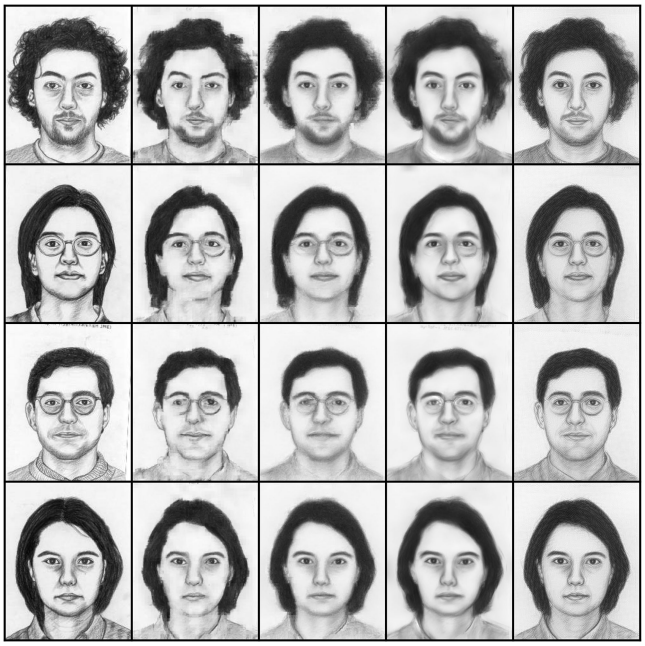


Fig. 12. Face sketch synthesis results using different methods on the AR face sketch database. The first column shows benchmark sketches, which is drawn by artist according to face photos. The second column shows synthesis results by MRF [17]. The third column shows results by MWF [18]. The fourth column shows results by SSD [19]. The last column is the synthesis results by our approach.

perfectly restores benchmark sketch's style. Shadow effects are also our advantages. In noses and chins, our results show the most obvious shadow effects compared to other three methods. Our results' face contours are also the most full and sharp, which give best pencil-like visual conceptions.

5.2.2 Synthesis Results of AR Database

The AR database has more complicated face structures than CUHK database, which could be used to test the robustness of our approach. We choose four typical faces for the test, two of them wear glasses. As mentioned before, we adopt the leave-one-out strategy for the AR faces comparison, which is using 122 face-sketch pairs of the whole 123 pairs as training data except the one to be synthesized. Fig. 12 shows the results comparison between MRF [17], MWF [18], SSD [19], and our method. In the second, third rows of Fig. 12, face regions with glasses are hard to process. We can see clearly that eyes and glass borders are inconsistent or mixed together. Due to global search processes, our approach is able to distinguish eyes and glass borders perfectly and maintains sharp edges for round shape of glasses. From Figs. 11 and 12, we could see clearly that our approach outperforms other methods in different databases and preserves face details best among all existing methods.

5.2.3 Detailed Synthesis Examples Citing

Fig. 13 gives some detailed synthesis examples to prove the advantages of our face sketch synthesis. First subject of Fig. 13 is a girl with big eyes and obvious double-fold eyelids. From the comparison of enlarged view of the right eye, we could see that only our synthesis gives clear outline of this double-fold eyelids. But all the four methods have a common drawback that the actual eye shape is rounder than any of synthesis sketch eyes. However, our approach still could not exactly restore the shape of small face components and this would be reserved for further researches. The second row first column shows a man with red collar, and we select a region with junction of neck, collar and background. From the enlarged view, we could see clearly that our approach has the best detailed expression in junction of different components. All the other three methods could not express such junction and merge all edges together. The third subject is a man and the magnification region is his mouth. Since the colors of the skin and mouth are close, the former three methods have inconsistency around edges of the mouth. Last control group shows that our approach has a certain ability to handle regions with light reflections. The forth subject wears glasses and his right eye hides behind the glass with reflections. All the former three methods could not restore the eye but just mix it with the glass. Our approach perfectly expresses the eye despite the light reflections of glasses, which shows the robustness.

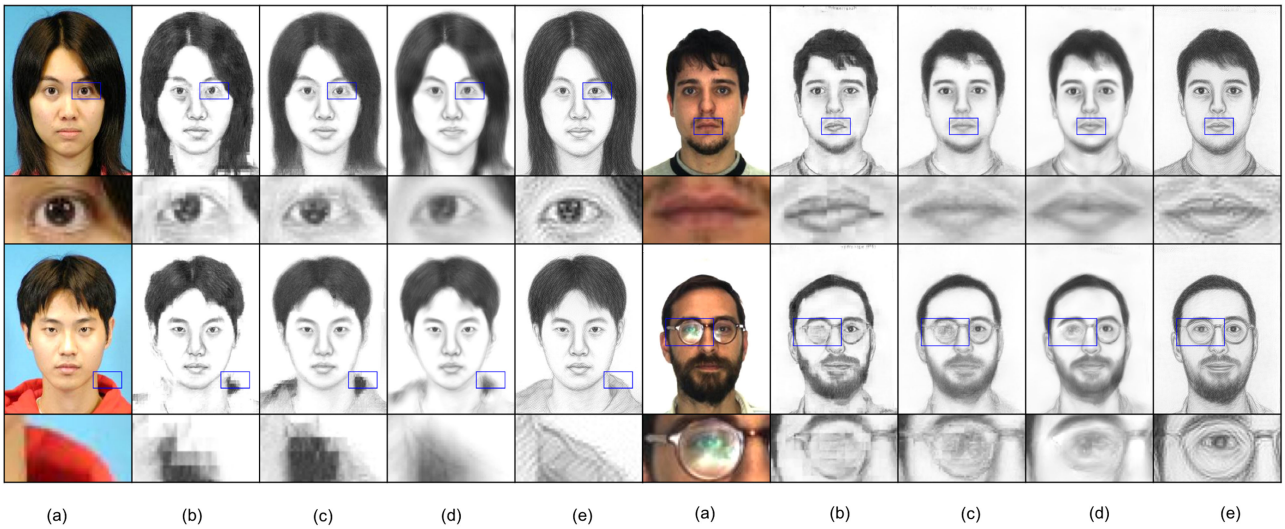


Fig. 13. Qualitative evaluation on different methods with local detail magnification. (a) Input face photos. (b) Synthesis results by MRF [17]. (c) Synthesis results by MWF [18]. (d) Synthesis results by SSD [19]. (e) Our synthesis results. This picture is better for watch with color version.

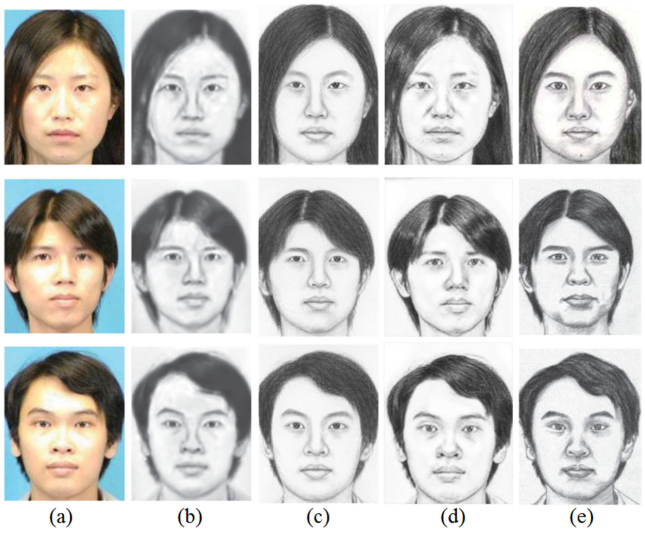


Fig. 14. Comparisons between our proposed method and two different CNN-based generative models. (a) Input face photo. (b) Synthesis results by [27]. (c) Synthesis result by our approach. (d) Synthesis results by [46]. (e) Benchmark sketches drawn by artist.

5.2.4 Comparison with CNN-Based Models

In [27], a fully convolutional network is trained for photo-sketch mapping, which is composed of six convolutional layers, and small size filters (3×3 and 1×1) are defined for each layer. The comparison between this work and ours is shown in Fig. 14. There are two main shortcomings in this method. One is the lack of training pairs. The CUHK face sketch training set has 88 training pairs, which is inadequate for such a complicated photo-sketch mapping. From Fig. 14, we could find that the result of [27] seems like a gray version of original color face image and has little sketch attribute, which indicates that the trained network still cannot capture core differences between photo and sketch. Another shortcoming lies in that the synthesis quality in [27] is dependent on input image size. CNN convolves input image in each layer and serves as a kind of smooth filtering. The value of each pixel is influenced by large region of image's other parts. This explains why the result of [27] is blurred too much. Our approach also can accurately restore sketch strokes and avoid general blurring effects.

Recently, a lot generative adversarial network (GAN) based models [46], [47] has been released. The methods could map a set of images directly into another category of images by training a discriminator and a generator at the same time. We also make comparison with CycleGAN [46] in Fig. 14. We use the 88 training pairs of CUFS to train a CycleGAN model with 400 epochs, then the generator is used to generate face sketch from a probe photo. From Fig. 14d, we may say that the state-of-the-art GAN-based model did quite well in shape reconstruction. Compared to Fig. 14b, results of Fig. 14d solve the blurring effect and have sharp edges. However, our method in Fig. 14c still prevails GAN-based method in two points: (1) our results have more pencil-like style than the results by CycleGAN [46]. Our method does not map original photo pixel-wisely into a sketch and thus is closer to human-like pencil drawing art; (2) In some complex small face area, GAN-based model still has lots of noise, while our method could overcome that. In the nose area of sketches in Fig. 14d, large parts of noise pixels exist, while in our results there are no such drawback.

TABLE 2
SSIM and FSIM Evaluation between Different Methods

Database	Metric	MRF [17]	MWF [18]	SSD [19]	Ours
CUHK	SSIM	0.6951	0.7132	0.7167	0.6939
	FSIM	0.7236	0.7329	0.7169	0.7345
AR	SSIM	0.7105	0.7404	0.7407	0.7108
	FSIM	0.7496	0.7579	0.7441	0.7932

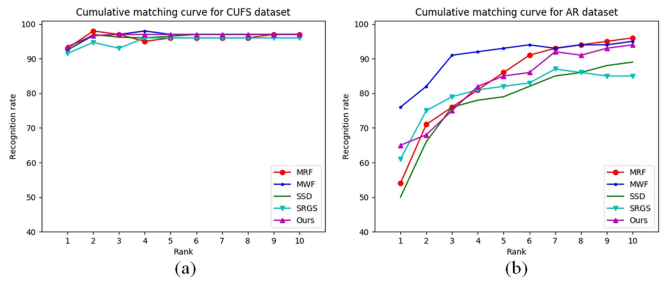


Fig. 15. Cumulative curve comparisons on different datasets. We compared cumulative curve for PCA recognition method [49] between results from five state-of-the-art synthesis methods. The five methods are MRF [17], MWF [18], SSD [19], SGRS [48], and ours. (a) The cumulative curve for CUFS dataset. (b) The cumulative curve for AR dataset.

5.3 Quantitative Comparisons

5.3.1 SSIM and FSIM Evaluation

Table 2 gives the quantitative comparisons of different methods. We use SSIM (Structural Similarity) and FSIM (Feature Similarity) [43] to measure our approach. We compute the whole mean SSIM and FSIM between benchmark sketches and synthesis results via different methods. For SSIM computation, we use default configuration of image processing library scikit-image. For FSIM, we use the default code from author's page [43]. In Table 2, we find that our approach has no advantage in SSIM scores, but has advantages in FSIM scores. SSIM metric is based on the statistical information of images. It measures the overall structure difference of two images. Since the MRF [17], MWF [18] and SSD [19] methods directly use patches from training set, their overall structure information may be better preserved. Compared to SSIM, FSIM also considers phase consistency in Fourier components and gradient magnitudes under different operators. Generally, FSIM may measure the regional structural similarity such as contours and hair better than SSIM. In actual conceptions, the MWF results did appears better than MRF and SSD. And our results outperform the MRF, MWF, and SSD based methods, where our approach preserves regional strokes information much better.

5.3.2 PCA-Based Recognition Rate Evaluation

To further validate our synthesis results quantitatively, we use the same recognition rate method used in [48] to compare the cumulative curves of different methods. Like [48], we use PCA recognition method[49] to evaluate whether a synthesized sketch similar to hand-drawn sketch. The results are shown in Fig. 15. It can be seen clearly from

TABLE 3
Quantitative User Study

Sketch group (No.)	Methods	Short term recognition speed (s)	Short term retention (%)	Long term recognition speed (s)	Long term retention (%)	Attractiveness
1	MWF [18]	1.8	52	3.5	44	Medium
	SSD [19]	2.3	46	3.9	39	Medium
	Ours	1.6	83	2.2	62	High
2	MWF [18]	1.9	55	3.3	43	Medium
	SSD [19]	2.4	31	4.2	40	Medium
	Ours	1.8	67	2.5	66	High
3	MWF [18]	2.0	42	3.8	36	Medium
	SSD [19]	2.2	41	4.1	33	Medium
	Ours	1.9	69	2.4	59	High

Fig. 15 that our proposed method has high-quality performance compared to the state-of-the-art methods.

5.3.3 User Study Evaluation I

First, we design two quantitative variables to evaluate the performance of synthesis results. The first is the recognition speed (s) for the users. For each query photo, the users are presented with five different face sketches then we record the time used for users to identify exactly the person of the photo. The second variable is the retention percentage of users. After a certain time, the users are presented those sketches again to test if they could remember those identities. We measure the mean value of these two variables. We also ask users which of the following three words best describe the sketch presented before them for attractiveness: "Low", "Medium" and "High". The above two variables (recognition speed, and retention) are measured in a short term and a long term. Short term means the first time we conduct such experiments. Long term means the same experiment conducted ten days later. In normal sense, long term test is more challenge than short term test. We conduct three sets of such test sketches and each set contains five different sketches. The details are shown in Table 3. From Table 3, we could find that our results outperform MWF and SSD methods in both recognition speed and retention no matter in short term or in long term. Our results also achieve highest score in "Attractiveness" in user study.

5.3.4 User Study Evaluation II

To evaluate the quality of our synthesis sketch subjectively, we also conduct another user study. We invite 50 users, half of which have artistic background, and the rest half do not. There are two sets of questions. The first set is to evaluate how much the synthesized sketches reflect the face structure of original face photo. The other set of questions is to find out how much the synthesized sketches resemble the stylistic style of sketch drawn by artist. As previously demonstrated, the MRF-based synthesized sketch has noisier edges and worse visual conception than the MWF and SSD based synthesized sketch, thus we choose only MWF and SSD as methods for comparison. We invite 50 volunteers to join in our user study. Each participant is asked to grade the structure similarity and stylistic similarity of six sampled sketches using 9-point scale (with 9 points being the highest level). The results are shown in Tables 4 and 5.

From the statistics of Tables 4 and 5, we could see that our synthesis sketch has the highest scores in both structures similarity and stylistic similarity. In other words, our sketch preserves the best face structures compared to the original photo and presents best sketch style. The MWF-based sketch has overall lower score than the SSD-based sketch, and it makes sense since SSD preserves face structure better. Note that our approach also has the smallest standard deviation, which means the study participants have least divergence for the synthesis quality of our synthesis results.

5.4 Extending Synthesis Results

Since our method's key advantages lie in that we utilize neural networks to extract features, robustness would be maintained and our framework is adapted for face photos

TABLE 4
Synthesis Performance Study with Artistic Background

Survey item	Methods	μ	σ	95% confidence interval	
				Lower Bound	Upper Bound
Structure	MWF [18]	5.48	1.92	5.22	5.76
	SSD [19]	5.44	1.65	6.21	6.67
	Ours	7.81	0.83	7.69	7.92
Stylistic	MWF [18]	4.91	1.33	4.72	5.10
	SSD [19]	4.81	1.23	4.63	4.98
	Ours	7.98	1.11	7.82	8.13

TABLE 5
Synthesis Performance Study without Artistic Background

Survey item	Methods	μ	σ	95% confidence interval	
				Lower Bound	Upper Bound
Structure	MWF [18]	5.81	2.15	5.57	6.06
	SSD [19]	5.92	1.94	5.70	6.14
	Ours	6.81	1.83	6.60	7.02
Stylistic	MWF [18]	5.23	2.05	5.00	5.46
	SSD [19]	5.61	1.95	5.39	5.83
	Ours	6.92	1.89	6.71	7.14

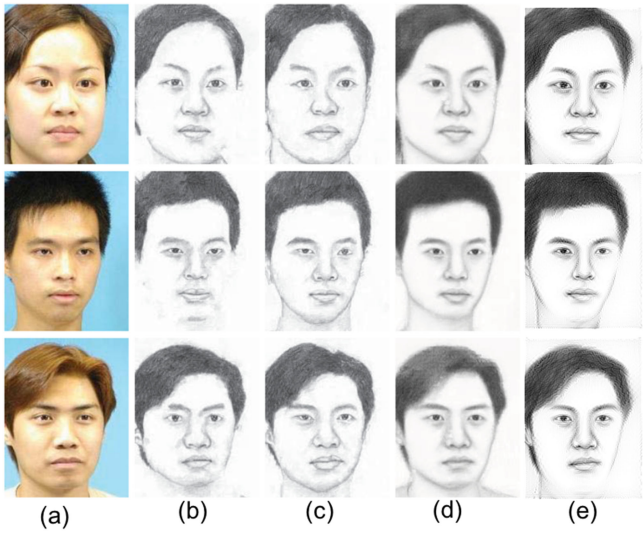


Fig. 16. Pose variation comparisons for synthesized results by different methods. (a) Input face photo. (b) Synthesis results by MRF [17]. (c) Synthesis results by [15]. (d) Synthesis results by MWF [18]. (e) Face sketch synthesis results using our approach.

under different poses and lighting conditions. In Fig. 16, we compare different synthesis methods for face with poses variation. We could find that our approach outperforms other methods in detailed expression. Especially in nose, eyes and mouth regions, our results maintain sharp edges compared to other methods. However, there are some drawbacks in our results, one of which lies in that our method could not contain face contours when input face has pose variation. The reason may be that in the training dataset, all of the boundaries between background and faces have a certain lighting direction. When matching with front face that lighting direction can be correctly considered. As respect to different poses, the lighting conditions have changed to difficult situations, which result in inaccurate matching between neural patches. In our following work, we plan to enhance on more complicated poses and light conditions. In addition, we have used more datasets to test our approach, i.e., XM2VTSand CUFSF. The two datasets contain sketches with shape exaggeration drawn by artists. The XM2VTS database has 295 pairs of face and sketch, and the CUFSF database has 1194 pairs of face and sketch. We apply the leave-one-out strategy to test our synthesis results on each of the dataset. Fig. 17 shows our more results on the XM2VTS dataset and CUFSF dataset. From Fig. 17, we can see that our results are with good visual effects. Please note that the artists' sketch drawing style for the CUFSF dataset are quite different from the CUHK Student, AR and XM2VTS dataset. Thus, the result style of Fig. 17b may look differently from results on other three datasets.

6 Conclusion and Future Work

In this paper, we present a neural representation based face sketch synthesis framework. The proposed framework pieces neural patches together in key layers, and utilizes them to serve as target feature maps. By the guidance of the target feature maps, an input noise image is gradually optimized to a face sketch image. Our approach utilizes the Enhanced 3D PatchMatch to search in training photo

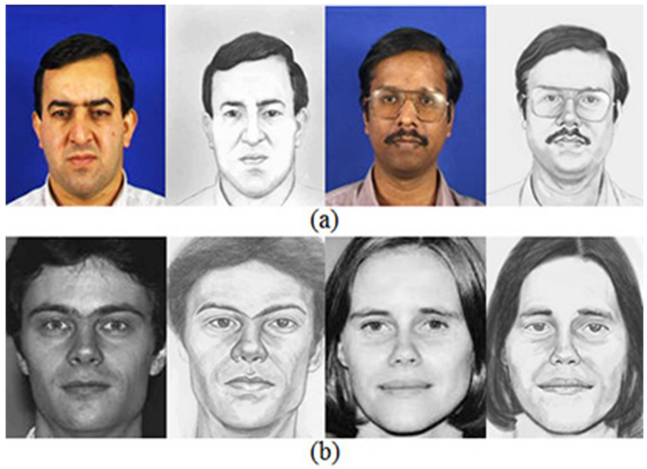


Fig. 17. More experiments on different datasets using our approach. (a) Results on XM2VTS dataset. (b) Results on CUFSF dataset.

feature map spaces in the first stage. Then, the cross-layer cost aggregation in object regions is used in order to find the best match with inter-layer consistency for query neural patch in the second stage. Since our method synthesizes image based on neural patches rather than RGB patches, our results could express sharp sketch edges and avoid mosaic effects at the same time. Compared to former feature maps based methods, the removal of content loss and adding of style loss term stabilize the general sketch tone of results, and make their histogram resemble true pencil sketch's. Our approach is also previous to end-to-end CNN-based model [27] in that our results have much sharper edges and look like be made up of sketch strokes. In conclusion, our approach makes a perfect balance between vivid sketch artistic style and accurate face structure. Our work still has its limitations. The results produced by our method may not be fully satisfactory when the face poses or light conditions are too complicated. We will work on to address such complications as our future work. We will also work on incorporating latest face registration techniques into our Enhanced 3D PatchMatch to ensure the spatial correspondence between patches in the training face photo and face sketch pairs to open our method for more general datasets.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (61572316, 61671290, 61872241), National Key Research and Development Program of China (2016 YFC1300302, 2017YFE0104000), and Science and Technology Commission of Shanghai Municipality (16DZ0501100, 17411952600).

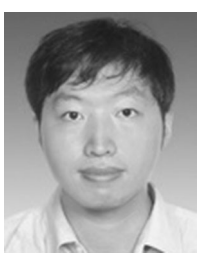
References

[1] W. Zhang, X. Wang, and X. Tang, "Coupled information-theoretic encoding for face photo-sketch recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011, pp. 513–520.

[2] T. Isenberg, P. Neumann, S. Carpendale, M. C. Sousa, and J. A. Jorge, "Non-photorealistic rendering in context: An observational study," in *Proc. 4th Int. Symp. Non-Photorealistic Animation Rendering*, 2006, pp. 115–126.

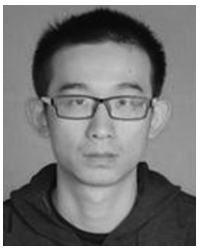
[3] A. Gooch, B. Gooch, P. Shirley, and E. Cohen, "A non-photorealistic lighting model for automatic technical illustration," in *Proc. 25th Annu. Conf. Comput. Graph. Interactive Tech.*, 1998, pp. 447–452.

[4] G. Winkenbach and D. H. Salesin, "Computer-generated pen-and-ink illustration," in *Proc. 21st Annu. Conf. Comput. Graph. Interactive Tech.*, 1994, pp. 91–100.
[5] Y. J. Huang, W. C. Lin, I. C. Yeh, and T. Y. Lee, "Geometric and textural blending for 3d model stylization," *IEEE Trans. Vis. Comput. Graph.*, vol. 24, no. 2, pp. 1114–1126, Feb. 2018.
[6] S. S. Lin, C. C. Morace, C. H. Lin, L. F. Hsu, and T. Y. Lee, "Generation of escher arts with dual perception," *IEEE Trans. Vis. Comput. Graph.*, vol. 24, no. 2, pp. 1103–1113, Feb. 2018.
[7] H. Kang, S. Lee, and C. K. Chui, "Flow-based image abstraction," *IEEE Trans. Vis. Comput. Graph.*, vol. 15, no. 1, pp. 62–76, Jan./Feb. 2009.
[8] S. Brooks, "Mixed media painting and portraiture," *IEEE Trans. Vis. Comput. Graph.*, vol. 13, no. 5, pp. 1041–1054, Sep./Oct. 2007.
[9] W. Zhang, C. Cao, S. Chen, J. Liu, and X. Tang, "Style transfer via image component analysis," *IEEE Trans. Multimedia*, vol. 15, no. 7, pp. 1594–1601, Nov. 2013.
[10] E. B. Lum and K.-L. Ma, "Non-photorealistic rendering using watercolor inspired textures and illumination," in *Proc. Pacific Graphics*, 2001, pp. 322–330.
[11] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *CoRR*, vol. abs/1508.06576, pp. 1–16, 2015.
[12] Y. C. Lai, B. A. Chen, K. W. Chen, W. L. Si, C. Y. Yao, and E. Zhang, "Data-driven npr illustrations of natural flows in chinese painting," *IEEE Trans. Vis. Comput. Graph.*, vol. 23, no. 12, pp. 2535–2549, Dec. 2017.
[13] F.-L. Zhang, J. Wang, E. Shechtman, Z.-Y. Zhou, J.-X. Shi, and S. M. Hu, "PlenoPatch: Patch-based plenoptic image manipulation," *IEEE Trans. Vis. Comput. Graph.*, vol. 23, no. 5, pp. 1561–1573, May 2017.
[14] Z. Zhu, H.-Z. Huang, Z.-P. Tan, K. Xu, and S.-M. Hu, "Faithful completion of images of scenic landmarks using internet images," *IEEE Trans. Vis. Comput. Graph.*, vol. 22, no. 8, pp. 1945–1958, Aug. 2016.
[15] W. Zhang, X. Wang, and X. Tang, "Lighting and pose robust face sketch synthesis," in *Proc. 11th Eur. Conf. Comput. Vis.: Part VI*, 2010, pp. 420–433.
[16] X. Tang and X. Wang, "Face sketch synthesis and recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2003, pp. 687–694.
[17] X. Wang and X. Tang, "Face photo-sketch synthesis and recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 11, pp. 1955–1967, Nov. 2009.
[18] H. Zhou, Z. Kuang, and K.-Y. K. Wong, "Markov weight fields for face sketch synthesis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 1091–1097.
[19] Y. Song, L. Bao, Q. Yang, and M.-H. Yang, "Real-time exemplar-based face sketch synthesis," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 800–813.
[20] C. T. Tu, Y. H. Chan, and Y. C. Chen, "Facial sketch synthesis using 2D direct combined model-based face-specific Markov network," *IEEE Trans. Image Process.*, vol. 25, no. 8, pp. 3546–3561, Aug. 2016.
[21] X. Gao, N. Wang, D. Tao, and X. Li, "Face sketch-photo synthesis and retrieval using sparse representation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 8, pp. 1213–1226, Aug. 2012.
[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst. - Vol. 1*, 2012, pp. 1097–1105.
[23] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 2, pp. 295–307, Feb. 2016.
[24] Z. Cheng, Q. Yang, and B. Sheng, "Deep colorization," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 415–423.
[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.
[26] E. Simo-Serra , S. Iizuka, K. Sasaki, and H. Ishikawa, "Learning to simplify: Fully convolutional networks for rough sketch cleanup," *ACM Trans. Graph.*, vol. 35, no. 4, pp. 121:1–121:11, 2016.
[27] L. Zhang, L. Lin, X. Wu, S. Ding, and L. Zhang, "End-to-end photo-sketch generation via fully convolutional representation learning," in *Proc. 5th ACM Int. Conf. Multimedia Retrieval*, 2015, pp. 627–634.
[28] C. Li and M. Wand, "Combining Markov random fields and convolutional neural networks for image synthesis," in *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2479–2486.
[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, pp. 1–14, 2014.
[30] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "PatchMatch: A randomized correspondence algorithm for structural image editing," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 1–11, 2009.
[31] N. Wang, D. Tao, X. Gao, X. Li, and J. Li, "A comprehensive survey to face hallucination," *Int. J. Comput. Vis.*, vol. 106, no. 1, pp. 9–30, 2014.
[32] M. Zhu, N. Wang, X. Gao, and J. Li, "Deep graphical feature learning for face sketch synthesis," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, 2017, pp. 3574–3580.
[33] Q. Liu, X. Tang, H. Jin, H. Lu, and S. Ma, "A nonlinear approach for face sketch synthesis and recognition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2005, pp. 1005–1010.
[34] N. Wang, X. Gao, L. Sun, and J. Li, "Bayesian face sketch synthesis," *IEEE Trans. Image Process.*, vol. 26, no. 3, pp. 1264–1274, Mar. 2017.
[35] Y. Song, L. Bao, S. He, Q. Yang, and M.-H. Yang, "Stylizing face images via multiple exemplars," *Comput. Vis. Image Understanding*, vol. 162, pp. 135–145, 2017.
[36] N. Wang, X. Gao, and J. Li, "Random sampling for fast face sketch synthesis," *Pattern Recognit.*, vol. 76, pp. 215–227, 2018.
[37] Y. Song, J. Zhang, L. Bao, and Q. Yang, "Fast preprocessing for robust face sketch synthesis," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, 2017, pp. 4530–4536.
[38] K. Zhang, Y. Fang, D. Min, L. Sun, S. Yang, S. Yan, and Q. Tian, "Cross-scale cost aggregation for stereo matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 1590–1597.
[39] C. Lu, L. Xu, and J. Jia, "Combining sketch and tone for pencil drawing production," in *Proc. Symp. Non-Photorealistic Animation Rendering*, 2012, pp. 65–73.
[40] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 5188–5196.
[41] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 818–833.
[42] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization," *ACM Trans. Math. Softw.*, vol. 23, no. 4, pp. 550–560, 1997.
[43] L. Zhang, L. Zhang, X. Mou, and D. Zhang, "FSIM: A feature similarity index for image quality assessment," *IEEE Trans. Image Process.*, vol. 20, no. 8, pp. 2378–2386, Aug. 2011.
[44] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
[45] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006.
[46] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2242–2251.
[47] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5967–5976.
[48] S. Zhang, X. Gao, N. Wang, J. Li, and M. Zhang, "Face sketch synthesis via sparse representation-based greedy search," *IEEE Trans. Image Process.*, vol. 24, no. 8, pp. 2466–2477, Aug. 2015.
[49] K. Delac, M. Grgic, and S. Grgic, "Independent comparative study of PCA, ICA, and LDA on the FERET data set," *Int. J. Imaging Syst. Technol.*, vol. 15, no. 5, pp. 252–260, 2005.

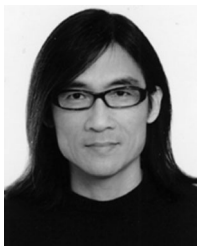


Bin Sheng received the PhD degree in computer science and engineering from The Chinese University of Hong Kong, Hong Kong, China. He is currently an associate professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China. His current research interests include non-photorealistic rendering, machine learning, virtual reality and computer graphics.

Ping Li received the PhD degree from The Chinese University of Hong Kong, Hong Kong, China. He is currently an assistant professor with the Macau University of Science and Technology, Macau, China. His current research interests include image/video stylization, big data visualization, GPU acceleration, and creative media. He has one image/video processing national invention patent, and has excellent research project reported worldwide by *ACM TechNews*.



Chenhao Gao received the BEng degree in automobile engineering from Tongji University, Shanghai, China. He is currently working toward the MEng degree in computer science in the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China. His current research interests include face sketch synthesis, deep neural networks, image processing, and computer vision.



Kwan-Liu Ma received the PhD degree in computer science from the University of Utah. He is a professor of computer science with the University of California, Davis, where he leads VIDi Labs and directs the UC Davis Center for Visualization. His research interests include visualization, highperformance computing, and user interface design. He received the IEEE VGTC 2013 Visualization Technical Achievement Award for his significant research contributions. He has served as a papers co-chair for SciVis, InfoVis, EuroVis, and PacificVis, and also on the editorial board of the *IEEE Transactions Visualization and Computer Graphics* (2007-2011). He is a fellow of the IEEE.