# ICCV 2017 Rebuttal – Face Sketch Synthesis

We thank the reviewers for their detailed comments and suggestions, and we address their main concerns below.

**To R1**

Q1: Technical novelty

Ans: To ~~the best of~~ our knowledge, this paper is the first attempt to transfer styles from multiple images to one target image, as well as application of style transfer in face sketch synthesis. It is very different from previous ~~face sketch synthesis methods~~ (works) which commonly compose a sketch from example patches. As ~~demonstrated~~ shown in Sec 5.1, directly and naïvely applying style transfer to face sketch synthesis simplest does not work. To tackle this problem, we propose transferring local styles from multiple images. However, ~~representing styles from multiple images~~ (this) is a non-trivial task, and, ~~in fact,~~ many of our early attempts failed. The key to success here lies in the introduction of our novel pyramid column feature, which provides a feature-space patch-based approach for fusing local styles from multiple images.

Q2: Boundary discontinuities

Ans: In our implementation, we ~~do not decompose the photos into patches when we compute the style features. Instead, we~~ keep the original feature maps of the training sketches and extract the pyramid column feature based on the patch location (~~as illustrated in~~ (see) Fig. 4). Note that the receptive field of conv5\_1 of VGG is 132 x132, which implies ~~that~~ the receptive fields of neighbouring pyramid column features do overlap ~~with each other~~. In fact, shape boundary is mostly provided by the content image. The discontinuities observed in Fig. 8 is largely caused by the limited size of the training data, which results in an insufficient training of the content network. ~~As a result~~ (Therefore), it may ~~experience~~ (have) difficulties in generating edges for weak boundaries in ~~previously~~ unseen real world photos.

Q3: Normalized Gram matrix as a ~~performance~~ metric

Ans: As pointed out in Sec 5.3, ~~nearly all face sketch synthesis methods can achieve a very high recognition rate. This shows that all methods can generate sketches which “look alike” in terms of key facial components.~~ The main difference between the sketches ~~they produce therefore~~ lies in the subtle details such as texture and shading, which enable people to distinguish synthesized sketches from artist-drawn sketches. Since Gram matrix of CNN features provides a good representation of style in texture synthesis [4], we believe it equally provides a good quantitative measure for texture quality. We do, however, admit that this may not be a fair comparison as our method explicitly minimize the Gram matrix difference in our optimization.

Q4: Difference from [17, 18]

Ans: The key difference between [17, 18] and ours is the synthesis of texture and shading. [18] does not have an explicit texture generation, while [17] employs a feed forward network to generate texture. ~~Different from them,~~ our method is inspired by style transfer, and we propose the pyramid column feature to represent target textures ~~from multiple images~~. Note the L1 loss is only exploited in training the content network. As shown in ~~Figure~~ (Fig.) 10, the content image alone still suffers from over-smoothing. We overcome the ~~over-smoothing~~ problem by adding texture and shading ~~to the content image~~ based on style transfer.

Q5: Motivation of key component loss

Ans: People are more sensitive to key facial components, ~~such as mouth and nose,~~ as they are more discriminative. This observation has been verified and exploited in facial components recognition tasks [i]. We therefore design our model to focus more on these components by imposing an additional loss ~~on these components~~. ~~Figure~~ (Fig.) 9 and ~~Figure~~ (Fig.) 10 show the results of imposing different weight on this loss term.

[i].  *Heisele, Bernd, et al. "Face recognition: component-based versus global approaches."*

**To R2**

Q1: How do we preserve sketch details?

Our method is inspired by the way how artists draw face sketches, and we preserve sketch details through style transfer. As ~~show~~ (shown) in Figure 10, the content image (with \beta\_1 = \beta\_2 = 0) generated by the content network outlines the key facial components, but still suffers from over-smoothing. We introduce the pyramid column feature, which provides a feature-space patch-based approach for fusing local styles from multiple images. ~~Texture and~~ ~~shading (i.e.,~~ sketch details~~)~~ are then added to the content image using style transfer technique based on the pyramid column features. ~~Figure~~ (Fig.) 10 shows the results of style transfer with varying weights on each loss term.

Q2: Limitations.

Ans: In real world photos, the background and boundary discontinuities are not well handled.

Due to the difficulty in collecting a large training dataset, the content network is insufficiently trained, and this may result in boundary discontinuities (see also Q2 for R1). Another limitation is the running time. The whole process now takes around 100s. The patch searching and optimization take most of the time because they run on CPU.

Q3: Portrait size

The difference in portrait sizes is because some methods cannot generate sketches with the same size as the input photos. For example, in FCNN [18], the author did not add padding before convolution to avoid border effects. Hence, the resulting sketches are smaller. We will try to resize these images to make them having the same content scale.

**To R3**

Q1: Complexity and memory

Our network is implemented by Keras with Theano backend. The program runs on a NVIDIA Titan X GPU with 12GB memory and 4.0GHZ Core i7 CPU.

Content network on GPU: ~~We use SGD to train the network, and the learning rate decrease from 0.001 to 0.0001. It takes about 8 hours to converge. After the model is trained, we can generate a content image in 0.4s.~~  (It takes about 8 hours to train the network with SGD and 0.4s per image in the test phase.)

Patch feature searching on CPU: ~~We first precomputed the feature maps of all training photos and store them on the disk.~~ The photo is divided into 324 patches, and it takes around 30s to get the target sketch feature of all patches.

Texture optimization on CPU: ~~We use LBFGS to update the content image. The optimization process takes about 66s.~~ (We use LBFGS to do the texture optimization process and it takes about 66s. )

The whole process takes about 100s, and all the programs above take less than 1GB ~~GPU~~ memory.

Q2: Real world quantitative evaluation

Ans: We agree that it would be better to have real world quantitative evaluation, however, due to the lack of public real world benchmark datasets, we can only use photos under laboratory conditions, which are also used by other methods. For real world photos, since we do not have the corresponding sketches drawn by artists, it will be more difficult for quantitative evaluation.

Q3: Do we need face alignment?

Ans: Yes, the face should be roughly aligned when detecting the corresponding patches. However, we do not need the alignment to be very accurate. Our method is robust to small scale and shift variation. For example, in Fig. 7 and Fig. 8, there are slight scale changes and ~~offsets~~ (misaligntment) between photos, but the generated sketches are still good.