DeepFaceDrawing: Deep Generation of Face Images from Sketches

SHU-YU CHEN†, Institute of Computing Technology, CAS and University of Chinese Academy of Sciences
WANCHAO SU†, School of Creative Media, City University of Hong Kong
LIN GAO*, Intelligent Graphics Laboratory, IDEI, ICT, CAS and University of Chinese Academy of Sciences
SHIHONG XIA, Institute of Computing Technology, CAS and University of Chinese Academy of Sciences
HONGBO FU, School of Creative Media, City University of Hong Kong

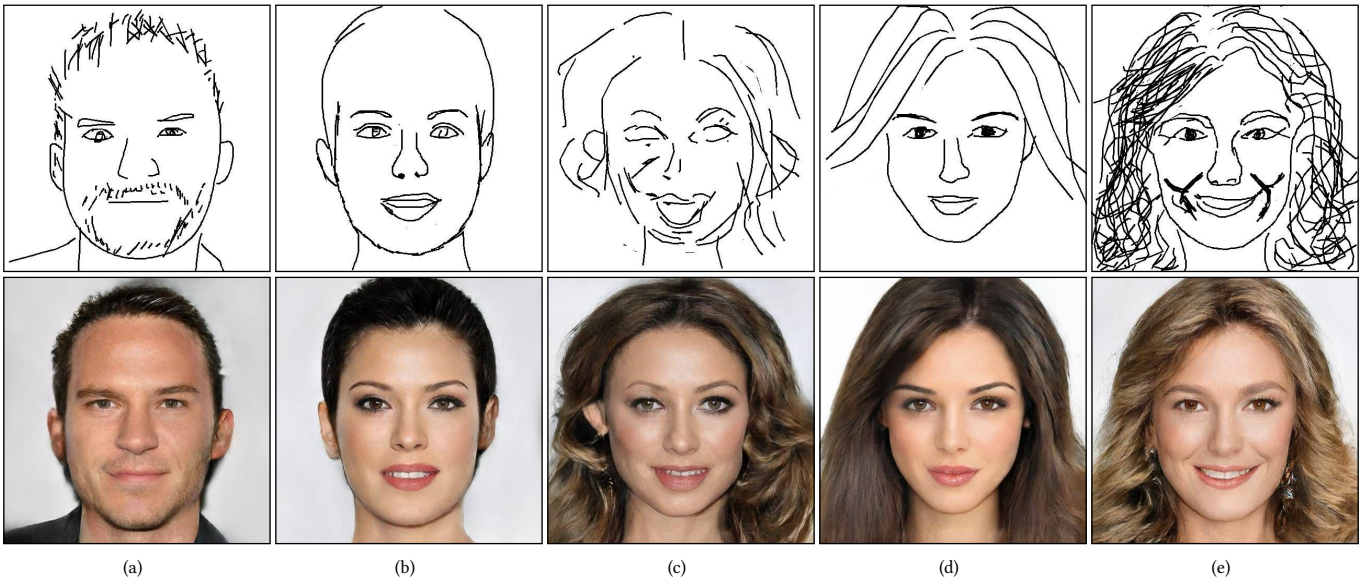

Fig. 1. Our DeepFaceDrawing system allows users with little training in drawing to produce high-quality face images (Bottom) from rough or even incomplete freehand sketches (Top). Note that our method faithfully respects user intentions in input strokes, which serve more like soft constraints to guide image synthesis.

Recent deep image-to-image translation techniques allow fast generation of face images from freehand sketches. However, existing solutions tend to overfit to sketches, thus requiring professional sketches or even edge maps as input. To address this issue, our key idea is to implicitly model the shape space of plausible face images and synthesize a face image in this space to approximate an input sketch. We take a local-to-global approach. We first learn feature embeddings of key face components, and push corresponding parts of input sketches towards underlying component manifolds defined by the feature vectors of face component samples. We also propose another deep neural network to learn the mapping from the embedded component features to realistic images with multi-channel feature maps as intermediate results to improve the information flow. Our method essentially uses input sketches as soft constraints and is thus able to produce high-quality face images even from rough and/or incomplete sketches. Our tool is easy to use even for non-artists, while still supporting fine-grained control of shape details. Both qualitative and quantitative evaluations show the superior generation ability of our system to existing and alternative solutions. The usability and expressiveness of our system are confirmed by a user study.

CCS Concepts: • **Human-centered computing** → **Graphical user interfaces**; • **Computing methodologies** → *Perception*; *Texturing*; **Image processing**.

Additional Key Words and Phrases: image-to-image translation, feature embedding, sketch-based generation, face synthesis

1 INTRODUCTION

Creating realistic human face images from scratch benefits various applications including criminal investigation, character design, educational training, etc. Due to their simplicity, conciseness and ease of use, sketches are often used to depict desired faces. The recently proposed deep learning based image-to-image translation techniques (e.g., [19, 38]) allow automatic generation of photo images from sketches for various object categories including human faces, and lead to impressive results.

Most of such deep learning based solutions (e.g., [6, 19, 26, 38]) for sketch-to-image translation often take input sketches almost fixed and attempt to infer the missing texture or shading information between strokes. To some extent, their problems are formulated more like *reconstruction* problems with input sketches as *hard* constraints. Since they often train their networks from pairs of real

† Authors contributed equally.
* Corresponding author.
Webpage: http://geometrylearning.com/DeepFaceDrawing/

images and their corresponding edge maps, due to the data-driven nature, they thus require test sketches with quality similar to edge maps of real images to synthesize realistic face images. However, such sketches are difficult to make especially for users with little training in drawing.

To address this issue, our key idea is to implicitly learn a space of plausible face sketches from real face sketch images and find the closest point in this space to *approximate* an input sketch. In this way, sketches can be used more like *soft* constraints to guide image synthesis. Thus we can increase the plausibility of synthesized images even for rough and/or incomplete input sketches while respecting the characteristics represented in the sketches (e.g., Figure 1 (a-d)). Learning such a space globally (if exists) is not very feasible due to the limited training data against an expected high-dimensional feature space. This motivates us to implicitly model component-level manifolds, which makes a better sense to assume each component manifold is low-dimensional and locally linear [32]. This decision not only helps locally span such manifolds using a limited amount of face data, but also enables finer-grained control of shape details (Figure 1 (e)).

To this end we present a novel deep learning framework for sketch-based face image synthesis, as illustrated in Figure 3. Our system consists of three main modules, namely, *CE* (*Component Embedding*), *FM* (*Feature Mapping*), and *IS* (*Image Synthesis*). The *CE* module adopts an auto-encoder architecture and separately learns five feature descriptors from the face sketch data, namely, for "left-eye", "right-eye", "nose", "mouth", and "remainder" for locally spanning the component manifolds. The *FM* and *IS* modules together form another deep learning sub-network for conditional image generation, and map component feature vectors to realistic images. Although *FM* looks similar to the decoding part of *CE*, by mapping the feature vectors to 32-channel feature maps instead of 1-channel sketches, it improves the information flow and thus provides more flexibility to fuse individual face components for higher-quality synthesis results.

Inspired by [25], we provide a shadow-guided interface (implemented based on *CE*) for users to input face sketches with proper structures more easily (Figure 8). Corresponding parts of input sketches are projected to the underlying facial component manifolds and then mapped to the corresponding feature maps for conditions for image synthesis. Our system produces high-quality realistic face images (with resolution of 512×512), which faithfully respect input sketches. We evaluate our system by comparing with the existing and alternative solutions, both quantitatively and qualitatively. The results show that our method produces visually more pleasing face images. The usability and expressiveness of our system are confirmed by a user study. We also propose several interesting applications using our method.

2 RELATED WORK

Our work is related to existing works for drawing assistance and conditional face generation. We focus on the works closely related to ours. A full review on such topics is beyond the scope of our paper.

2.1 Drawing Assistance

Multiple guidance or suggestive interfaces (e.g., [17]) have been proposed to assist users in creating drawings of better quality. For example, Dixon et al. [7] proposed *iCanDraw*, which provides corrective feedbacks based on an input sketch and facial features extracted from a reference image. *ShadowDraw* by Lee et al. [25] retrieves real images from an image repository involving many object categories for an input sketch as query and then blends the retrieved images as shadow for drawing guidance. Our shadow-guided interface for inputting sketches is based on the concept of *ShadowDraw* but specially designed for assisting in face drawing. Matsui et al. [29] proposed *DrawFromDrawings*, which allows the retrieval of reference sketches and their interpolation with an input sketch. Our solution for projecting an input sketch to underlying component manifolds follows a similar retrieval-and-interpolation idea but we perform this in the learned feature spaces, without explicit correspondence detection, as needed by *DrawFromDrawings*. Unlike the above works, which aim to produce quality sketches as output, our work treats such sketches as possible inputs and we are more interested in producing realistic face images even from rough and/or incomplete sketches.

Another group of methods (e.g., [1, 18]) take a more aggressive way and aim to automatically correct input sketches. For example, Limpaecher et al. [27] learn a correction vector field from a crowdsourced set of face drawings to correct a face sketch, with the assumption that such face drawings and the input sketch is for a same subject. Xie et al. [41] and Su et al. [36] propose optimization-based approaches for refining sketches roughly drawn on a reference image. We refine an input sketch by projecting individual face components of the input sketch to the corresponding component manifolds. However, as shown in Figure 5, directly using such refined component sketches as input to conditional image generation might cause artifacts across facial components. Since our goal is sketch-based image synthesis, we thus perform sketch refinement only implicitly.

2.2 Conditional Face Generation

In recent years, conditional generative models, in particular, conditional Generative Adversarial Networks [11] (GANs), have been popular for image generation conditioned on various input types. Karras et al. [22] propose an alternative for the generator in GAN that separates the high level face attributes and stochastic variations in generating high quality face images. Based on conditional GANs [30], Isola et al. [19] present the *pix2pix* framework for various image-and-image translation problems like image colorization, semantic segmentation, sketch-to-image synthesis, etc. Wang et al. [38] introduce *pix2pixHD*, an improved version of *pix2pix* to generate higher-resolution images, and demonstrate its application to image synthesis from semantic label maps. Wang et al. [37] generate an image given a semantic label map as well as an image exemplar. Sangkloy et al. [34] take hand-drawn sketches as input and colorize them under the guidance of user-specified sparse color strokes. These systems tend to overfit to conditions seen during training, and thus when sketches being used as conditions, they achieve quality results only given edge maps as input. To address this issue, instead

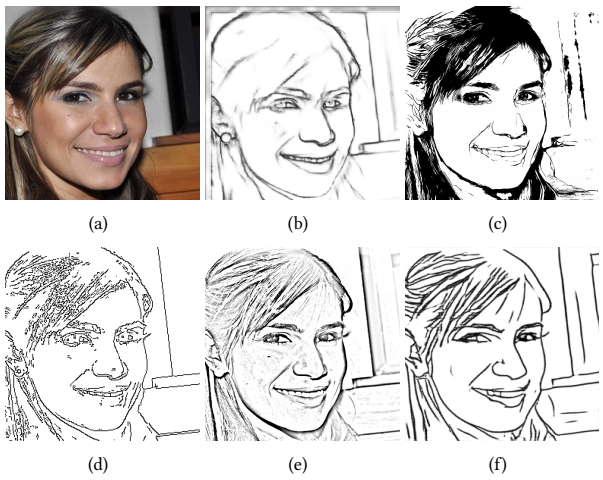

Fig. 2. The comparisons of different edge extraction methods. (a): Input real image. (b): Result by HED [42]. (c): Result by APDrawingGAN [43]. (d): Canny edges [4]. (e): the result by the Photocopy filter in Photoshop. (f): Simplification of (e) by [35]. Photo (a) courtesy of © LanaLucia.

of training an end-to-end network for sketch-to-image synthesis, we exploit the domain knowledge and condition GAN on feature maps derived from the component feature vectors.

Considering the known structure of human faces, researchers have explored component-based methods (e.g., [15]) for face image generation. For example, given an input sketch, Wu and Dai [40] first retrieve best-fit face components from a database of face images, then compose the retrieved components together, and finally deform the composed image to approximate a sketch. Due to their synthesis-and-deforming strategy, their solution requires a well-drawn sketch as input. To enable component-level controllability, Gu et al. [12] use auto-encoders to learn feature embeddings for individual face components, and fuse component feature tensors in a mask-guided generative network. Our *CE* module is inspired by their work. However, their local embeddings learned from real images are mainly used to generate portrait images with high diversity while ours learned from sketch images are mainly for implicitly refining and completing input sketches.

Conditional GANs have also been adopted for local editing of face images, via interfaces either based on semantic label masks [12, 24, 37] or sketches [20, 31]. While the former is more flexible for applications such as component transfer and style transfer, the latter provides a more direct and finer control of details, even within face components. Deep sketch-based face editing is essentially a sketch-guided image completion problem, which requires the completion of missing parts such that the completed content faithfully reflects an input sketch and seamlessly connects to the known context. It thus requires different networks from ours. The *SN-patchGAN* proposed by Jo and Park [20] is able to produce impressive details for example for a sketched earring. However, this also implies that their solution requires high-quality sketches as input. To tolerate the errors in hand-drawn sketches, Portenier et al. [31] propose to use smoothed edge maps as part of the input to their conditional completion network. Our work takes a step further to implicitly model face component manifolds and perform manifold projection.

Several attempts have also been made to generate images from incomplete sketches. To synthesize face images from line maps possibly with some missing face components, Li et al. [26] proposed a conditional self-attention GAN with a multi-scale discriminator, where a large-scale discriminator enforces the completeness of global structures. Although their method leads to visually better results than *pix2pix* [19] and *SkethyGAN* [5], due to the direct condition on edge maps, their solution has poor ability to handle hand-drawn sketches. Ghosh et al. [10] present a shape generator to complete a partial sketch before image generation, and present interesting auto-completion results. However, their synthesized images still exhibit noticeable artifacts, since the performance of their image generation step (i.e., *pix2pixHD* [38] for single-class generation and *SkinnyResNet* [10] for multi-class generation) heavily depends on the quality of input sketches. A similar problem exists with the progressive image reconstruction network proposed by You et al. [45], which is able to reconstruct images from extremely sparse inputs but still requires relatively accurate inputs.

To alleviate the heterogeneity of input sketches and real face images, some researchers resort to the unpaired image-to-image methods (e.g., [16, 44, 48]). These methods adopt self-consistent constraints to solve the lack of paired data. While the self-consistent mechanism ensures the correspondence between the input and the reconstructed input, there is no guarantee for the correspondence between the input and the transformed representation. Since our goal is to transform sketches to the corresponding face images, these frameworks are not suitable for our task. In addition, there are some works leveraging the image manifolds. For example, Lu et al. [28] learn a fused representation from shape and texture features to construct a face retrieval system. In contrast, our method not only retrieves but also interpolates the face representations in generation. Zhu et al. [47] first construct a manifold with the real image dataset, then predict a dense correspondence between a projected source image and an edit-oriented "feasible" target in the manifold, and finally apply the dense correspondence back to the original source image to complete the visual manipulation. In contrast, our method directly interpolates the nearest neighbors of the query and feeds the interpolation result to the subsequent generation process. Compared to Zhu et al. [47], our method is more direct and efficient for the sketch-based image generation task.

3 METHODOLOGY

The 3D shape space of human faces has been well studied (see the classic morphable face model [3]). A possible approach to synthesize realistic faces from hand-drawn sketches is to first project an input sketch to such a 3D face space [13] and then synthesize a face image from a generated 3D face. However, such a global parametric model is not flexible enough to accommodate rich image details or support local editing. Inspired by [8], which shows the effectiveness of a local-global structure for faithful local detail synthesis, our method aims for modeling the shape spaces of face components in the image domain.

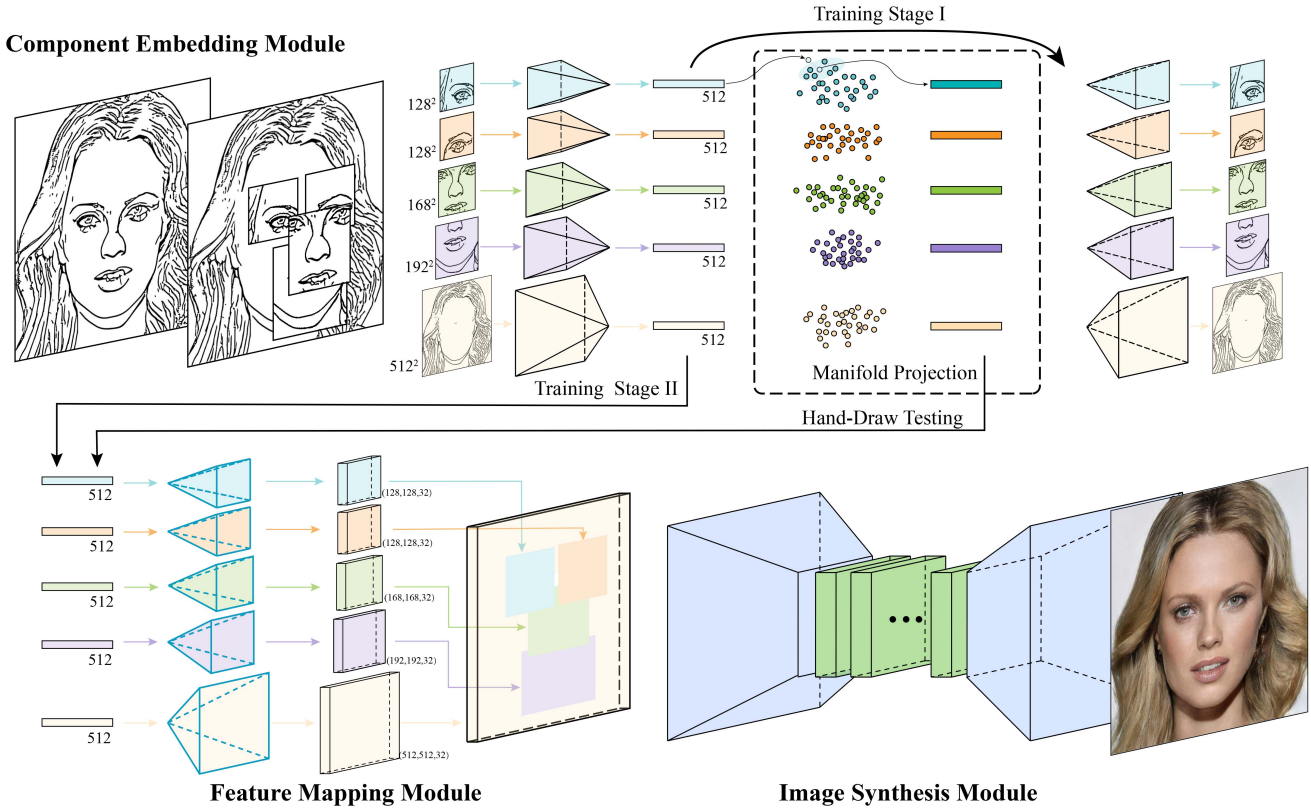

Fig. 3. Illustration of our network architecture. The upper half is the *Component Embedding* module. We learn feature embeddings of face components using individual auto-encoders. The feature vectors of component samples are considered as the point samples of the underlying component manifolds and are used to refine an input hand-drawn sketch by projecting its individual parts to the corresponding component manifolds. The lower half illustrates a sub-network consisting of the *Feature Mapping* (*FM*) and the *Image Synthesis* (*IS*) modules. The *FM* module decodes the component feature vectors to the corresponding multi-channel feature maps ($H \times W \times 32$), which are combined according to the spatial locations of the corresponding facial components before passing them to the *IS* module.

To achieve this, we first learn the feature embeddings of face components (Section 3.2). For each component type, the points corresponding to component samples implicitly define a manifold. However, we do not explicitly learn this manifold, since we are more interested in knowing the closest point in such a manifold given a new sketched face component, which needs to be refined. Observing that in the embedding spaces semantically similar components are close to each other, we assume that the underlying component manifolds are locally linear. We then follow the main idea of the classic locally linear embedding (LLE) algorithm [32] to project the feature vector of the sketched face component to its component manifold (Section 3.3).

The learned feature embeddings also allow us to guide conditional sketch-to-image synthesis to explicitly exploit the information in the feature space. Unlike traditional sketch-to-image synthesis methods (e.g., [19, 38]), which learn conditional GANs to translate sketches to images, our approach forces the synthesis pipeline to go through the component feature spaces and then map 1-channel feature vectors to 32-channel feature maps before the use of a conditional GAN (Section 3.2). This greatly improves the information flow and benefits component fusion. Below we first discuss our data preparation procedure (Section 3.1). We then introduce our novel pipeline for sketch-to-image synthesis (Section 3.2), and our approach for manifold projection (Section 3.3). Finally present our shadow-guided interface (Section 3.4).

3.1 Data Preparation

To train our network, it requires a reasonably large-scale dataset of face sketch-image pairs. There exist several relevant datasets like the CUHK face sketch database [39, 46]. However, the sketches in such datasets involve shading effects while we expect a more abstract representation of faces using sparse lines. We thus contribute to a new dataset of pairs of face images and corresponding synthesized sketches. We build this on the face image data of CelebAMask-HQ [24], which contains high-resolution facial images with semantic masks of facial attributes. For simplicity, we currently focus on front faces, without decorative accessories (e.g., glasses, face masks).

To extract sparse lines from real images, we have tried the following edge detection methods. As shown in Figure 2 (b) and (d),

the holistically-nested edge detection (HED) method [42] and the traditional Canny edge detection algorithm [4] tend to produce edge maps with discontinuous lines. APDrawingGAN [43], a very recent approach for generating portrait drawings from face photos leads to artistically pleasing results, which, however, are different from our expectation (e.g., see the regional effects in the hair area and missing details around the mouth in Figure 2 (c)). We also resorted to the Photocopy filter in Photoshop, which preserves facial details but meanwhile brings excessive noise (Figure 2 (e)). By applying the sketch simplification method by Simo-Serra et al. [35] to the result by the Photocopy filter, we get an edge map with the noise reduced and the lines better resembling hand-drawn sketches (Figure 2 (f)). We thus adopt this approach (i.e., Photocopy + sketch simplification) to prepare our training dataset, which contains 17K pairs of sketch-image pairs (see an example pair in Figure 2 (f) and (a)), with 6247 for male subjects and 11456 for female subjects. Since our dataset is not very large-scale, we reserve the data in the training process as much as possible to provide as many samples as possible to span the component manifolds. Thus we set a training/testing ratio to 20:1 in our experiments. It results in 16,860 samples for training and 842 for testing.

3.2 Sketch-to-Image Synthesis Architecture

As illustrated in Figure 3, our deep learning framework takes as input a sketch image and generates a high-quality facial image of size 512×512. It consists of two sub-networks: The first sub-network is our *CE* module, which is responsible for learning feature embeddings of individual face components using separate auto-encoder networks. This step turns component sketches into semantically meaningful feature vectors. The second sub-network consists of two modules: *FM* and *IS*. *FM* turns the component feature vectors to the corresponding feature maps to improve the information flow. The feature maps of individual face components are then combined according to the face structure and finally passed to *IS* for face image synthesis.

Component Embedding Module. Since human faces share a clear structure, we decompose a face sketch into five components, denoted as S^c, $c \in \{1, 2, 3, 4, 5\}$ for "left-eye", "right-eye", "nose", "mouth", and "remainder", respectively. To handle the details in-between components, we define the first four components simply by using four overlapping windows centered at individual face components (derived from the pre-labeled segmentation masks in the dataset), as illustrated in Figure 3 (Top-Left). A "remainder" image corresponding to the "remainder" component is the same as the original sketch image but with the eyes, nose and mouth removed. Here we treat "left-eye" and "right-eye" separately to best explore the flexibility in the generated faces (see two examples in Figure 4). To better control of the details of individual components, for each face component type we learn a local feature embedding. We obtain the feature descriptors of individual components by using five auto-encoder networks, denoted as $\{E_c, D_c\}$ with E_c being an encoder and D_c a decoder for component c.

Each auto-encoder consists of five encoding layers and five decoding layers. We add a fully connected layer in the middle to ensure the latent descriptor is of 512 dimensions for all the five components.

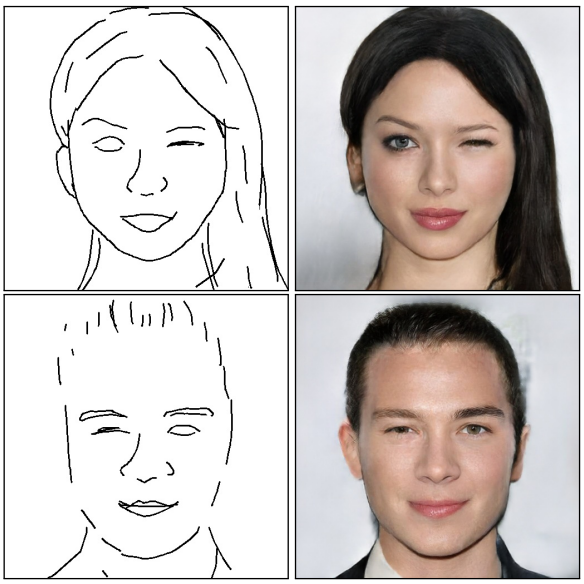

Fig. 4. Two examples of generation flexibility supported by using separate components for the left and right eyes.

We experimented with different numbers of dimensions for the latent representation (128, 256, 512) – we found that 512 dimensions are enough for reconstructing and representing the sketch details. Instead, lower-dimensional representations tend to lead to blurry results. By trial and error, we append a residual block after every convolution/deconvolution operation in each encoding/decoding layer to construct the latent descriptors instead of only using convolution and deconvolution layers. We use Adam solver [23] in the training process. Please find the details of the network architectures and the parameter settings in the supplemental materials.

Feature Mapping Module. Given an input sketch, we can project its individual parts to the component manifolds to increase its plausibility (Section 3.3). One possible solution to synthesize a realistic image is to first convert the feature vectors of the projected manifold points back to the component sketches using the learned decoders $\{D_c\}$, then perform component-level sketch-to-image synthesis (e.g., based on [38]), and finally fuse the component images together to get a complete face. However, this straightforward solution easily leads to inconsistencies in synthesized results in terms of both local details and global styles, since there is no mechanism to coordinate the individual generation processes.

Another possible solution is to first fuse the decoded component sketches into a complete face sketch (Figure 5 (b)) and then perform sketch-to-image synthesis to get a face image (Figure 5 (c)). It can be seen that this solution also easily causes artifacts (e.g., misalignment between face components, incompatible hair styles) in the synthesized sketch, and such artifacts are inherited to the synthesized image, since existing deep learning solutions for sketch-to-image synthesis tend to use input sketches as rather hard constraints, as discussed previously.

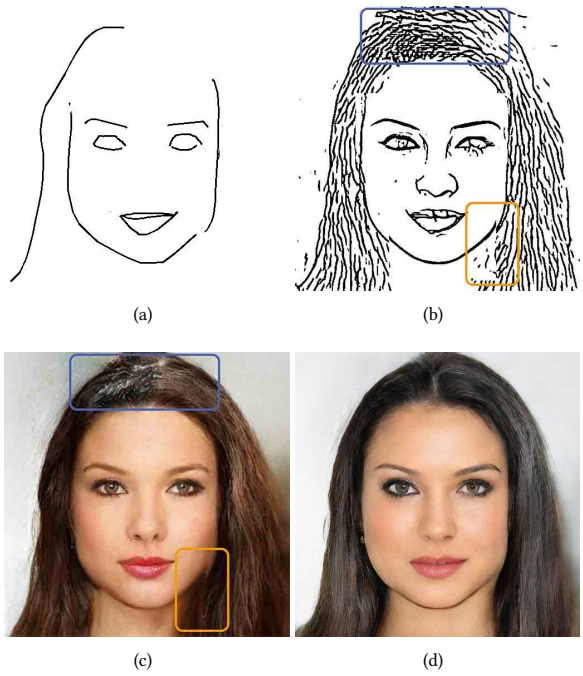

Fig. 5. Given the same input sketch (a), image synthesis conditioned on the feature vectors after manifold projection achieves a more realistic result (d) than that (c) by image synthesis conditioned on an intermediate sketch (b). See the highlighted artifacts in both the intermediate sketch (b) and the corresponding synthesized result (c) by pix2pixHD [38].

We observe that the above issues mainly happen in the overlapping regions of the cropping windows for individual components. Since sketches only have one channel, the incompatibility of neighboring components in the overlapping regions is thus difficult to automatically resolve by sketch-to-image networks. This motivates us to map the feature vectors of sampled manifold points to multi-channel feature maps (i.e., 3D feature tensors). This significantly improves the information flow, and fusing the feature maps instead of sketch components helps resolve the inconsistency between face components.

Since the descriptors for different components bear different semantic meanings, we design the *FM* module with five separate decoding models converting feature vectors to spatial feature maps. Each decoding model consists of a fully connected layer and five decoding layers. For each feature map, it has 32 channels and is of the same spatial size as the corresponding component in the sketch domain. The resulting feature maps for "left-eye", "right-eye", "nose", and "mouth" are placed back to the "remainder" feature maps according to the exact positions of the face components in the input face sketch image to retain the original spatial relations between face components. As illustrated in Figure 3 (Bottom-Center), we use a fixed depth order (i.e., "left/right eyes" > "nose" > "mouth" > "remainder") to merge the feature maps.

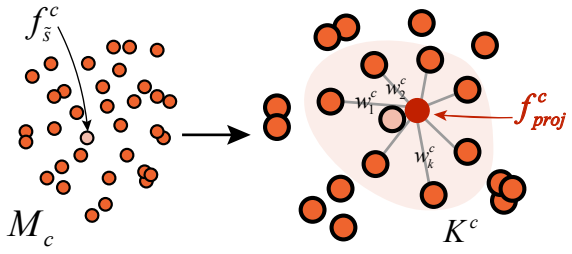

Fig. 6. Illustration of manifold projection. Given a new feature vector $f^c_{\tilde{s}}$, we replace it with the projected feature vector f^c_{proj} using K nearest neighbors of $f^c_{\tilde{s}}$.

Image Synthesis Module. Given the combined feature maps, the *IS* module converts them to a realistic face image. We implement this module using a conditional GAN architecture, which takes the feature maps as input to a generator, with the generation guided by a discriminator. Like the global generator in *pix2pixHD* [38], our generator contains an encoding part, a residual block, and a decoding unit. The input feature maps go through these units sequentially. Similar to [38], the discriminator is designed to determine the samples in a multi-scale manner: we downsample the input to multiple sizes and use multiple discriminators to process different inputs at different scales. We use this setting to learn the high-level correlations among parts implicitly.

Two-stage Training. As illustrated in Figure 3, we adopt a two-stage training strategy to train our network using our dataset of sketch-image pairs (Section 3.1). In Stage I, we train only the *CE* module, by using component sketches to train five individual auto-encoders for feature embeddings. The training is done in a self-supervised manner, with the mean square error (MSE) loss between an input sketch image and the reconstructed image. In Stage II, we fix the parameters of the trained component encoders and train the entire network with the unknown parameters in the *FM* and *IS* modules together in an end-to-end manner. For the GAN in the *IS*, besides the GAN loss, we also incorporate a L_1 loss to further guide the generator and thus ensure the pixel-wise quality of generated images. We use the perceptual loss [21] in the discriminator to compare the high-level difference between real and generated images. Due to the different characteristics of female and male portraits, we train the network using the complete set but constrain the searching space into the male and female spaces for testing.

3.3 Manifold Projection

Let $\mathcal{S} = \{s_i\}$ denote a set of sketch images used to train the feature embeddings of face components (Section 3.2). For each component c, we can get a set of points in the c-component feature space by using the trained encoders, denoted as $\mathcal{F}^c = \{f^c_i = E_c(s^c_i)\}$. Although each feature space is 512-dimensional, given that similar component images are placed closely in such feature spaces, we tend to believe that all the points in $\mathcal{F}^c$ are in an underlying low-dimensional manifold, denoted as $\mathcal{M}^c$, and further assume each component manifold is locally linear: each point and its neighbors lie on or close to a locally linear patch of the manifold [32].

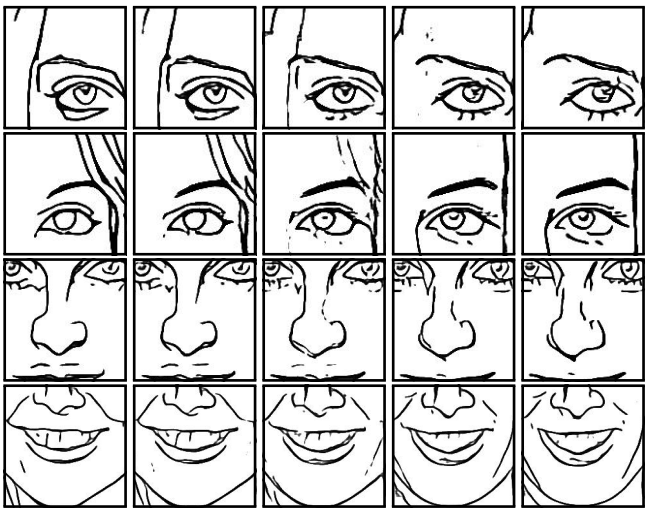

Fig. 7. Illustration of linear interpolation between pairs of randomly selected neighboring component sketches (Leftmost and Rightmost) in the corresponding feature spaces. The middle three images are decoded from the uniformly interpolated feature vectors.

Given an input sketch $\tilde{s}$, to increase its plausibility as a human face, we project its c-th component to $\mathcal{M}^c$. With the locally linear assumption, we follow the main idea of LLE and take a *retrieval-and-interpolation* approach to project the c-th component feature vector of $E_c(\tilde{s}^c)$, denoted as $f_{\tilde{s}}^c$ to $\mathcal{M}^c$, as illustrated in Figure 3.

As illustrated in Figure 6, given the c-th component feature vector $f_{\tilde{s}}^c$, we first find the K nearest samples in $\mathcal{F}^c$ under the Euclidean space. By trial and error, we found that K=10 is sufficient in providing face plausibility while maintaining adequate variations. Let $\mathcal{K}^c = \{s_k^c\}$ (with $\{s_k\} \subset \mathcal{S}$) denote the resulting set of K nearest samples, i.e., the neighbors of $\tilde{s}^c$ on $\mathcal{M}^c$. We then seek a linear combination of these neighbors to reconstruct $\tilde{s}^c$ by minimizing the reconstruction error. This is equivalent to solving for the interpolation weights through the following minimization problem:

$$\min ||f_{\tilde{s}}^c - \sum_{k \in \mathcal{K}^c} w_k^c \cdot f_k^c||_2^2, \quad s.t. \sum_{k \in \mathcal{K}} w_k^c = 1, \tag{1}$$

where w_k^c is the unknown weight for sample s_k^c. The weights can be found by solving a constrained least-squares problem for individual components independently. Given the solved weights $\{w_k^c\}$, the projected point of $\tilde{s}^c$ on $\mathcal{M}^c$ can be computed as

$$f_{proj}^c = \sum_{k \in \mathcal{K}^c} w_k^c \cdot f_k^c. \tag{2}$$

f_{proj}^c is the feature vector of the refined version of $\tilde{s}^c$, and can be passed to the *FM* and *IS* modules for image synthesis.

To verify the local continuity of the underlying manifolds, we first randomly select a sample from $\mathcal{S}$, and for its c-th component randomly select one of its nearest neighbors in the correspondence feature space. We then perform linear interpolation between such a pair of component sketches in the c-th feature space, and reconstruct the interpolated component sketches using the learned c-th decoder D_c. The reconstructed results are shown in Figure 7. It can be seen that as we change the interpolation weight continuously, it results in smooth changes between the consecutive reconstructed component sketches from a pair of selected sketches. This shows the feasibility of our descriptor interpolation.

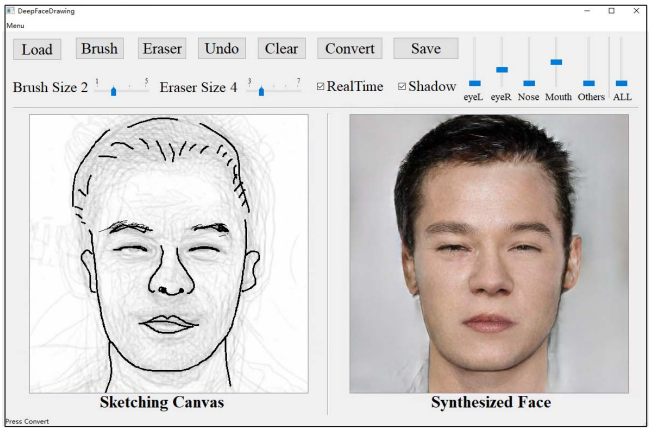

Fig. 8. A screenshot of our shadow-guided sketching interface (Left) for facial image synthesis (Right). The sliders at the up-right corner can be used to control the degree of interpolation between an input sketch and a refined version after manifold project for individual components.

3.4 Shadow-guided Sketching Interface

To assist users, especially those with little training in drawing, inspired by *ShadowDraw* [25], we provide a shadow-guided sketching interface. Given a current sketch $\tilde{s}$, we first find K (K = 10 in our implementation) most similar sketch component images from $\mathcal{S}$ according to $\tilde{s}^c$ by using the Euclidean distance in the feature space. The found component images are then blended as shadow and placed at the corresponding components' positions for sketching guidance (Figure 8 (Left)). Initially when the canvas is empty, the shadow is more blurry. The shadow is updated instantly for every new input stroke. The synthesized image is displayed in the window on the right. Users may choose to update the synthesized image instantly or trigger an "Convert" command. We show two sequences of sketching and synthesis results in Figure 18.

Users with good drawing skills tend to trust their own drawings more than those with little training in drawing. We thus provide a slider for each component type to control the blending weights between a sketched component and its refined version after manifold projection. Let wb^c denote the blending weight for component c. The feature vector after blending can be calculated as:

$$f_{blend}^c = wb^c \times f_{\tilde{s}}^c + (1 - wb^c) \times f_{proj}^c. \tag{3}$$

Feeding f_{blend}^c to the subsequent trained modules, we get a new synthesized image.

Figure 9 shows an example of synthesized results under different values of wb^c. This blending feature is particularly useful for creating faces that are very different from any existing samples or their blending. For example, for the female data in our training set, most of the subjects have long hairstyles. Always pushing our input sketch to such samples would not allow us to create short-hairstyle

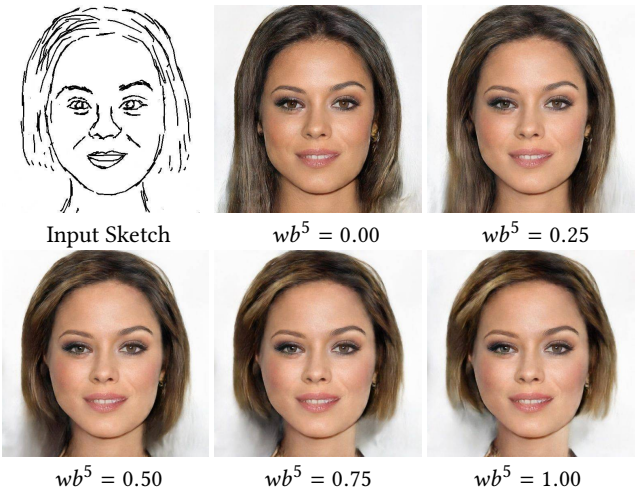

Fig. 9. Interpolating an input sketch and its refined version (for the "remainder" component in this example) after manifold projection under different blending weight values. $wb^c = 1$ means a full use of an input sketch for image synthesis, while by setting $wb^c = 0$ we fully trust the data for interpolation.

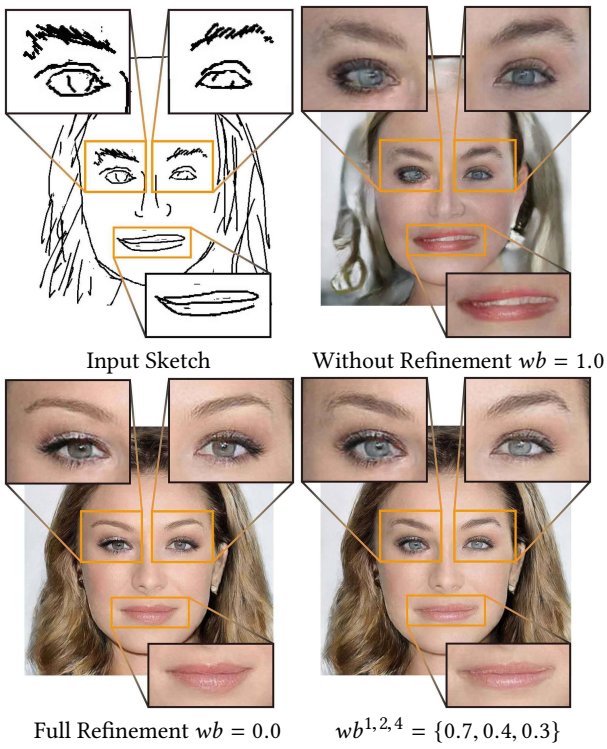

Fig. 10. Blending an input sketch and its refined version after manifold projection for the "left-eye", "right-eye", and "mouth" components. Upper Right: result without any sketch refinement; Lower Left: result with full-degree sketch refinement; Lower Right: result with partial-degree sketch refinement.

effects. This is solved by trusting the input sketch for its "remainder" component by adjusting its corresponding blending weight. Figure 10 shows another example with different blending weights for different components. It can be easily seen that the result with automatic refinement (lower left) is visually more realistic than that without any refinement (upper right). Fine-tuning of the blending weights leads to a result better reflecting the input sketch more faithfully.

4 EXPERIMENTS

We have done extensive evaluations to show the effectiveness of our sketch-to-image face synthesis system and its usability via a pilot study. Below we present some of the obtained results. Please refer to the supplemental materials for more results and an accompanying video for sketch-based image synthesis in action.

Figure 11 shows two representative results where users progressively introduce new strokes to add or stress local details. As shown in the demo video, running on a PC with an Intel i7-7700 CPU, 16GB RAM and a single Nvidia GTX 1080Ti GPU, our method achieves real-time feedback. Thanks to our local-to-global approach, generally more strokes lead to new or refined details (e.g., the nose in the first example, and the eyebrows and wrinkles in the second example), with other areas largely unchanged. Still due to the combination step, local editing might still introduce subtle but global changes. For example, for the first example, the local change of lighting in the nose area leads to the change of highlight in the whole face (especially in the forehead region). Figure 18 shows two more complete sequences of progressive sketching and synthesis, with our shadow-guided interface.

4.1 Usability Study

We conducted a usability study to evaluate the usefulness and effectiveness of our system. 10 subjects (9 male and 1 female, aged from 20 to 26) were invited to participate in this study. We first asked them to self-assess their drawing skills through a nine-point Likert scale (1: novice to 9: professional), and divided them into three groups: 4 novice users (drawing skill score: 1 – 3), 4 middle users (4 – 6), and 2 professional users (7 – 9). Before the drawing session, each participant was given a short tutorial about our system (about 10 minutes). The participants used an iPad with iPencil to remotely control the server PC for drawing. Then each of them was asked to create at least 3 faces using our system. The study ended with a questionnaire to get user feedbacks on *ease-of-use*, *controllability*, *variance of results*, *quality of results*, and *expectation fitness*. The additional comments on our system were also welcome.

Figure 12 gives a gallery of sketches and synthesized faces by the participants. It can be seen that our system consistently produce realistic results given input sketches with different styles and levels of abstraction. For several examples, the participants attempted to depict beard styles via hatching and our system captured the users' intention very well.

Figure 13 shows a radar plot, summarizing quantitative feedbacks on our system for participant groups with different levels of drawing skills. The feedbacks for all the groups of participants were positive in all the measured aspects. Particularly, the participants with good drawing skills felt a high level of controllability, while they gave

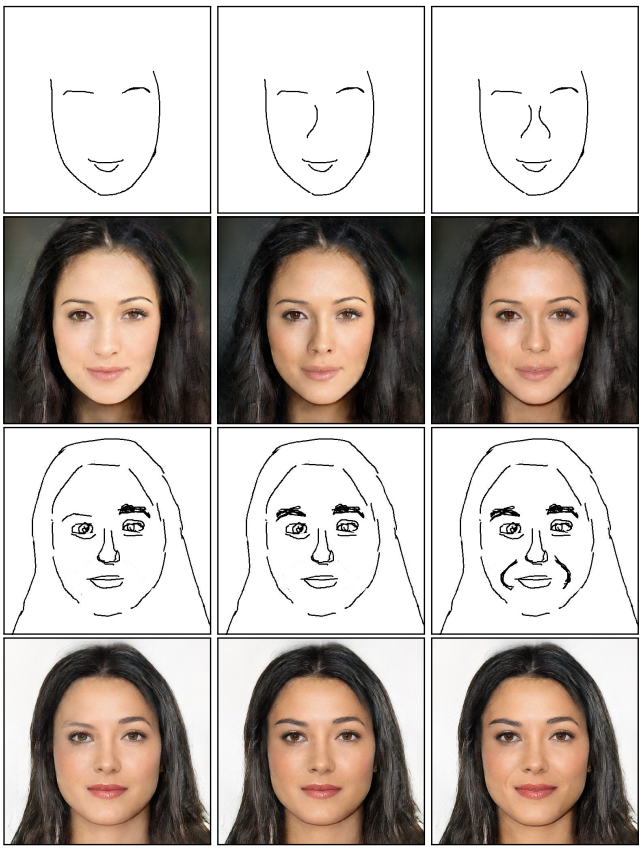

Fig. 11. Representative results through progressive sketching for adding details (1st example) and stressing local details (2nd example).

slightly lower scores for the degree of result variance. Using our system, the average time needed for drawing a face sketch among the participants with different drawing abilities are: 17′14″ (professional), 3′17″ (middle) and 2′26″ (novice). It took much longer for professionals, since they spent more time sketching and refining details. For the refinement sliders, the most frequently used slider was for the "remainder" component (56.69%), which means for more than half of the results, the "remainder" slider was manipulated. In contrast, for the other components we have 21.66% for "left-eye", 12.74% for "right-eye", 12.10% for "nose" and 19.75% for "mouth". For all the adjustments made in the components, participants trust the "remainder" component most, with the averaged confidence 0.78; The least trusted component is "mouth" (0.56); other component confidences are 0.70 ("left-eye"), 0.61 ("right-eye") and 0.58 ("nose"). The averaged confidences implied the importance of sketch refinement in creating the synthesized faces in this study.

All of the participants felt that our system was powerful to create realistic faces using such sparse sketches. They liked the intuitive shadow-guided interface, which was quite helpful for them to construct face sketches with proper structures and layouts. On the other hand, some users, particularly those with good drawing skills, felt that the shadow guidance was sometimes distracting when editing details. This finding is consistent with the conclusions in the original *ShadowDraw* paper [25]. One of the professional users mentioned that automatic synthesis of face images given sparse inputs saved a lot of efforts and time compared to traditional painting software. One professional user mentioned that it would be better if our system could provide color control.

4.2 Comparison with Alternative Refinement Strategies

To refine an input sketch, we essentially take a component-level retrieval-and-interpolation approach. We compare this method with two alternative sketch refinement methods by globally or locally retrieving the most similar sample in the training data. For fair comparison, we use the same *FM* and *IS* modules for image synthesis. For the local retrieval method, it is the same as our method except that for manifold projection we simply retrieve the closest (i.e., top-1 instead of top-K) component sample in each component-level feature space without any interpolation. For the global retrieval method, we replace the *CE* module with a new module for the feature embeddings of entire face sketches. Specifically, we first learn the feature embeddings of the entire face sketch images, and given a new sketch we find the most similar (i.e., top-1) sample in the whole-face feature space. For each component in the globally retrieved sample image, we then encode it using the corresponding trained component-level encoder (i.e., E_c), and pass all the component-level feature vectors to our *FM* and *IS* for image synthesis. Note that we do not globally retrieve real face images, since our goal here is for a fair comparison of the sketch refinement methods.

Figure 14 shows comparison results. From the overlay of input sketches and the retrieved or interpolated sketches, it can be easily seen that the component-level retrieval method returns samples closer to the input component sketches than the global-retrieval method, mainly due to the limited sample data. Thanks to the interpolation step, the interpolated sketches almost perfectly fit the input sketches. Note that we show the decoded sketches after interpolation here only for the comparison purpose, and our conditional image synthesis sub-network takes the interpolated feature vectors as input (Section 3.2).

4.3 Perceptive Evaluation Study

As shown in Figure 14 (Right), the three refinement methods all lead to realistic face images. To evaluate the visual quality and the faithfulness (i.e., the degree of fitness to input sketches) of synthesize results, we conducted a user study.

We prepared a set of input sketches, containing in total 22 sketches, including 9 from the usability study (Section 4.1) and 13 from the authors. This sketch set (see the supplementary materials) covered inputs with different levels of abstraction and different degrees of roughness. We applied the three refinement methods to each input sketch to generate the corresponding synthesized results (see two representative sets in Figure 15).

The evaluation was done through an online questionnaire. 60 participants (39 male, 21 female, aged from 18 to 32) participated in this study. Most of them had no professional training in drawing. We showed each participant four images including input sketch and the three synthesized images, placed side by side in a random order. Each participant was asked to evaluate the quality and faithfulness

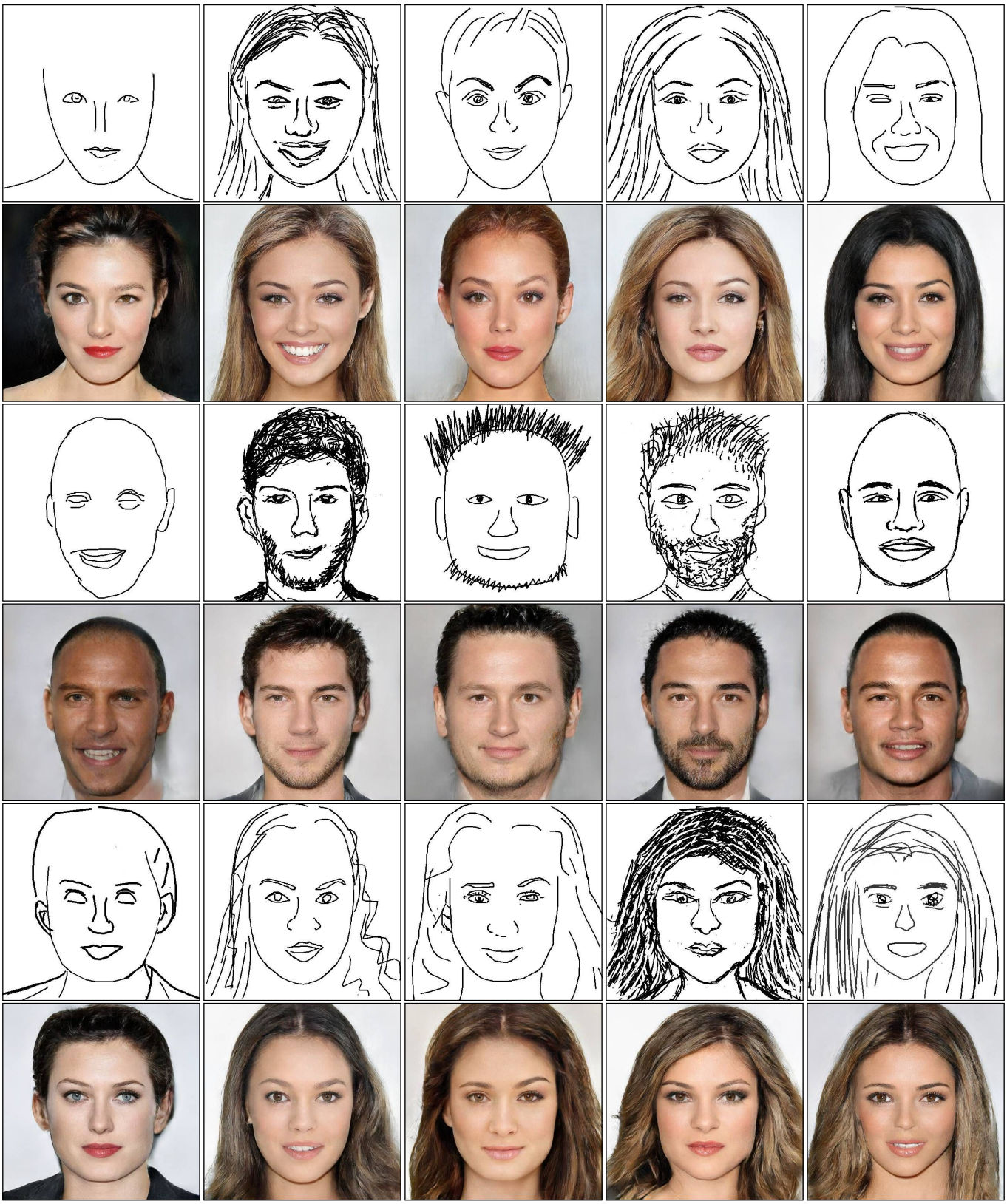

Fig. 12. Gallery of input sketches and synthesized results in the usability study.

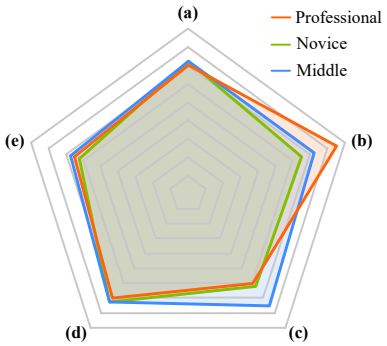

Fig. 13. The summary of quantitative feedback in the usability study. (a) Ease of use. (b) Controllability. (c) Variance of results. (d) Quality of results. (e) Expectation fitness.

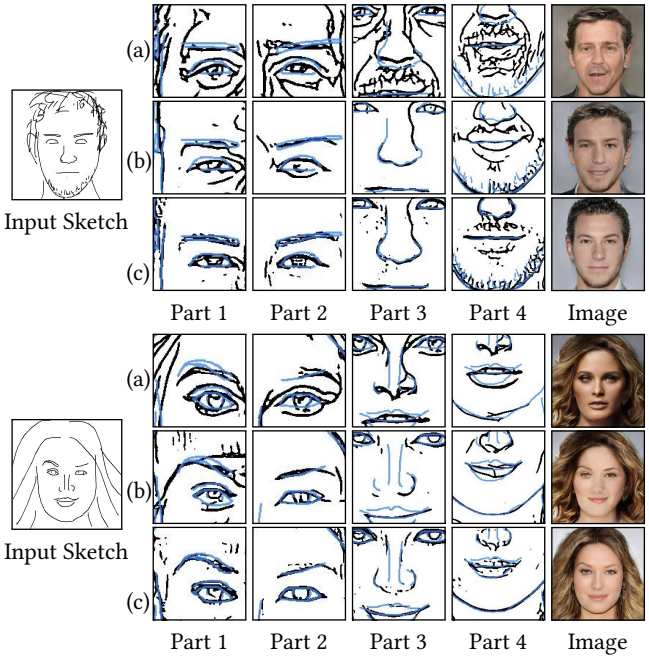

Fig. 14. Comparisons of using global retrieval (a), component-level retrieval (b), and our method (essentially component-level retrieval followed by interpolation) (c) for sketch refinement. The right column shows the corresponding synthesized results. For easy comparison we overlay input sketches (in light blue) on top of the retrieved or interpolated sketches by different methods.

both in a seven-point Likert scale (1 = strongly negative to 7 = strongly positive). In total, to evaluate either the faithfulness or the quality, we got 60 (participants) × 22 (sketches) = 1,320 subjective evaluations for each method.

Figure 16 plots the statistics of the evaluation results. We performed one-way ANOVA tests on the quality and faithfulness scores, and found significant effects for both quality ($F_{(2,63)} = 47.26, p < 0.001$) and faithfulness ($F_{(2,63)} = 51.72, p < 0.001$). Paired t-tests

Fig. 15. Two representative sets of input sketches and synthesized results used in the perceptive evaluation study. From left to right: input sketch, the results by sketch refinement through global retrieval, local retrieval, and local retrieval with interpolation (our method).

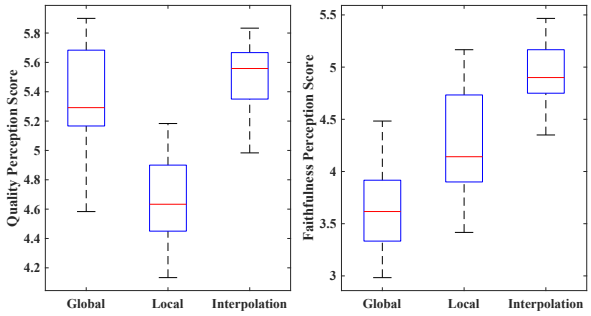

Fig. 16. Box plots of the average quality and faithfulness perception scores over the participants for each method.

further confirmed that our method (mean: 4.85) led to significantly more faithful results than both the global (mean: 3.65; [$t = -29.77$, $p < 0.001$] and local (mean: 4.23; [$t = -16.34$, $p < 0.001$]) retrieval methods. This is consistent with our expectation, since our method provides the largest flexibility to fit to input sketches.

In terms of visual quality our method (mean: 5.50) significantly outperformed the global retrieval method (mean: 5.37; [$t = -3.94$, $p < 0.001$]) and the local retrieval method (mean: 4.68; [$t = -24.60$, $p < 0.001$]). It is surprisingly to see that the quality of results by our method was even higher than the global retrieval method, since we had expected that the visual quality of the results by the global method and ours would be similar. This is possibly because some information is lost after first decomposing an entire sketch into components and then recombining the corresponding feature maps.

4.4 Comparison with Existing Solutions

We compare our method with the state-of-the-art methods for image synthesis conditioned on sketches, including *pix2pix* [19], *pix2pixHD* [38] and *Lines2FacePhoto* [26] and *iSketchNFill* [10] in terms of visual quality of generated faces. We use their released source code but for fair comparisons we train all the networks on our training dataset (Section 3.1). The (input and output) resolution for our method and *pix2pixHD* is 512 × 512, while we have 256 × 256 for *pix2pix* and *Lines2FacePhoto* according to their default setting. In addition, for

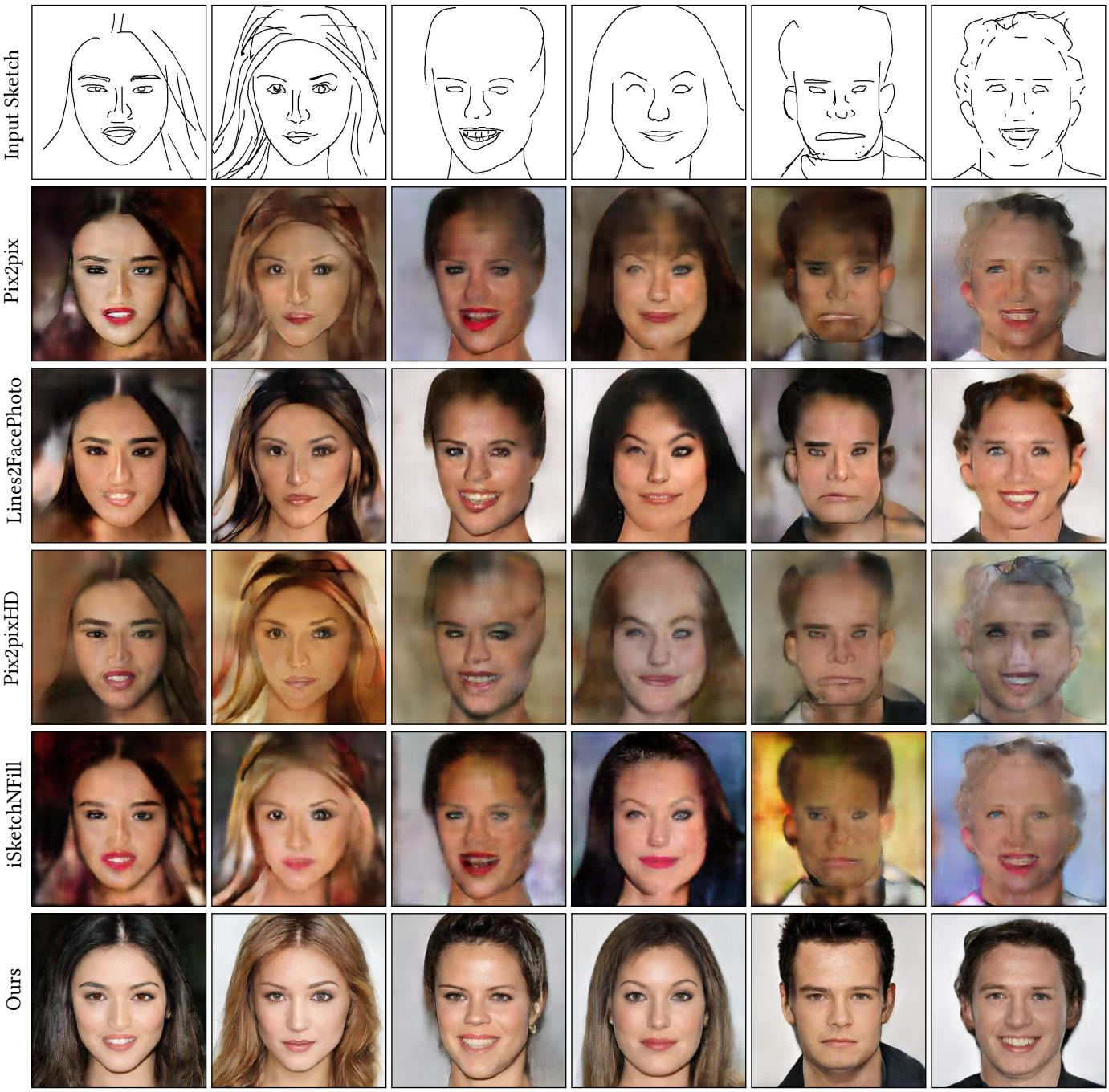

Fig. 17. Comparisons with the state-of-the-art methods given the same input sketches (Top Row).

Lines2FacePhoto, following their original paper, we also convert each sketch to a distance map as input for both training and testing. For *iSketchNFill*, we train their shape completion module before feeding it to *pix2pix* [19] (acting as the appearance synthesis module). The input and output resolutions in their method are 256×256 and 128×128, respectively.

Figure 17 shows representative testing results given the same sketches as input. It can be easily seen that our method produces more realistic synthesized results. Since the input sketches are rough and/or incomplete, they are generally different from the training data, making the compared methods fail to produce realistic faces. Although *Lines2FacePhoto* generates a relatively plausible result

given an incomplete sketch, its ability to handle data imperfections is rather limited. We attempted to perform quantitative evaluation as well. However, none of the assessment metrics we tried, including Fréchet Inception Distance [14] and Inception Score [33], could faithfully reflect visual perception. For example, the averaged values of the Inception Score were 2.59 and 1.82 (the higher, the better) for pix2pixHD and ours, respectively. However, it is easily noticeable that our results are visually better than those by pix2pixHD.

5 APPLICATIONS

Our system can be adapted for various applications. In this section we present two applications: face morphing and face copy-paste.

5.1 Face Morphing

Traditional face morphing algorithms [2] often require a set of keypoint-level correspondence between two face images to guide semantic interpolation. We show a simple but effective morphing approach by 1) decomposing a pair of source and target face sketches in the training dataset into five components (Section 3.2); 2) encoding the component sketches as feature vectors in the corresponding feature spaces; 3) performing linear interpolation between the source and target feature vectors for the corresponding components; 4) finally feeding the interpolated feature vectors to the *FM* and *IS* module to get intermediate face images. Figure 19 shows examples of face morphing using our method. It can be seen that our method leads to smoothly transforming results in identity, expression, and even highlight effects.

5.2 Face Copy-Paste

Traditional copy-paste methods (e.g., [9]) use seamless stitching methods on colored images. However, there will be situations where the hue of local areas is irrelevant. To address this issue, we recombine face components for composing new faces, which can maintain the consistency of the overall color and lighting. Specifically, it can be achieved by first encoding face component sketches (possibly from different subjects) as feature vectors and then combining them as new faces by using the *FM* and *IS* modules. This can be used to either replace components of existing faces with corresponding components from another source, or combining components from multiple persons. Figure 20 presents several synthesized new faces by re-combining eyes, nose, mouth and the remainder region from four source sketches. Our image synthesis sub-network is able to resolve the inconsistencies between face components from different sources in terms of both lighting and shape.

6 CONCLUSION AND DISCUSSIONS

In this paper we have presented a novel deep learning framework for synthesizing realistic face images from rough and/or incomplete freehand sketches. We take a local-to-global approach by first decomposing a sketched face into components, refining its individual components by projecting them to component manifolds defined by the existing component samples in the feature spaces, mapping the refined feature vectors to the feature maps for spatial combination, and finally translating the combined feature maps to realistic images. This approach naturally supports local editing and makes the involved network easy to train from a training dataset of not very large scale. Our approach outperforms existing sketch-to-image synthesis approaches, which often require edge maps or sketches with similar quality as input. Our user study confirmed the usability of our system. We also adapted our system for two applications: face morphing and face copy-paste.

Our current implementation considers individual components rather independently. This provides flexibility (Figure 4) but also introduces possible incompatibility problems. This issue is more obvious for the eyes (Figure 21), which are often symmetric. This might be addressed by introducing a symmetry loss [15] or explicitly requiring two eyes from the same samples (similar to Figure 20).

Our work has focused on refining an input sketch component-by-component. In other words our system is generally able to handle errors within individual components, but is not designed to fix the errors in the layouts of components (Figure 21). To achieve proper layouts, we resort to a shadow-guided interface. In the future, we are interested in modeling spatial relations between facial components and fixing input layout errors.

Our system takes black-and-white rasterized sketches as input and currently does not provide any control of color or texture in synthesized results. In a continuous drawing session, small changes in sketches sometimes might cause abrupt color changes. This might surprise users and is thus not desirable for usability. We believe this can be potentially addressed by introducing a color control mechanism in generation. For example, we might introduce color constraints by either adding them in the input as additional hints or appending them to the latent space as additional guidance. In addition, adding color control is also beneficial for applications such as face morphing and face copy-and-paste.

Like other learning-based approaches, the performance of our system is also dependent on the amount of training data. Although component-level manifolds of faces might be low dimensional, due to the relatively high-dimensional space of our feature vectors, our limited data only provides very sparse sampling of the manifolds. In the future we are interested in increasing the scale of our training data, and aim to model underlying component manifolds more accurately. This will also help our system to handle non-frontal faces, faces with accessories. It is also interesting to increase the diversity of results by adding random noise to the input. Explicitly learning such manifolds and providing intuitive exploration tools in a 2D space would be also interesting to explore.

Our current system is specially designed for faces by making use of the fixed structure of faces. How to adapt our idea to support the synthesis of objects of other categories is an interesting but challenging problem.

ACKNOWLEDGMENTS

This work was supported by Beijing Program for International S&T Cooperation Project (No. Z191100001619003), Royal Society Newton Advanced Fellowship (No. NAF\R2\192151), Youth Innovation Promotion Association CAS, CCF-Tencent Open Fund and Open Project Program of the National Laboratory of Pattern Recognition (NLPR) (No. 201900055). Hongbo Fu was supported by an unrestricted gift from Adobe and grants from the Research Grants

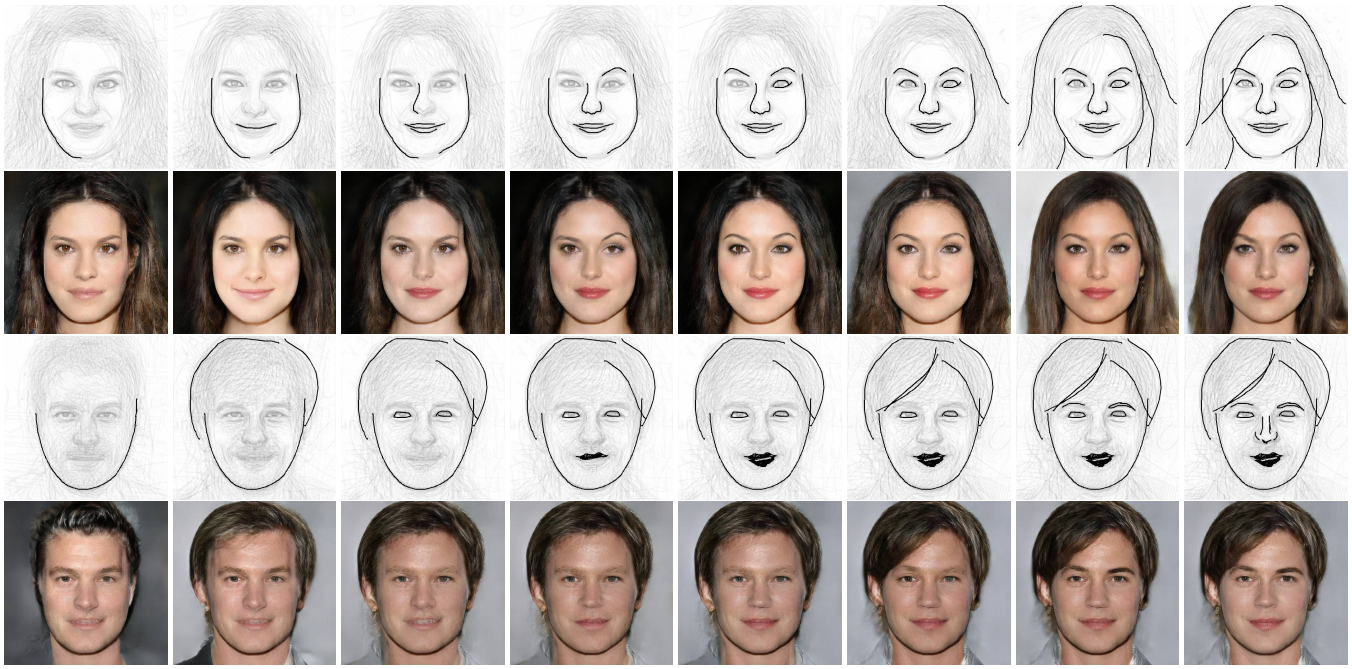

Fig. 18. Two sequences of progressive sketching (under shadow guidance) and synthesis results.

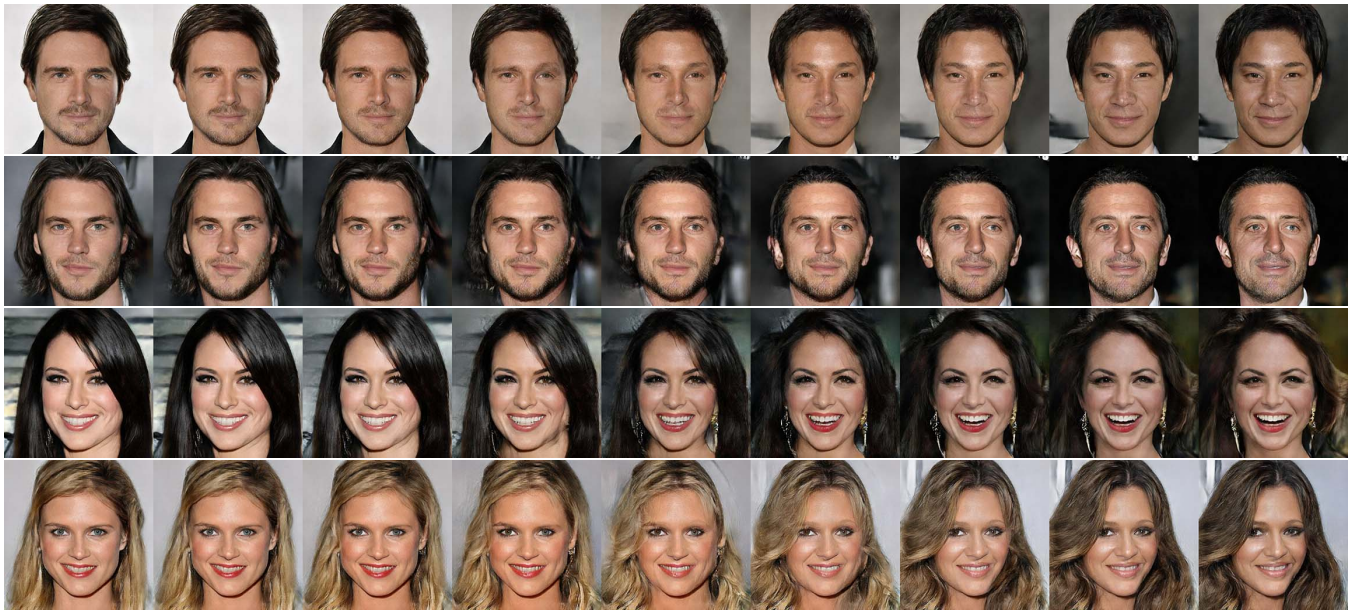

Fig. 19. Examples of face morphing by interpolating the component-level feature vectors of two given face sketches (Leftmost and Rightmost are corresponding synthesized images).

Council of the Hong Kong Special Administrative Region, China (No. CityU 11212119, 11237116), City University of Hong Kong (No. SRG 7005176), and the Centre for Applied Computing and Interactive Media (ACIM) of School of Creative Media, CityU.

REFERENCES

[1] James Arvo and Kevin Novins. 2000. Fluid Sketches: Continuous Recognition and Morphing of Simple Hand-Drawn Shapes. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*. ACM, 73–80.

[2] Martin Bichsel. 1996. Automatic interpolation and recognition of face images by morphing. In *Proceedings of the Second International Conference on Automatic Face*

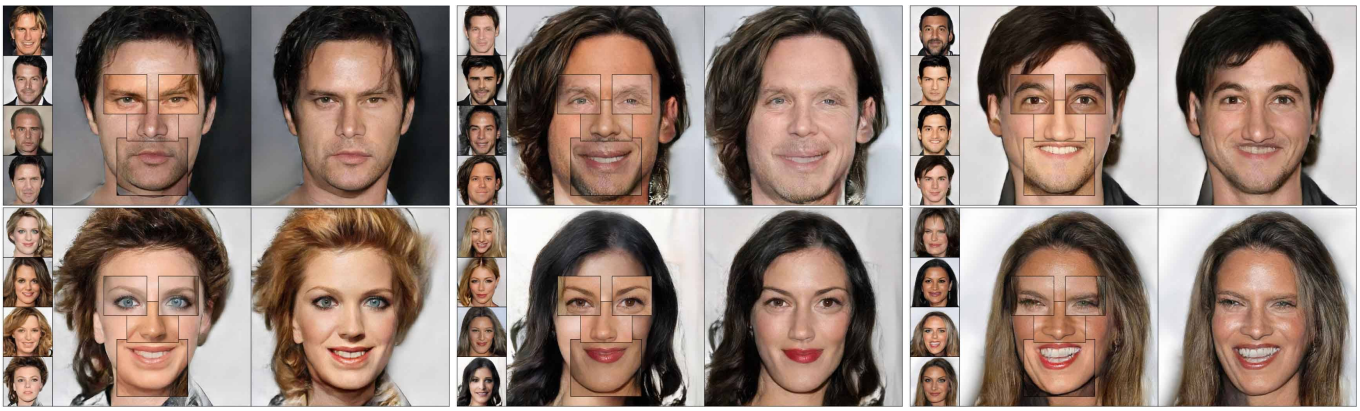

Fig. 20. In each set, we show color image (Left) of the source sketches (not shown here), a new face sketch (Middle) by directly recombining the cropped source sketches in the image domain, and a new face (Right) synthesized by using our method with the recombined sketches of the cropped components (eyes, nose, mouth, and remainder) as input.

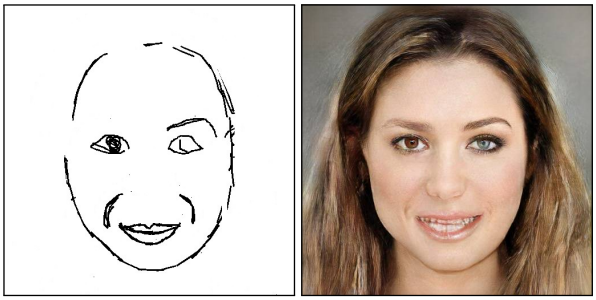

Fig. 21. A less successful example. The eyes in the generated image are of different colors. For the sketched mouth, it is slightly below an expected position, leading to a blurry result for this component.

and Gesture Recognition. IEEE, 128–135.

[3] Volker Blanz and Thomas Vetter. 1999. A Morphable Model for the Synthesis of 3D Faces. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 187âĂŞ194.

[4] John Canny. 1986. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8, 6 (1986), 679–698.

[5] Wengling Chen and James Hays. 2018. Sketchygan: Towards diverse and realistic sketch to image synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 9416–9425.

[6] Tali Dekel, Chuang Gan, Dilip Krishnan, Ce Liu, and William T Freeman. 2018. Sparse, smart contours to represent and edit images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 3511–3520.

[7] Daniel Dixon, Manoj Prasad, and Tracy Hammond. 2010. iCanDraw: using sketch recognition and corrective feedback to assist a user in drawing human faces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 897–906.

[8] Lin Gao, Jie Yang, Tong Wu, Yu-Jie Yuan, Hongbo Fu, Yu-Kun Lai, and Hao(Richard) Zhang. 2019. SDM-NET: Deep Generative Network for Structured Deformable Mesh. *ACM Trans. Graph.* 38, 6 (2019), 243:1–243:15.

[9] Shiming Ge, Xin Jin, Qiting Ye, Zhao Luo, and Qiang Li. 2018. Image editing by object-aware optimal boundary searching and mixed-domain composition. *Computational Visual Media* 4 (01 2018). https://doi.org/10.1007/s41095-017-0102-8

[10] Arnab Ghosh, Richard Zhang, Puneet K Dokania, Oliver Wang, Alexei A Efros, Philip HS Torr, and Eli Shechtman. 2019. Interactive Sketch & Fill: Multiclass Sketch-to-Image Translation. In *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 1171–1180.

[11] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 (NIPSâĂŹ14)*. MIT Press, Cambridge, MA, USA, 2672âĂŞ2680.

[12] Shuyang Gu, Jianmin Bao, Hao Yang, Dong Chen, Fang Wen, and Lu Yuan. 2019. Mask-Guided Portrait Editing with Conditional GANs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 3436–3445.

[13] Xiaoguang Han, Chang Gao, and Yizhou Yu. 2017. DeepSketch2Face: a deep learning based sketching system for 3D face and caricature modeling. *ACM Trans. Graph.* 36, 4, Article Article 126 (2017), 12 pages.

[14] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 6626–6637.

[15] Rui Huang, Shu Zhang, Tianyu Li, and Ran He. 2017. Beyond face rotation: Global and local perception gan for photorealistic and identity preserving frontal view synthesis. In *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2439–2448.

[16] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. 2018. Multimodal unsupervised image-to-image translation. In *European Conference on Computer Vision (ECCV)*. 172–189.

[17] Emmanuel Iarussi, Adrien Bousseau, and Theophanis Tsandilas. 2013. The drawing assistant: Automated drawing guidance and feedback from photographs. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*. ACM, 183âĂŞ192.

[18] Takeo Igarashi, Satoshi Matsuoka, Sachiko Kawachiya, and Hidehiko Tanaka. 1997. Interactive Beautification: A Technique for Rapid Geometric Design. In *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology (UIST âĂŹ97)*. Association for Computing Machinery, 105âĂŞ114.

[19] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1125–1134.

[20] Youngjoo Jo and Jongyoul Park. 2019. SC-FEGAN: Face Editing Generative Adversarial Network with User's Sketch and Color. In *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 1745–1753.

[21] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision (ECCV)*. Springer-Verlag, 694–711.

[22] Tero Karras, Samuli Laine, and Timo Aila. 2019. A style-based generator architecture for generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4401–4410.

[23] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. http://arxiv.org/abs/1412.6980 cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[24] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. 2019. MaskGAN: Towards Diverse and Interactive Facial Image Manipulation. *arXiv preprint arXiv:1907.11922* (2019).

[25] Yong Jae Lee, C Lawrence Zitnick, and Michael F Cohen. 2011. Shadowdraw: real-time user guidance for freehand drawing. *ACM Trans. Graph.* 30, 4, Article Article 27 (2011), 10 pages.

[26] Yuhang Li, Xuejin Chen, Feng Wu, and Zheng-Jun Zha. 2019. LinesToFacePhoto: Face Photo Generation From Lines With Conditional Self-Attention Generative Adversarial Networks. In *Proceedings of the 27th ACM International Conference on Multimedia*. ACM, 2323–2331.

[27] Alex Limpaecher, Nicolas Feltman, Adrien Treuille, and Michael Cohen. 2013. Real-time drawing assistance through crowdsourcing. *ACM Trans. Graph.* 32, 4, Article Article 54 (2013), 8 pages.

[28] Zongguang Lu, Yang Jing, and Qingshan Liu. 2017. Face image retrieval based on shape and texture feature fusion. *Computational Visual Media* 3, 4 (12 2017), 359âĂŞ368. https://doi.org/10.1007/s41095-017-0091-7

[29] Yusuke Matsui, Takaaki Shiratori, and Kiyoharu Aizawa. 2016. DrawFromDrawings: 2D drawing assistance via stroke interpolation with a sketch database. *IEEE Transactions on Visualization and Computer Graphics* 23, 7 (2016), 1852–1862.

[30] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).

[31] Tiziano Portenier, Qiyang Hu, Attila Szabo, Siavash Arjomand Bigdeli, Paolo Favaro, and Matthias Zwicker. 2018. Faceshop: Deep sketch-based face image editing. *ACM Trans. Graph.* 37, 4, Article Article 99 (2018), 13 pages.

[32] Sam T Roweis and Lawrence K Saul. 2000. Nonlinear dimensionality reduction by locally linear embedding. *Science* 290, 5500 (2000), 2323–2326.

[33] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *Advances in neural information processing systems*. Curran Associates, Inc., 2234–2242.

[34] Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. 2017. Scribbler: Controlling deep image synthesis with sketch and color. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 5400–5409.

[35] Edgar Simo-Serra, Satoshi Iizuka, Kazuma Sasaki, and Hiroshi Ishikawa. 2016. Learning to Simplify: Fully Convolutional Networks for Rough Sketch Cleanup. *ACM Trans. Graph.* 35, 4, Article Article 121 (2016), 11 pages.

[36] Qingkun Su, Wing Ho Andy Li, Jue Wang, and Hongbo Fu. 2014. EZ-sketching: three-level optimization for error-tolerant image tracing. *ACM Trans. Graph.* 33, 4, Article Article 54 (2014), 9 pages.

[37] Miao Wang, Guo-Ye Yang, Ruilong Li, Run-Ze Liang, Song-Hai Zhang, Peter M Hall, and Shi-Min Hu. 2019. Example-guided style-consistent image synthesis from semantic labeling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1495–1504.

[38] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2018. High-resolution image synthesis and semantic manipulation with conditional gans. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 8798–8807.

[39] Xiaogang Wang and Xiaoou Tang. 2008. Face photo-sketch synthesis and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31, 11 (2008), 1955–1967.

[40] Di Wu and Qionghai Dai. 2009. Sketch realizing: lifelike portrait synthesis from sketch. In *Proceedings of the 2009 Computer Graphics International Conference*. ACM, 13–20.

[41] Jun Xie, Aaron Hertzmann, Wilmot Li, and Holger Winnemöller. 2014. PortraitSketch: face sketching assistance for novices. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM, 407–417.

[42] Saining Xie and Zhuowen Tu. 2015. Holistically-Nested Edge Detection. In *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 1395–1403.

[43] Ran Yi, Yong-Jin Liu, Yu-Kun Lai, and Paul L Rosin. 2019. APDrawingGAN: Generating Artistic Portrait Drawings from Face Photos with Hierarchical GANs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 10743–10752.

[44] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. 2017. Dualgan: Unsupervised dual learning for image-to-image translation. In *IEEE International Conference on Computer Vision (ICCV)*. 2849–2857.

[45] Sheng You, Ning You, and Minxue Pan. 2019. PI-REC: Progressive Image Reconstruction Network With Edge and Color Domain. *arXiv preprint arXiv:1903.10146* (2019).

[46] Wei Zhang, Xiaogang Wang, and Xiaoou Tang. 2011. Coupled information-theoretic encoding for face photo-sketch recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 513–520.

[47] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. 2016. Generative Visual Manipulation on the Natural Image Manifold. In *European Conference on Computer Vision (ECCV)*.

[48] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*. 2223–2232.