applied sciences

MDPI

Article

Fast Continuous Structural Similarity Patch Based Arbitrary Style Transfer

Bing Wu [1,2,*], Youdong Ding [1] and Qingshuang Dong [2]

[1] Shanghai Film Academy, Shanghai University, Shanghai 200072, China
[2] School of Literature and Media, Taishan University, Tai'an 271021, China
* Correspondence: wb0227@shu.edu.cn

Received: 17 July 2019; Accepted: 8 August 2019; Published: 12 August 2019

Abstract: Style transfer is using a pair of content and style images to synthesize a stylized image which has both the structure of the content image and the style of style image. Existing optimization-based methods are limited in their performance. Some works using a feed-forward network allow arbitrary style transfer but cannot reflect the style. In this paper, we present a fast continuous structural similarity patch based arbitrary style transfer. Firstly, we introduce the structural similarity index (SSIM) to compute the similarity between all of the content and style patches for obtaining their similarity. Then a local style patch choosing procedure is applied to maximize the utilization of all style patches and make the swapped style patch continuous matching with respect to the spatial location of style at the same time. Finally, we apply an efficient trained feed-forward inverse network to obtain the final stylized image. We use more than 80,000 natural images and 120,000 style images to train that feed-forward inverse network. The results show that our method is able to transfer arbitrary style with consistency, and the result comparison stage is made to show the effectiveness and high-quality of our stylized images.

Keywords: style transfer; similarity structure patch; continuity of local style patch; feed-forward network

1. Introduction

Painting is a popular form of art and has many different kinds of styles, e.g., oil painting, watercolor painting, Chinese ink painting, etc. To paint a single piece of art, an artist needs to devote lots of time to finish it. In addition, it is hard to draw a good enough painting for general people and even harder to draw a painting with a given painting style. Therefore, this motivates us to research some efficient methods to automatically generate high-quality artworks.

There are already lots of classical methods to synthesize artistic painting by using the texture synthesis method [1–3]. This kind of method only uses low-level features of images, so it is hard to capture image structures effectively. Recently, Gatys et al. [4] first introduced convolutional neural network (CNN) to synthesize different kinds of painting styles on natural images and achieved impressive visual results. Their work opened up a new field called Neural Style Transfer and lots of works [5–10] have been made by this CNN feature extraction. The success of this method has also introduced to the market the ability to let users transfer their desired style. However, some of the methods adopt optimization process for generating such stylized image, which may need several minutes. This expensive performance can be solved by taking a feed-forward network [11,12]. Although these methods can be used for fast style transfer, they need to retrain a new feed-forward network to adjust a new style.

In order to address the above problems, we firstly extract style and content patches from one layer of CNN, instead of using several different CNN layers. Then, a SSIM style swap module is proposed

to efficiently stylize all of the content patches. In this module, the structural similarity index (SSIM) is introduced to get the structural similarity of style patches to every content patch, and a local style patch choosing method is applied to rebuild high-quality stylized content patches. Finally, thanks to one layer patch extraction we can conveniently use a trained inverse network, or feed-forward network, to construct the stylized image from structural similarity patch efficiently. The structure of our method is shown in Figure 1. The proposed method is not only able to fast synthesise stylized image but also able to build a high-quality stylized image. In addition, several different kinds of effects can be achieved by changing parameters in our flexible method.

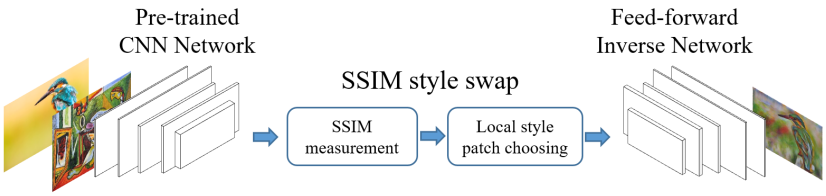


Figure 1. The structure of our procedure.

2. Related Work

Style transfer as optimization. Gatys et al. [4] inspired by the power of CNN. They found content and style features can be extracted by CNN different layers, and those features are capable of combining together to get a stylized image by specifying an optimization problem. In their method, they started from a noise image and divided the optimization problem into content and style loss. The content loss is defined by a squared Euclidean distance between feature representation and the content image. The style loss exploits Gram-based visual texture modeling technique to model the style. This method is limited to preserve the coherence of fine structures and details during stylization. Later, Li et al. [8] derived some different style representations by considering style transfer in the domain of transfer learning, this representation has a good performance in preserving the fine structures and details during stylization. Li and Wand [6] firstly proposed a Markov Random Fields-based Neural style transfer method. They constructed a patch-based loss function, where every synthetic patch needs to match its corresponding nearest neighbor target patch. Then this kind of patch-matching loss function is combined additively with Gatys et al.'s formulation. Above approaches are capable of transferring arbitrary style images, however, their optimization frameworks lower the performance to construct the stylized image.

Feed-forward style network. Another efficient way to reconstruct a stylized image is to use a feed-forward style network after getting features from content and style images. Several methods have been proposed to fast synthesize one or more fixed styles [11–14] by using the feed-forward style network. However, these approaches need to retrain their inverse network when inputting new styles. Chen and Schemidt [7] adopted a normalized cross-correlation measurement to match closest patches between style and content images and introduced a direct style swap procedure to stylize the image. Although this method is much more flexible than previous one-model-to-one or multiple styles, the stylized results are less appealing since the content patches are directly swapped with style patches which are not representative of the desired style. Consequently, the content part is well preserved while the style of the result does not fully reflect the style image. In addition, their method is also limited to tune the style effect for a given content image, since they can only change the size of the patch for the stylization.

In this paper, we introduce the SSIM index to compute the structural similarity between patches. Since non-photorealistic painting generally is not fully the same as its original structure, artists always use some kind of style to similarly represent the content. So we introduce this SSIM to compute the similarity of two patches and replace the structural similar style patch to the corresponding content patch. Although the SSIM is used to predict the perceived quality of pictures, this perception-based model can consider the luminance, contrast and spatial inter-dependencies between two images.

These features are estimated by the average, variance, and co-variance respectively. In addition, the direct style swap procedure in Chen et al. [7] ignores the local style relationship which makes the stylized results not show enough style effect of the original style image. Therefore, we introduce a local style patch choosing method to select good candidates of style patches to well represent the original style.

3. Patch-Based Style Transfer

An image is composed of a huge number of small patches. If we can find a correct patch from style image and replace it in the content image without losing the structure of the content image, we can rebuild the content image with the specified style. Therefore, the main idea of the patch-based operation is constructing a stylized content image patch-by-patch for specified content and style images.

3.1. Structural Similarity Measurement

For the given content image C and style image S, we firstly use a set of pre-trained CNN convolutional layers, like VGG-19 network, to compute activations of content $\Phi(C)$ and style $\Phi(S)$, where $\Phi(\cdot)$ is a function that maps the original image to some intermediate activation space by a pre-trained CNN network. In order to get small patches to present the content and style image, we extract patches from content and style activations row-by-row and column-by-column.

After extracted patches of content and style activations, our final goal is to determine all style patches σ_i^{SSIM} for given input content patches $\sigma_i(C)_{i\in n_c}$ and style patches $\sigma_j(S)_{j\in n_s}$, where n_c and n_s are the number of extracted content and style patches. Instead of using a simple normalized cross-correlation measurement [7] for content and style patches, we introduce the structural similarity index (SSIM) to measure the similarity between two patches. The SSIM for two patches X and Y with size $W \cdot H$ is calculated as follows:

$$\begin{cases} SSIM(X,Y) = \frac{(2\mu_X\mu_Y+C_1)(2\sigma_{XY}+C2)}{(\mu_X^2+\mu_Y^2+C1)(\sigma_X^2+\sigma_Y^2+C2)}, \\ \mu_x = \frac{1}{H\cdot W}\sum_{i=1}^{H}\sum_{j=1}^{W} X(i,j), \\ \sigma_X^2 = \frac{1}{H\cdot W-1}\sum_{i=1}^{H}\sum_{j=1}^{W}(X(i,j)-\mu_X)^2, \\ \sigma_Y^2 = \frac{1}{H\cdot W-1}\sum_{i=1}^{H}\sum_{j=1}^{W}(Y(i,j)-\mu_Y)^2, \\ \sigma_{XY} = \frac{1}{H\cdot W-1}\sum_{i=1}^{H}\sum_{j=1}^{W}((X(i,j)-\mu_X)(Y(i,j)-\mu_Y)). \end{cases} \tag{1}$$

where μ_X and μ_Y are the average value of patch X and Y respectively, σ_X^2 and σ_Y^2 are the variance of X and Y respectively, σ_{XY} is the covariance of X and Y, $C1 = (K_1L)^2$ and $C2 = (K_2L)^2$ are two variables usually used to stabilize the division with weak denominator, $L = 255$ is the dynamic range of the pixel-values, and K_1 and K_2 are two coefficients which respectively set as 0.01 and 0.03 by default.

To have an efficient computation of SSIM, we introduce an auxiliary tensor $T_{a,b,j} = SSIM(\sigma_{a,b}(C), \sigma_j(S))$, where $\sigma_{a,b}(C)$ represents content patch $\sigma_{a:a+s,b:b+s}(C)$ with patch size s, and $T_{a,b,j}$ represents the SSIM value for content patch $\sigma_{a,b}(C)$ measured at style patch j. For the computation of SSIM, the average and variance can be computed by using some simple layers, and the covariance can be computed by using convolution filters for each patch. After building the network for the average, variance and covariance, we can compute SSIM and get T with n_c spatial locations and n_s feature channels.

3.2. Choosing of Local Style Patches

After we computed SSIM for all content and style patches, we had to find proper style patches to their corresponding content patches to represent the final patches σ^{SSIM}. Chen et al. [7] directly used

the maximum value of cross correlation to determine style patches—this method is the main reason that the stylized image does not show enough style of the original style image. In order to address the above problem, we introduce a local style patch choosing method to rebuild the content image, since patches with similar structure are expected to occur at their own neighbor area. Firstly, we want to constrain content patches in local region and only swap to style patches for spatial positions that also locate in local area. Thus, our expected objective is

$$\arg\max \sum_{k=0}^{n_c} \sum_{h=0, h\neq k}^{n_c} g(I_k, I_h) v_k v_h \tag{2}$$

$$g(I_k, I_h) = \frac{1}{\xi\sqrt{2\pi}} e^{-\frac{\|p(I_k)-p(I_h)\|^2}{2\xi^2}} \tag{3}$$

where I_k and I_h are indices of style patches in style activation space, $v_k \in \{SSIM(\sigma_k(C), \sigma_{I_k}(S)) | I_k = 1, ..., n_s\}$ and $v_h \in \{SSIM(\sigma_h(C), \sigma_{I_h}(S)) | I_h = 1, ..., n_s\}$ are respectively SSIM values computed from k-th and h-th content patch to all style patches, as shown in Figure 2. The depth of color for SSIM value represents the value of SSIM, and the deeper the color, the higher the value. The weight function $g(I_k, I_h)$ is a Gaussian function to constrain the spatial location of style patches, where $p(I)$ is a function to convert the index of style patch to its spatial position in style activation, and ξ is a parameter to adjust the range of the region.

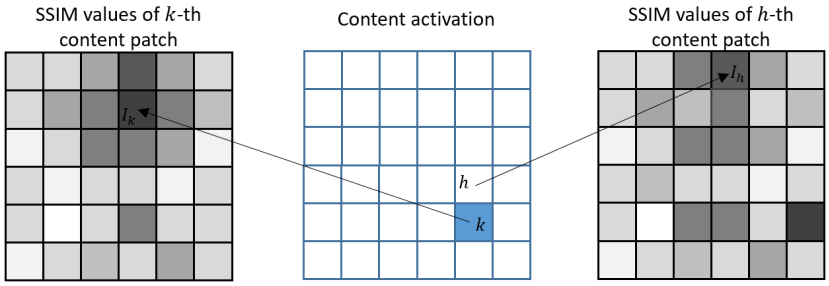


Figure 2. The spatial continuity of style patches in content activation.

However, since there is no analytical gradient for Equation (2), it is hard to efficiently find a global maximum. In order to solve this problem, we proposed a seed propagation method to find the corresponding style patch for every content patch. Although this propagation method only finds a local maximum, it shows good results for the continuity of local style patches. In addition, we can conveniently apply other constraints to this propagation method to get a better result.

As shown in Figure 2, the maximum SSIM value over all elements in T is determined firstly, such that we can find a pair of content patch k and its corresponding style patch I_k according to this SSIM value. Therefore, we can start from this content patch and propagate to all the other patches in the content activation. Without loss of generality, the determined patch k in content activation is going to propagate to an undetermined patch h during the propagation step. Since the style patch I_k has already been chosen as the best style patch for the content patch k, we can test the overall weight for every style patch I_h measured at content patch h by

$$w_{I_h} = \rho_{I_h} \cdot g(I_k, I_h) \cdot v_{k,I_k} \cdot v_{h,I_h} \tag{4}$$

where ρ is a global punitive vector with the size of n_s, which is used to penalize already used style patches. The initial value of every element in this punitive vector is 1.0 and will be penalized by $\rho_p \in (0, 1)$ at I if we reuse the same I-th style patch. This penalty item is used to avoid too much reuse of the same style patch which will significantly influence the final style results. Finally, we can directly select the largest weight as the best style patch for the content patch h. We follow the above criterion to propagate to all content patches in the content activation.

After we obtained all indices of style patches I_i for corresponding content patches i in content activation, another auxiliary tensor $T'_{a,b}$, like the tensor $T_{a,b}$, is introduced to fast convert the content activation to style patches with best matching, and it builds by

$$T'_{a,b,j} = \begin{cases} 1, & \text{if } j = I_i; \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Then, we adopt a backward convolution, or transposed convolution, to construct σ^{SSIM} by using T' as input and style activation patches $\sigma_j(S)$ as filters. In addition, since backward convolution will sum up the values from overlapping patches, we need to add an element-wise division for every spatial location after that backward convolution by the number of overlapping patches to average those values.

4. Inverse Network

Some prior methods of style transfer, like Gatys et al. [4], use a squared-error loss function on the activation space with target activation σ_i^{SSIM} to optimize the stylized image. However, this kind of method needs a lot of time to solve an optimization problem and might be too expensive when applying video stylization. Therefore, we build an inverse network and train it such that we can use the target activation σ_i^{SSIM} to restore its optimal corresponding stylized image. In particular, we need to train this network to have a versatile ability to adopt multiple kinds of content and style images.

Since we want our target activation to restore to its corresponding stylized image, the theory is almost the same as previous squared-error loss function. We define our inverse function as follows:

$$\arg\inf_f \ \mathbb{E}_H \|\Phi(f(H)) - H\|_F^2 + \lambda \ell_{TV}(f(H)) \tag{6}$$

where f represents a deterministic function which is the operation by inverse neural network, H is random activation features, $\|\cdot\|$ is the Frobenius norm, and $\ell_{TV}(\cdot)$ is the total variation regularization term which is usually used in image construction method [11,15,16]. This regularization acts as a natural image prior to spatially smoother results for the re-upsampled image. The total variation regularization for the synthesized image with dimension $h \cdot w \cdot d$ is computed as following:

$$\ell_{TV}(I) = \sum_{i=1}^{h-1} \sum_{j=1}^{w} \sum_{k=1}^{d} (I_{i+1,j,k} - I_{i,j,k})^2 + \sum_{i=1}^{h} \sum_{j=1}^{w-1} \sum_{k=1}^{d} (I_{i,j+1,k} - I_{i,j,k})^2 \tag{7}$$

Training the Inverse Network

Since the pre-trained CNN layers, including convolutional, maxpooling and ReLU layers, define the forward function $\Phi(\cdot)$ to compute the activations, these functions can map multiple different inputs into the same one, so it is non-injective when we inverse from the result. However, there are already many works that have adopted inverse network [17–19] for this kind of inverse mapping. We then train a parametric neural network by idea in Equation (6), then we set the training function as:

$$\min_\nu \frac{1}{n} \sum_1^n \|\Phi(f(H_i;\nu)) - H_i\|_F^2 + \lambda \ell_{TV}(f(H_i;\nu)) \tag{8}$$

where ν denotes the parameters of the inverse network, H_i is i-th training feature from datasets with size n. As mentioned before, we adopted a set of pre-trained CNN convolutional layers to compute content and style activations. Operations by these layers are defined as $\Phi(\cdot)$. After the structural similarity patch swapping, we use its output as the input of the feed-forward network to get the final stylized image. Therefore, we also need to design a similar inverse version of CNN structure to inverse the target activations. In this paper, the pre-trained CNN network and the corresponding inverse

network are shown in Figure 3. The first row is the dimension of activations, the second row from left to right is the pre-trained CNN network and the third row from right to left is the corresponding inverse network.

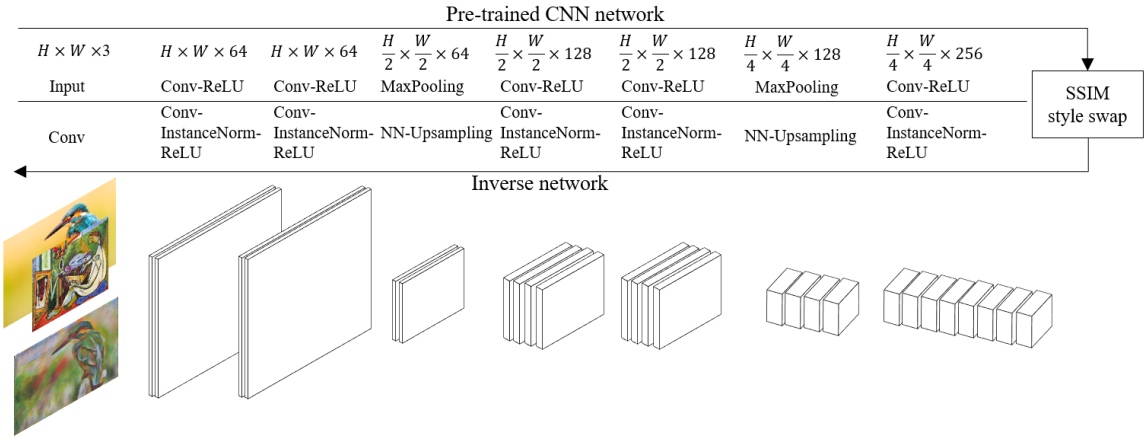


Figure 3. Pre-trained CNN and inverse network structure.

Once the inverse network has been trained, we then can use this trained network to inverse our swapped activation σ_i^{SSIM} to get the stylized image.

5. Experiments

In this section, we explain the datasets used for training the inverse network. We show that our method can transfer arbitrary content and style images, and our results can be tuned to achieve different effects. At the same time, the comparison of results is presented to show the effectiveness and high-quality of our method.

5.1. Training of Inverse Network

For obtaining activation patches from the pre-trained network, here we adopt the truncated network from input layer up to layer "relu3_1" in pre-trained VGG-19 network [20]. Then, the inverse network is determined by this pre-trained truncated VGG-19 network, which will compute an approximation inverse result of VGG-19 network (see Figure 3).

We use the Microsoft COCO dataset [21] and a dataset of paintings sourced from wikiart.org and hosted by Kaggle [22]. These two datasets are both roughly 80,000 natural and painting images respectively. However, the painting dataset does not contain enough painting style, like Chinese ink painting, which will not produce the correct style when we use Chinese ink painting as style input. Therefore, we additionally extend about 30,000 Chinese ink painting images into the training dataset to make our method fit more different styles. Finally, we combine those datasets for training such that the inverse network is capable of reconstructing the structure and pattern of content and style images. Figures 4 and 5 show results compared with and without ink painting training dataset. Results by using the model trained by ink painting dataset, with parameter $\xi = 3.0$, and $\rho_p = 0.5$, are better than results only trained by the Kaggle style dataset.

(**a**) Content image

(**b**) Style image

(**c**) Training without ink painting dataset

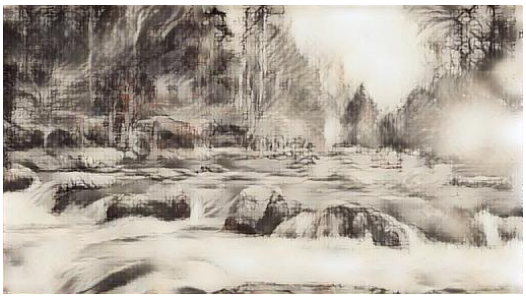
(**d**) Training with ink painting dataset

Figure 4. Comparison results with or without ink painting training dataset.

(**a**) Content image

(**b**) Style image

(**c**) Training without ink painting dataset



(**d**) Training with ink painting dataset

Figure 5. Comparison results with or without ink painting training dataset.

5.2. Style Transfer Results

(1) Consistency. The computed index in the SSIM style swap module is a deterministic value for every patch between given fixed content and style image. Moreover, the inverse network is also a deterministic function after training. Therefore, the transferred results would be the same with random initialization. In Figure 6a,b are results from Gatys et al. [4], Figure 6c,d are results by our method. The visual difference between Figure 6a,b is very obvious while the root-mean-square error (RMSE)

between Figure 6c,d is 0. Therefore, our method can directly apply to video style transfer and does not need to introduce optical-flow to constrain the coherence between frames.

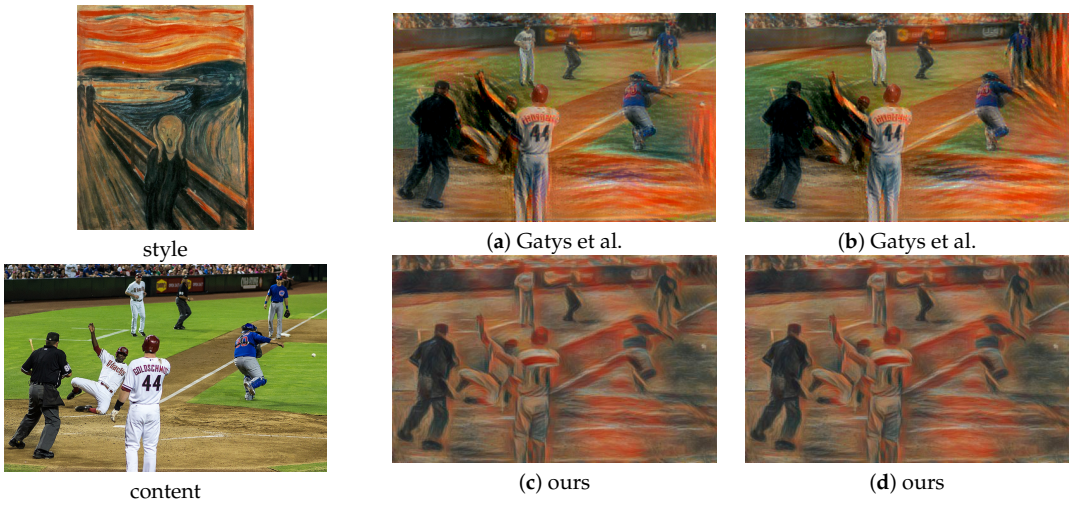
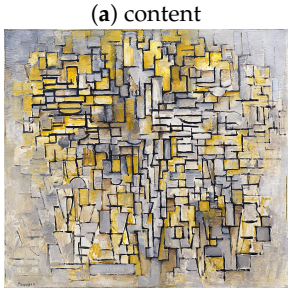


Figure 6. Results from random initialization.

(2) Style results tuning by our local style patch choosing method. The local style choosing method uses a Gaussian weight function to weight local style patches by the spatial distance of style patches and a penalty item to maximize utilization of all style patches in the style activation. In the following part, we will set several combinations of parameters and make comparisons with the previous method to show our method can make a higher quality result.

In Figure 7, we stylize Figure 7a the content image of "golden gate" with Figure 7b a style image of "composition in brown and gray". The result generated from Chen et al. [7] is shown in Figure 7c with the patch size of 3. Figure 7b shows a direct swap result, which means content and style patches have been directly swapped by their corresponding maximum SSIM values. Neither result has shown enough style of original style image. Figure 7e–h are respectively results by our local style patch choosing method with enabled and disabled the penalty item under different values of ξ. From this comparison, we can see that our method can represent a much better stylization of original style image than Chen et al. [7] and direct swap when we set a relatively lower value of ξ. Since the value of ξ can be treated as a parameter to control the range of choosing local style patches, a lower value can capture local style to make a better stylized result. In addition, the penalty item utilizes style patches in a maximum way such that the style will tend to the original style image, however, the method of Chen et al. [7] and direct swap may reuse the same style patches lots of times for different content patches. Therefore, our local style patch choosing method can stylize for a more impressive result with complex details. We should mention that in this experiment, our method uses the same penalty item with $\rho_p = 0.5$ and patch size of 3.

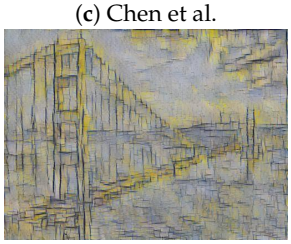
(**a**) content



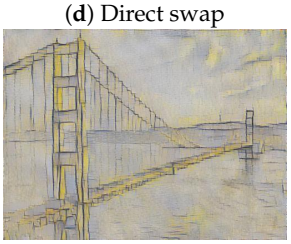
(**b**) style

(**c**) Chen et al.

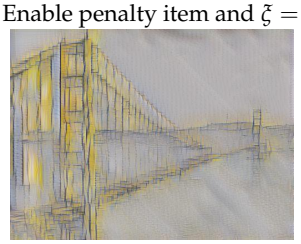
(**d**) Direct swap



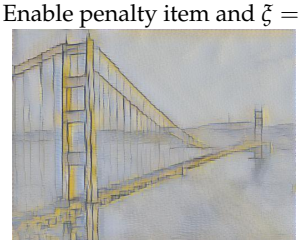
(**e**) Enable penalty item and $\xi = 1.0$



(**f**) Enable penalty item and $\xi = 3.0$



(**g**) Disable penalty item and $\xi = 1.0$



(**h**) Disable penalty item and $\xi = 3.0$

Figure 7. Results comparison for golden gate.

One natural way to adjust the degree of stylization is to use different patch sizes. Each column of Figure 8 shows the transferred results by different patch sizes, which includes 3×3, 5×5, and 7×7. Results of the first row are generated by the method of Chen et al. [7], results of the second row are computed by the direct swap, and results of the third and fourth row are respectively generated with our local style patch choosing method under $\rho_p = 0.2$ and $\rho_p = 0.5$. From results, we can see that the small detail structure from the content image has lost and the stylized results become smoother as the patch size increases. In results of Chen et al. [7] and direct swap, we can see there exists a clear boundary artifact at the boundary of the character. However, our method does not have this problem because our local style patch choosing method requires continuity of local style patches. In addition, our results show more different colors than others when representing a stylized result due to the penalty item.

In the method of Chen et al. [7], they can only change the size of the patch to tune the effect of style, so it is very limited to changing the effect of the stylized image. However, we also can obtain big different style effects by changing the ξ in Gaussian weight function, which will influence the search region for local style patches. In Figure 9, results of the first column are computed by the method of Chen et al. [7] under the different size of the patch, and the other images are the stylized results from our method with different patch size and ξ. From results, we can see that the style effect changing by patch size from Chen et al. [7] is limited. However, our method makes it easy to tune different kinds of effects. In our results, when we set a lower value of ξ, the stylized result becomes more abstract. This is because our method uses a maximum value from all SSIM values as the initial seed for the propagation. If a small value of ξ is used, a good enough candidate of style patch for a content patch may not exist at that range when we propagate to its neighbor. However, with a higher value of ξ, the quality of the stylized result is better than the previous method. For example, results from our method contain the ample detail of spoondrift like in the style image, and the effect of the sky is more impressive.

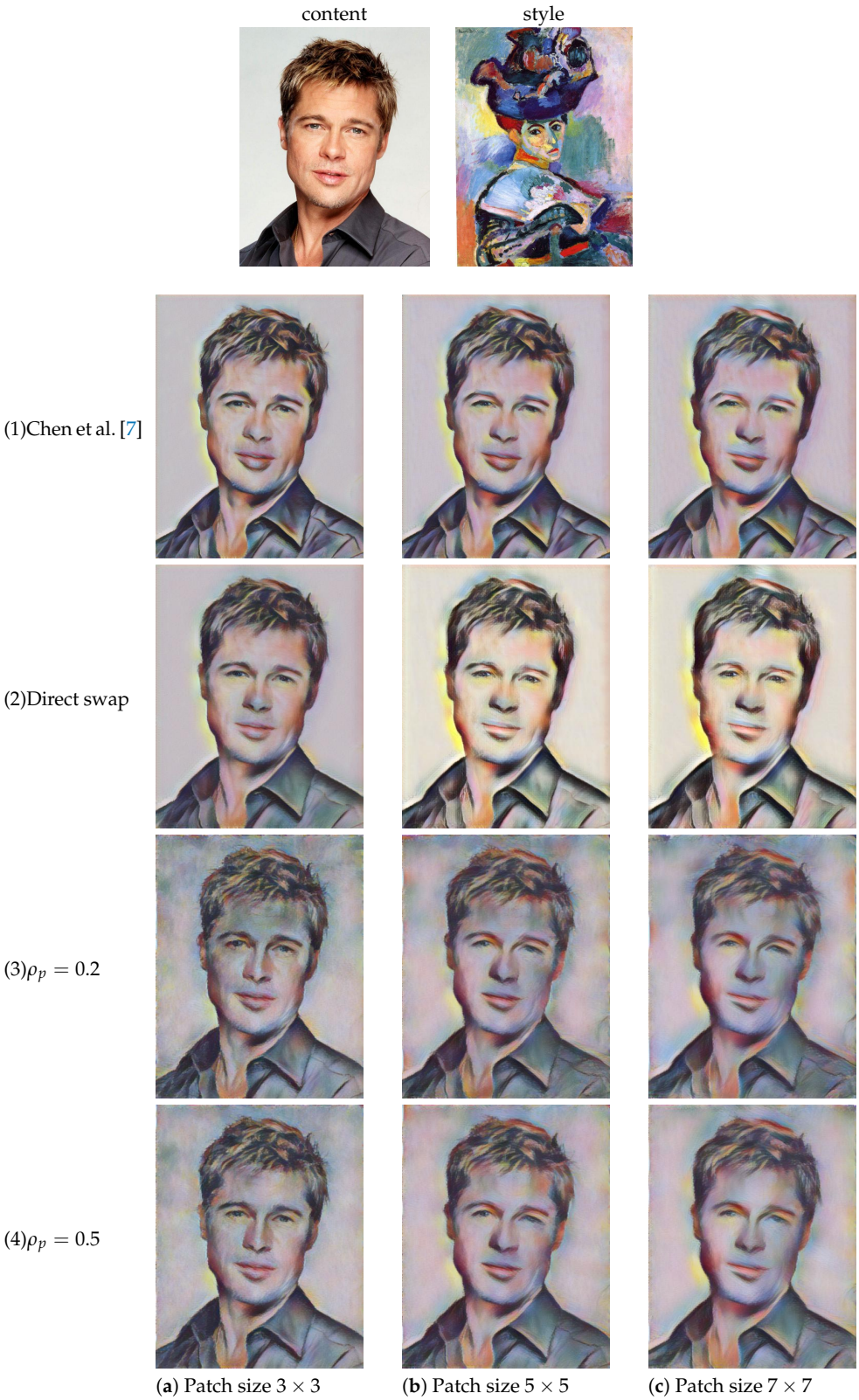
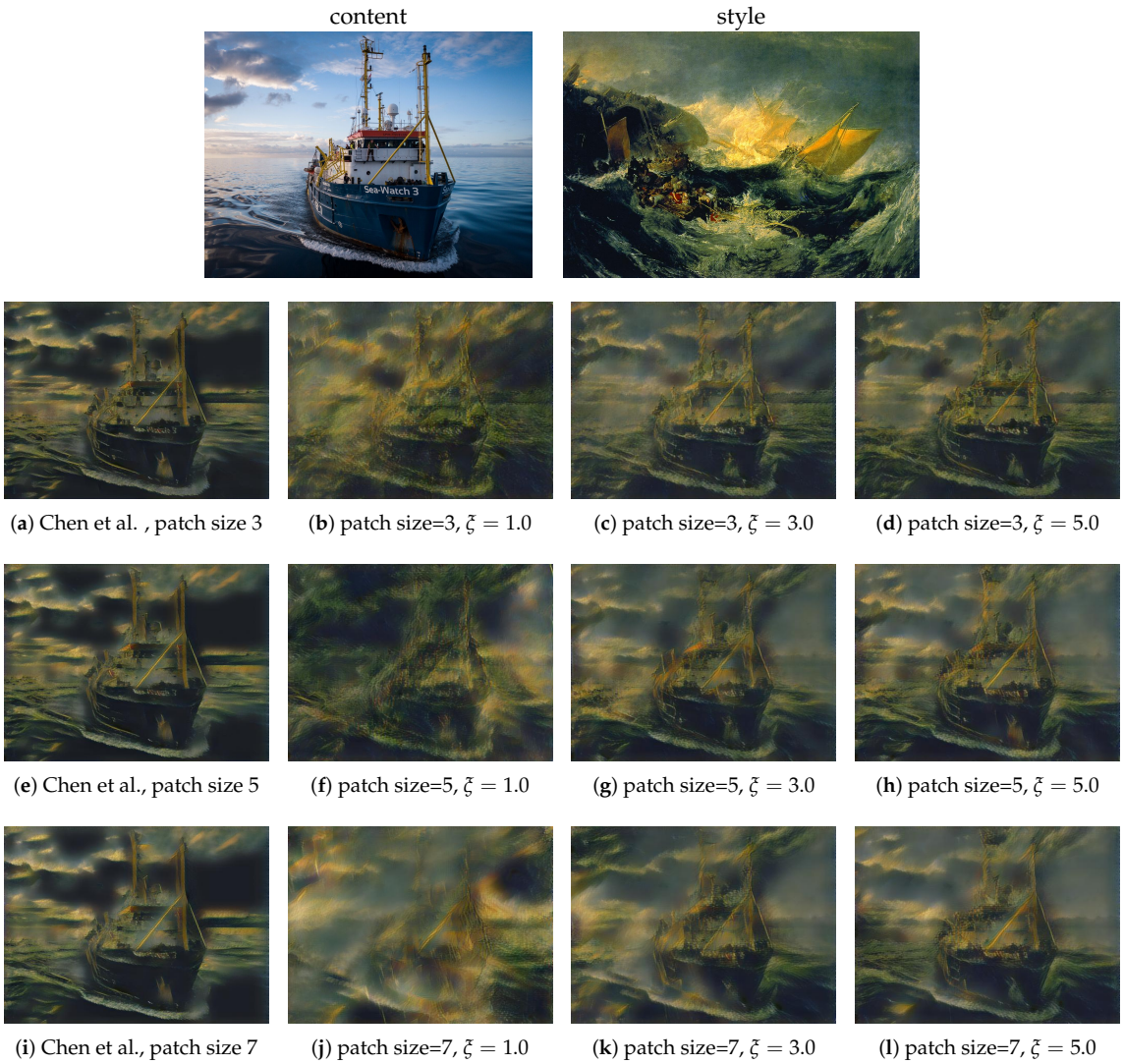


Figure 8. Results comparison with Chen et al., direct swap and our method under different patch size.



(**a**) Chen et al. , patch size 3 (**b**) patch size=3, $\xi = 1.0$ (**c**) patch size=3, $\xi = 3.0$ (**d**) patch size=3, $\xi = 5.0$

(**e**) Chen et al., patch size 5 (**f**) patch size=5, $\xi = 1.0$ (**g**) patch size=5, $\xi = 3.0$ (**h**) patch size=5, $\xi = 5.0$

(**i**) Chen et al., patch size 7 (**j**) patch size=7, $\xi = 1.0$ (**k**) patch size=7, $\xi = 3.0$ (**l**) patch size=7, $\xi = 5.0$

Figure 9. Style tuning by changing parameters of patch size and ξ.

(3) Comparison of several stylized images. We compare our results with Chen et al. [7] and Gatys et al. [4] in Figure 10 with two different styles. In Figure 10, two style images are shown at the top. The first and second rows are results for (a)-style, and third and fourth rows are results for (b)-style. During this comparison, we set our parameter $\xi = 5.0$, $\rho_p = 0.5$, and patch size of 3. Results from Chen et al. [7] contain too many details of the original content image by using cross-correlation measurement and do not fully reflect the effect of style image by their direct style swap. Unlike their method, our SSIM is a kind of statistical way to measure the similarity of patches, so it can ignore those too delicate details. In addition, our local style patch choosing method is more likely to show the style effect of the original style image. For results from Gatys et al. [4], their results sometimes make it hard to preserve the coherence of fine structures and details during stylization. For style comparison, the style patterns from our stylized results are close to the style pattern of results from Gatys et al. [4], however, our style is much better and smoother than their style. Therefore, our method can stylize input content and style images with higher quality than other methods from this comparison.

(**a**) La Muse, Pablo Picasso (**b**) Self-Portrait, Pablo Picasso

Content image ours method Chen et al. Gatys et al.

Figure 10. Comparisons.

(4) Comparison of computational cost. The performance comparison for Gatys et al. [4], Chen et al. [7] and ours is listed in Table 1. The computational time is tested on a GeForce GTX 1080 machine. The input content and style images are both with resolution 300×300, and the time starts recording from the very beginning of style transfer for every method. In our method, the computation of the SSIM index for content-style patches can be achieved in parallel in GPU, and the propagation stage in the local style choosing method can execute in parallel in CPU. Although our method is a little bit slower than the method of Chen et al. [7], we can achieve better stylized results and flexibly tune results to our desired effect.

Table 1. Comparison of average computation time.

Methods	Num. Iter.	Total Time(s)
Gatys et al. [4]	500	66.32
Chen et al. [7]	1	13.16
ours	1	4.57

6. Conclusions

In this paper, we present a fast structural similarity patch based style transfer method. Firstly, the content and style images are passed into a pre-trained CNN network to compute their feature activations. Then, the extraction procedure is applied to both content and style activations to get their corresponding small patches for the following style swap. The similarity structural index (SSIM) is introduced to calculate the SSIM value between all content patches and all style patches. After we computed the SSIM value between content and style patches, we introduce a local style patch choosing method to well represent our stylized image, which constrains the selection of local style and automatically maximize the utilization of all style patches. Finally, determined style patches are used to synthesize the final stylized image by a feed-forward network. Results show our stylization model is able to efficiently stylize arbitrary content and style images, and our presented flexible method is able to easily tune several more impressive effects than previous methods.

There are still some limitations for our method, such as, ignoring the strokes of the style image and lacking the global style measurement. In the future, we will continue to improve it and consider more sophisticated factors for the stylization of paintings.

Author Contributions: Conceptualization, B.W., Q.D. and Y.D.; methodology, B.W. and Q.D.; software, B.W.; validation, B.W. and Q.D.; formal analysis, Y.D.; investigation, Q.D.; resources, W.B.; data curation, W.B.; writing—original draft preparation, W.B.; writing—review and editing, W.B.; visualization, Q.D.; supervision, Y.D.; project administration, W.B.; funding acquisition, W.B.

Funding: This research was funded by Natural Science Foundation of Shanghai grant number 19ZR1419100, The Ministry of education of Humanities and Social Science project grant number 16YJC760056,

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kwatra, V.; Essa, I.; Bobick, A.; Kwatra, N. Texture optimization for example-based synthesis. *ACM Trans. Graph. (ToG)* **2005**, *24*, 795–802. [CrossRef]
2. Rosin, P.; Collomosse, J. *Image and Video-Based Artistic Stylisation*; Springer Science & Business Media: Berlin, Germany, 2012; Volume 42.
3. Elad, M.; Milanfar, P. Style transfer via texture synthesis. *IEEE Trans. Image Process.* **2017**, *26*, 2338–2351. [CrossRef] [PubMed]
4. Gatys, L.A.; Ecker, A.S.; Bethge, M. A neural algorithm of artistic style. *arXiv* **2015**, arXiv:1508.06576.
5. Frigo, O.; Sabater, N.; Delon, J.; Hellier, P. Split and match: Example-based adaptive patch sampling for unsupervised style transfer. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 553–561.
6. Li, C.; Wand, M. Combining markov random fields and convolutional neural networks for image synthesis. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2479–2486.
7. Chen, T.Q.; Schmidt, M. Fast patch-based style transfer of arbitrary style. *arXiv* **2016**, arXiv:1612.04337.
8. Li, Y.; Wang, N.; Liu, J.; Hou, X. Demystifying neural style transfer. *arXiv* **2017**, arXiv:1701.01036.
9. Risser, E.; Wilmot, P.; Barnes, C. Stable and controllable neural texture synthesis and style transfer using histogram losses. *arXiv* **2017**, arXiv:1701.08893.
10. Li, Y.; Fang, C.; Yang, J.; Wang, Z.; Lu, X.; Yang, M.H. Universal style transfer via feature transforms. *Adv. Neural Inf. Process. Syst.* **2017** , 386–396.
11. Johnson, J.; Alahi, A.; Fei-Fei, L. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*; Springer: Berlin, Germany, 2016; pp. 694–711.
12. Dumoulin, V.; Shlens, J.; Kudlur, M. A learned representation for artistic style. *arXiv* **2016**, arXiv:1610.07629.
13. Ulyanov, D.; Lebedev, V.; Vedaldi, A.; Lempitsky, V.S. Texture Networks: Feed-forward Synthesis of Textures and Stylized Images. *ICML* **2016**, *1*, 4.
14. Ulyanov, D.; Vedaldi, A.; Lempitsky, V. Instance normalization: The missing ingredient for fast stylization. *arXiv* **2016**, arXiv:1607.08022.

15. Aly, H.A.; Dubois, E. Image up-sampling using total-variation regularization with a new observation model. *IEEE Trans. Image Process.* **2005**, *14*, 1647–1659. [CrossRef]
16. Mahendran, A.; Vedaldi, A. Understanding deep image representations by inverting them. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 5188–5196.
17. Zeiler, M.D.; Krishnan, D.; Taylor, G.W.; Fergus, R. Deconvolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 13–18 June 2010; pp. 2528–2535.
18. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3431–3440.
19. Dosovitskiy, A.; Springenberg, J.T.; Tatarchenko, M.; Brox, T. Learning to generate chairs, tables and cars with convolutional networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 692–705. [CrossRef]
20. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556 .
21. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft Coco: Common Objects in Context. In *European Conference on Computer Vision*; Springer: Berlin, Germany, 2014; pp. 740–755.
22. Duck, S.Y. Painter by Numbers. 2019. Available online: wikiart.org (accessed on 21 January 2019).