# A Closed-form Solution to Photorealistic Image Stylization

Yijun Li[1], Ming-Yu Liu[2], Xueting Li[1], Ming-Hsuan Yang[1,2], and Jan Kautz[2]
[1]University of California, Merced [2]NVIDIA
{yli62,xli75,mhyang}@ucmerced.edu {mingyul,jkautz}@nvidia.com

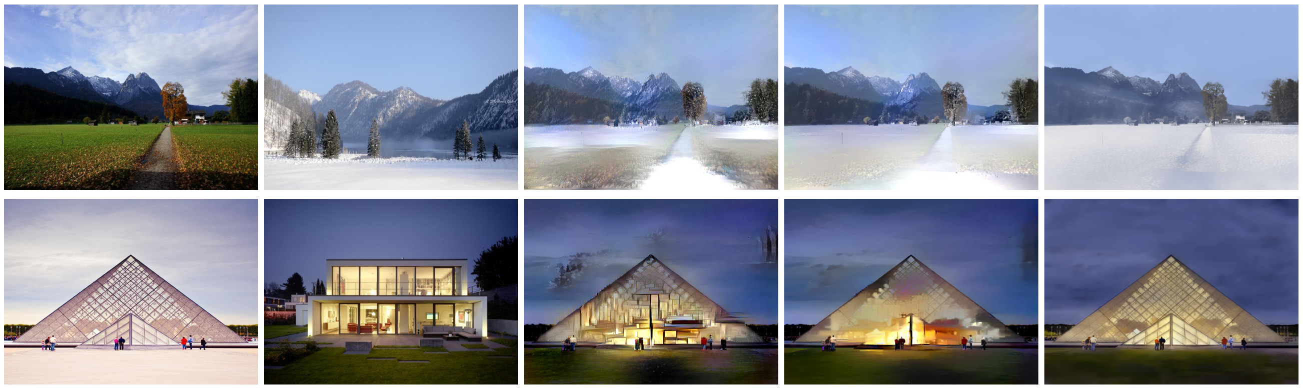

(a) Content (b) Style (c) Gatys et al. [6] (d) Luan et al. [21] (e) Ours

Figure 1: Given a content photo (a) and a style photo (b), photorealistic image style transfer algorithms aim at transferring the style of the style photo to the content photo as shown in (c), (d), and (e). Comparing to the existing algorithms [6, 21], the proposed algorithm outputs photos that have more consistent stylizations and much fewer artifacts. Moreover, it runs an order of magnitude faster thanks to its closed-form formulation.

## Abstract

*Photorealistic image style transfer algorithms aim at stylizing a content photo using the style of a reference photo with the constraint that the stylized photo should remains photorealistic. While several methods exist for this task, they tend to generate spatially inconsistent stylizations with noticeable artifacts. In addition, these methods are computationally expensive, requiring several minutes to stylize a VGA photo. In this paper, we present a novel algorithm to address the limitations. The proposed algorithm consists of a stylization step and a smoothing step. While the stylization step transfers the style of the reference photo to the content photo, the smoothing step encourages spatially consistent stylizations. Unlike existing algorithms that require iterative optimization, both steps in our algorithm have closed-form solutions. Experimental results show that the stylized photos generated by our algorithm are twice more preferred by human subjects in average. Moreover, our method runs 60 times faster than the state-of-the-art approach. Code and additional results are available at* https://github.com/NVIDIA/FastPhotoStyle.

## 1. Introduction

The goal of photorealistic image style transfer is to change the style of a photo to resemble that of another one. For a faithful stylization, the content in the output photo should remain the same, while the style of the output photo should resemble the one of the reference photo. Furthermore, the output photo should look like a real photo captured by a camera. Figure 1 shows two photorealistic image stylization examples. In one example, we transfer a summery photo to a snowy one, while in the other example, we transfer a day-time photo to a night-time photo. Photorealistic image style transfer algorithms find use in numerous applications, ranging from image editing to content creation.

Due to lack of expressive feature representations, classical stylization approaches are based on matching color statistics (e.g., color transfer [25, 24, 31] or tone transfer [1]) or are limited to specific scenarios (e.g., seasons [12] and headshot portraits [27]). Recently, Gatys et al. [5, 6] show that the correlations between deep features encode the visual style of an image and propose an optimization-based method, called the neural style transfer algorithm, for image style transfer. While the method shows impressive performance for *artistic* style transfer (converting images to paintings), it often introduces structural artifacts and distortions (e.g., extremely bright colors) when applied to the photorealistic image style transfer task as shown in Figure 1(c). In a follow-up work, Luan et al. [21] propose adding a regularization term to the optimization objective function of the neural style transfer algorithm and show this reduces distortions in the output images. However, the resulting algorithm tends to stylize semantically uniform regions in images inconsistently as shown in Figure 1(d).

In this paper, we propose a novel fast photorealistic image style transfer algorithm. It consists of two steps: the stylization step and the smoothing step. Both of the steps have closed-form solutions. The stylization step is based on the whitening and coloring transform (WCT) algorithm [17] and is referred to as the PhotoWCT step. The WCT algorithm matches deep feature statistics via feature projections and is developed for *artistic* style transfer. Similar to the neural style transfer algorithm, when the WCT algorithm is applied to the photorealistic image style transfer task, the output stylized photos often have structural artifacts. The proposed PhotoWCT step addresses the issue by incorporating unpooling layers in the WCT transform. We show this largely improves the photorealistic style transfer performance. The PhotoWCT step alone cannot guarantee generating spatially consistent stylization. This is resolved by the proposed smoothing step, which is formulated as a manifold ranking problem. We conduct extensive experiments with comparison to the state-of-the-art to validate the proposed algorithm. User studies on stylization effects and photorealism show the competitive advantages of the proposed algorithm. In addition, our algorithm runs 60 times faster across various image resolutions thanks to the closed-form formulation.

## 2. Related Works

Stylizing an image while keeping the output photorealistic is a long-standing problem. Existing stylization methods are mostly example-based and can be classified into two categories: global and local. Global methods usually construct a spatially-invariant transfer function through matching the means and variances of pixel colors [25], histograms of pixel colors [24], or both [4]. These approaches often only adjust global colors or tones [1] in effects. Local methods [28, 27, 36, 12, 33] perform spatial color mapping through finding dense correspondences between the content and style photos based on either low-level or high-level features. These approaches are slow in practice. Moreover, they are often limited to specific scenarios.

In a seminar work, Gatys et al. [5, 6] propose the neural style transfer algorithm for the *artistic* style transfer task. The core of the algorithm is to solve an optimization problem of matching the Gram matrices of deep features extracted from the content and style photos. A number of methods have been developed [14, 34, 11, 16, 3, 8, 9, 17] to further improve its stylization performance and speed. However, these works do not aim for preserving photorealism (see Figure 1(c)). Post-processing techniques [15, 22] are proposed to refine these results by matching the gradients between the content and output photo.

Another important line of related works is image-to-image translation [10, 35, 20, 32, 29, 19, 43] where the goal is to translate an image from one domain to another. Unlike image-to-image translation, photorealistic image style transfer does not require a training dataset to learn the translation. It just needs one single reference image. Furthermore, photorealistic image style transfer performs can make image translation more specific. Not only can it transfer a photo to a different domain (e.g., form day to night-time) but also can transfer the specific style (e.g., extent of darkness) in a reference style image.

Closest to our work is the method of Luan et al. [21], which significantly improves photorealism of the stylization results of the neural style transfer algorithm by enforcing local constraints as an additional loss function. However, it often generates noticeable artifacts, which cause inconsistent stylization (Figure 1(d)). Moreover, it is computationally expensive. The proposed algorithm aims at efficient and effective photorealistic image style transfer. We demonstrate that it performs favorably against state-of-the-art methods in terms of both quality and speed.

## 3. Photorealistic Image Stylization

Our photorealistic image style transfer algorithm consists of two steps as illustrated in Figure 2. The first step is a stylization transform $\mathcal{F}_1$ called PhotoWCT. Given a style photo $I_S$, $\mathcal{F}_1$ transfer the style of $I_S$ to the content photo $I_C$ while minimizing structural artifacts in the output image. Although $\mathcal{F}_1$ can faithfully stylize $I_C$, it often leads to inconsistent stylization in semantically similar regions. Therefore, we use a photorealistic smoothing function $\mathcal{F}_2$, to eliminate these artifacts. Our whole algorithm can be written as a two-step mapping function:

$$\mathcal{F}_2\Big(\mathcal{F}_1(I_C, I_S), I_C\Big). \tag{1}$$

In the following, we discuss the two steps in details.

### 3.1. Stylization

Our PhotoWCT is based on the WCT [17]. It improves the WCT for the photorealistic image style transfer task by using a novel network design. For completeness, we briefly review the WCT in this section.

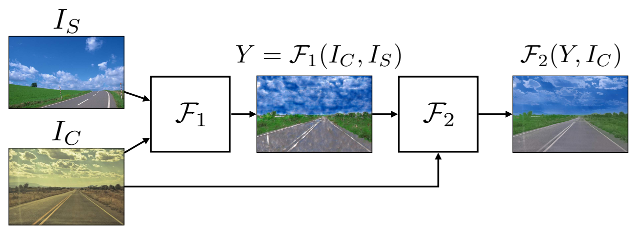


Figure 2: The proposed photorealistic image style transfer algorithm consists of two closed-form function mappings: $\mathcal{F}_1$ and $\mathcal{F}_2$. While $\mathcal{F}_1$ maps $I_C$ to an intermediate image carrying the style of $I_S$, $\mathcal{F}_2$ removes noticeable artifacts and produces a photorealistic stylized result.

**WCT.** The WCT formulates stylization as an image reconstruction problem with feature projections. To utilize WCT, an auto-encoder for general image reconstruction is first trained. Li et al. [17] employ the VGG-19 model [30] as the encoder $\mathcal{E}$, fix the encoder weights, and train a decoder $\mathcal{D}$ for reconstructing the input image. The decoder is designed to be symmetrical to the encoder, with upsampling layers (pink blocks in Figure 3(a)) used to enlarge the spatial resolutions of the feature maps. Once the auto-encoder is trained, a pair of projection functions are inserted at the network bottleneck to perform stylization through the whitening ($P_C$) and coloring ($P_S$) transforms. The key idea behind the WCT is to directly match feature correlations of the content image to those of the style image via the two projections. Specifically, given a pair of content image $I_C$ and style image $I_S$, the WCT first extracts their vectorised VGG features $H_C = \mathcal{E}(I_C)$ and $H_S = \mathcal{E}(I_S)$, and then transform the content feature $H_C$ via

$$H_{CS} = P_S P_C H_C, \tag{2}$$

where $P_C = E_C \Lambda_C^{-\frac{1}{2}} E_C^\top$, and $P_S = E_S \Lambda_S^{\frac{1}{2}} E_S^\top$. Here $\Lambda_C$ and $\Lambda_S$ are the diagonal matrices with the eigenvalues of the covariance matrix $H_C H_C^\top$ and $H_S H_S^\top$ respectively. The matrices $E_C$ and $E_S$ are the corresponding orthonormal matrices of the eigenvectors, respectively. After the transformation, the correlations of transformed features match those of the style features, i.e., $H_{CS} H_{CS}^\top = H_S H_S^\top$. Finally, the stylized image is obtained by directly feeding the transformed feature map into the decoder: $Y = \mathcal{D}(H_{CS})$. For better stylization performance, Li et al. [17] use a multi-level stylization strategy, which performs the WCT on the VGG features at different layers.

The WCT performs well for artistic style transfer. However it generates structural artifacts (e.g., distortions on object boundaries) for photorealistic image stylization (Figure 4(c)). Our PhotoWCT is proposed for suppressing these structural artifacts.

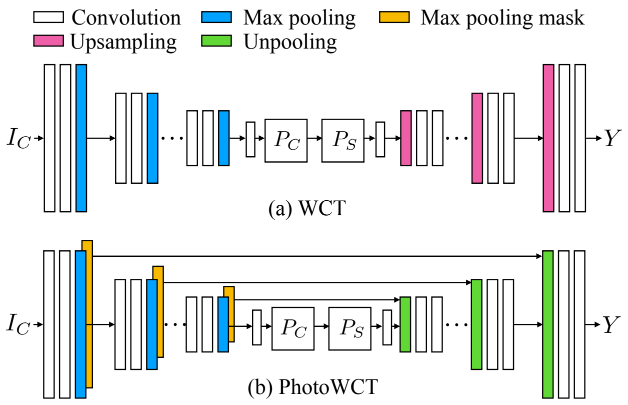


Figure 3: PhotoWCT and WCT. Both share the same encoder architecture and projection steps. In the PhotoWCT, we replace the upsampling layers (pink) with unpooling layers (green) for photorealistic stylization. Note that the unpooling layer is used together with the pooling mask (yellow) which records *where* carries the *argmax* over each max pooling region in the corresponding pooling layer [41].

**PhotoWCT.** Our PhotoWCT design is motivated by the observation that the max-pooling operation in the WCT reduces the spatial information in feature maps. Simply upsampling feature maps in the decoder fails to recover detailed structures of the input image. That is, we need to pass the lost spatial information to the decoder to facilitate reconstructing these fine details. Inspired by the success of the unpooling layer [39, 41, 23] in preserving spatial information, we propose to replace the upsampling layers in the WCT with unpooling layers for photorealistic stylization. As a result, the PhotoWCT function is formulated as

$$Y = \mathcal{F}_1(I_C, I_S) = \overline{\mathcal{D}}(P_S P_C H_C), \tag{3}$$

where $\overline{\mathcal{D}}$ is the decoder trained with unpooling layers for image reconstruction. Figure 3 illustrates the difference in network architecture between the WCT and PhotoWCT.

Figure 4(c) and (d) show the results using the WCT and PhotoWCT. As highlighted in close-ups, the straight lines along the building boundary in the content image become zigzagged when applying the WCT but remains straight when applying the PhotoWCT. The PhotoWCT-stylized image has much fewer structural artifacts.

## 3.2. Photorealistic Smoothing

The PhotoWCT-stylized result (Figure 4(d)) still looks less like a photo since semantically similar regions are stylized inconsistently. For example, as we stylize the day-time photo using the night-time photo in Figure 4, the stylized sky region would be more photorealistic if it were uniformly dark blue instead of partially dark and partially light blue.

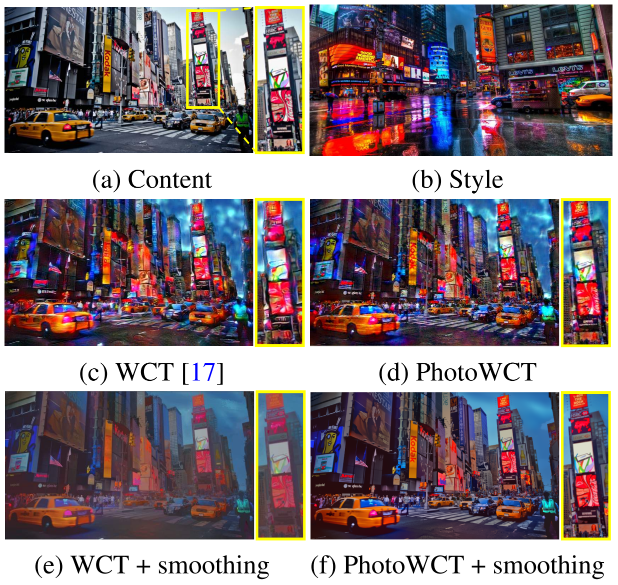


Figure 4: Comparison of image stylization using the WCT and the PhotoWCT. The PhotoWCT better preserves local structures in the content image, which is crucial for the image smoothing step as shown in (e) and (f).

This motivates us to employ the pixel affinities in the content photo to smooth the PhotoWCT-stylized result.

We aim to achieve two goals in the smoothing step. First, pixels with similar content in a local neighborhood should be stylized similarly. Second, the smoothed result should not deviate significantly from the PhotoWCT result in order to maintain the global stylization effects. As such, we first represent all pixels as nodes in a graph and define an affinity matrix $W = \{w_{ij}\} \in \mathbb{R}^{N \times N}$ ($N$ is the number of pixels) to describe pixel similarities. We define a smoothness term and a fitting term that model these two goals in the following optimization problem:

$$\underset{r}{\operatorname{argmin}} \frac{1}{2}(\sum_{i,j=1}^{N} w_{ij}\|\frac{r_i}{\sqrt{d_{ii}}} - \frac{r_j}{\sqrt{d_{jj}}}\|^2 + \lambda \sum_{i=1}^{N}\|r_i - y_i\|^2), \tag{4}$$

where $d_{ii} = \sum_j w_{ij}$ is the diagonal element in the degree matrix $D$ of $W$, i.e., $D = \text{diag}\{d_{11}, d_{22}, ..., d_{NN}\}$. Here, $y_i$ is the pixel color in the PhotoWCT-stylized result $Y$ and $r_i$ is the pixel color in the desired smoothed output $R$. In (4), $\lambda$ controls the balance of these two terms.

Our formulation is motivated by the graph-based ranking algorithms [42, 38]. In their algorithms, $Y$ is a binary input where each element indicates if a specific item is a query ($y_i = 1$ if $y_i$ is a query and $y_i = 0$ otherwise). The optimal solution $R$ is the ranking values of all the items based on their pairwise affinities. In our algorithm, we set $Y$ as the PhotoWCT-stylized result. The optimal solution $R$ is the smoothed version of $Y$ based on the pairwise pixel affinities, which encourages consistent stylization within semantically similar regions.

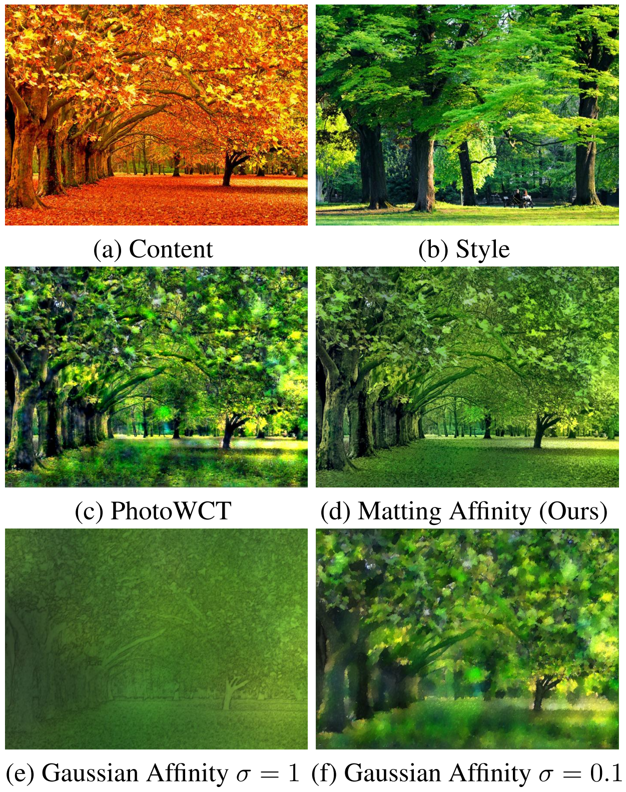


Figure 5: Smoothing with different affinities. To refine the PhotoWCT result in (c), it is hard to find an optimal $\sigma$ for the Gaussian Affinity that works globally well as shown in (e)-(f). In contrast, using the Matting Affinity can simultaneously smooth different regions well as shown in (d).

The above optimization problem is a simple quadratic problem with a closed-form solution, which is given by

$$R^* = (1 - \alpha)(I - \alpha S)^{-1} Y, \tag{5}$$

where $I$ is the identity matrix, $\alpha = \frac{1}{1+\lambda}$ and $S$ is the normalized Laplacian matrix computed from $I_C$, i.e., $S = D^{-\frac{1}{2}} W D^{-\frac{1}{2}} \in \mathbb{R}^{N \times N}$. As the constructed graph is often sparsely connected (i.e., most elements in $W$ are zero), the inverse operation in (5) is computationally efficient. With the close-form solution, the smoothing step can be written as a function mapping given by:

$$R^* = \mathcal{F}_2(Y, I_C) = (1 - \alpha)(I - \alpha S)^{-1} Y. \tag{6}$$

**Affinity.** The affinity matrix $W$ is computed using the content photo based on an 8-connected image graph assumption. While several choices of affinity metrics exist, a popular one is to define the affinity (denoted as the Gaussian

(Figure 4 panel captions: (a) Content (b) Style (c) WCT [17] (d) PhotoWCT (e) WCT + smoothing (f) PhotoWCT + smoothing. Figure 5 panel captions: (a) Content (b) Style (c) PhotoWCT (d) Matting Affinity (Ours) (e) Gaussian Affinity $\sigma = 1$ (f) Gaussian Affinity $\sigma = 0.1$)

affinity) as $w_{ij} = e^{-\|I_i - I_j\|^2/\sigma^2}$ where $I_i, I_j$ are the RGB values of adjacent pixels $i, j$ and $\sigma$ is a global scaling hyper-parameter [26]. However, it is difficult to determine the $\sigma$ value in practice. It often results in either over-smoothing the entire photo (Figure 5(e)) or stylizing the photo inconsistently (Figure 5(f)). To avoid selecting one global scaling hyper-parameter, we resort to the matting affinity [13, 40] where the affinity between two pixels is based on means and variances of pixels in a local window. Figure 5(d) shows that the matting affinity is able to simultaneously smooth different regions well.

**WCT plus Smoothing?** We note that the smoothing step can also remove structural artifacts in the WCT as shown in Figure 4(e). However, it leads to unsatisfactory stylization. The main reason is that the content photo and the WCT result are severely misaligned due to spatial distortions. For example, a stylized pixel of the building in the WCT result may correspond to a pixel of the sky in the content photo. Consequently this causes wrong queries in $Y$ for the smoothing step. That is why we need to use the PhotoWCT to remove distortions first. Figure 4(f) shows that the combination of PhotoWCT and smoothing leads to better photorealism while still maintaining faithful stylization.

## 4. Experimental Results

We first discuss the implementation details. We then present qualitative results comparing the proposed algorithm to several competing algorithms. Finally, we analyze the run-time and several algorithm design choices. Code and additional results are available at https://github.com/NVIDIA/FastPhotoStyle.

**Implementation details.** We use the layers from $conv1\_1$ to $conv4\_1$ of the VGG-19 [30] network as the encoder $\mathcal{E}$. We initialize the encoder using the pretrained weights, which are kept fixed during training. The architecture of the decoder $\overline{\mathcal{D}}$ is symmetrical to the encoder. We train the auto-encoder for minimizing the sum of the $L_2$ reconstruction loss and perceptual loss [11] using the Microsoft COCO dataset [18]. As suggested in [17] for better stylization effects, we apply the PhotoWCT on VGG features at multiple layers to better capture the characteristics of the style photo. The details of the network design are given in the appendix.

To obtain better photorealistic image stylization performance, we use semantic label maps for more localized content and style matching, which are also used by the competing algorithms [7, 21]. Specifically, in the PhotoWCT step, we compute a pair of projection matrices $P_C$ and $P_S$ for each semantic label. We gather the encoder feature vectors corresponding to the same semantic label in the content image to compute a $P_C$, and those in the style image to compute a $P_S$. The feature vectors with a specific semantic label in the content image is then transformed using the corresponding $P_C$ and $P_S$. We also use the post-processing filtering step in Luan et al. [21] for a fair comparison.

**Visual comparisons.** In Figure 6, we provide a visual comparison between the proposed algorithm and two photorealistic stylization methods [24, 21]. The method of Pitié et al. [24] performs a global image transfer through matching the color statistics between the content and style photos, while the method of Luan et al. [21] improves upon the neural style transfer algorithm [6] with a regularization term. From Figure 6, we find that the proposed algorithm outperforms the competing algorithms in generating photorealistic image stylization results. The method of Pitié et al. [24] tends to simply change the color of the content photo and fail to transfer the style. The method of Luan et al. [21] achieves good stylization results, but it renders output images with inconsistent stylizations and noticeable artifacts (e.g., the purple color on the house in the second row).

In Figure 7, we compare the proposed algorithm to two artistic style transfer methods [6, 9]: the neural style transfer algorithm [6] and its fast variant [9]. Both methods successfully stylize the content images but render noticeable structural artifacts and inconsistent stylizations across the images. In contrast, our method produces more photorealistic results.

**User studies.** As photorealistic image stylization is a highly subjective task, we resort to user studies using the Amazon Mechanical Turk (AMT) platform to better evaluate the performance of the proposed algorithm. We compare the proposed algorithm to three competing algorithms, namely the neural style transfer algorithm of Gatys et al. [6], the method of Huang and Belongie [9], and the state-of-the-art photorealistic style transfer algorithm of Luan et al. [21].

We use a set of 32 content–style pairs provided by Luan et al. [21] as a benchmark dataset. In each question, we show the AMT workers the content–style pair and the stylized results of each algorithm in a random order and ask the worker to select an image based on the instructions. A worker must have a lifetime HIT (Human Intelligent Task) approval rate greater than 98% to be qualified to answer the question. For a statistically significant test, each image pair is sent to 10 different workers. This constitutes to 320 questions for each study. We then calculate the average number of times the images from an algorithm is selected, which is the preference score.

We conduct two user studies. In one study, we ask the AMT workers to select which stylized photo better carries the style of the style photo. In the other study, we ask the workers to select which stylized photo looks more like a real photo. In both studies, detailed job instructions with examples are

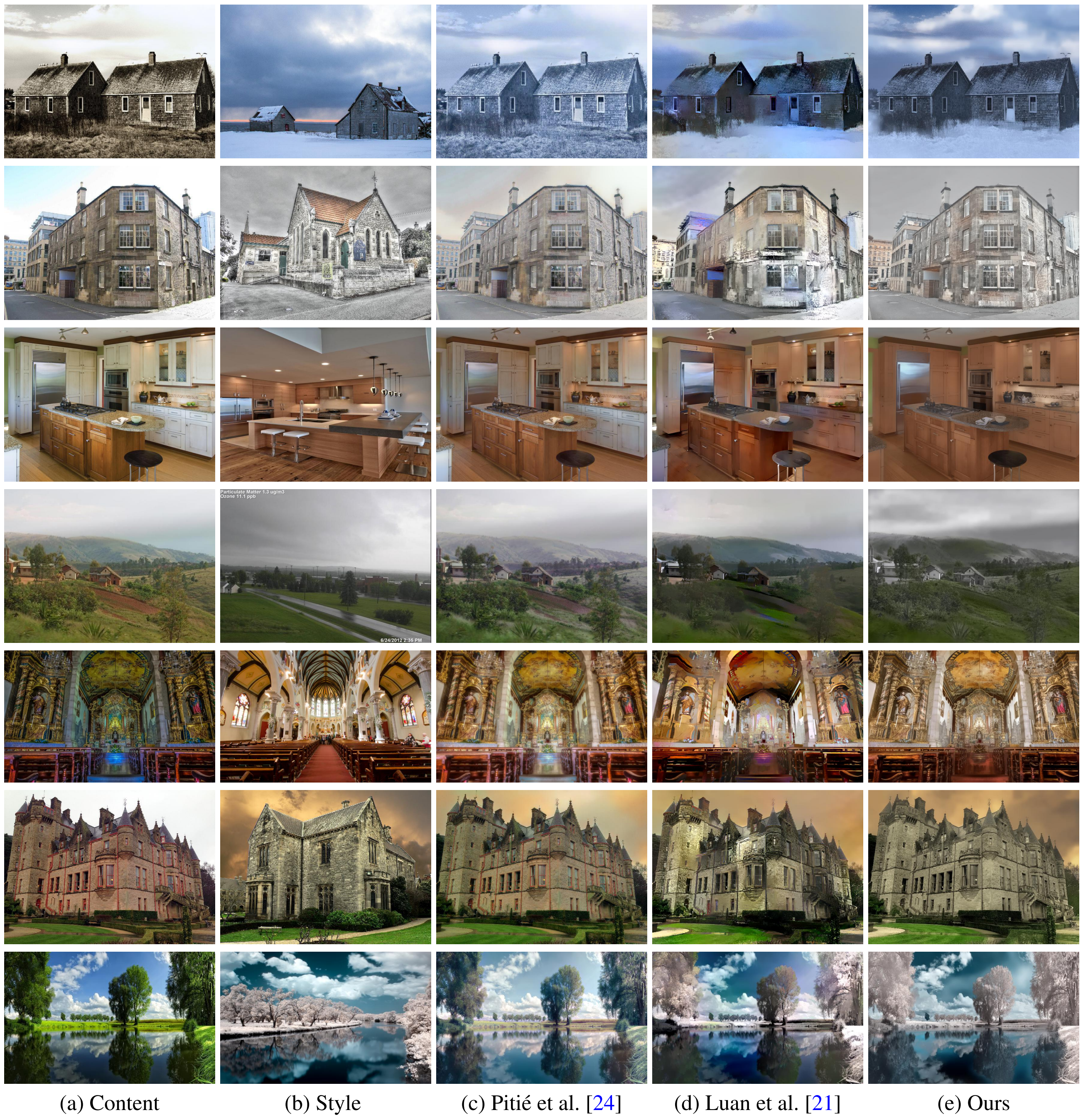


Figure 6: Comparison of different photorealistic stylization methods. We show diverse photo style examples, including season, weather, indoor and outdoor scenery. The method of Pitié et al. [24] often fails to ensure faithful stylization due to its simple color mapping. The results of Luan et al. [21] often contain obvious artifacts from inconsistent stylization effects (e.g., irregular colors on the house, hill, cabinet, ceiling, and tree). Our results are more photorealistic (zoom in).

provided to define the task properly for the worker. Through the two studies, we want to answer which algorithm better stylizes a content image and which algorithm renders better photorealistic output images, respectively. The user study results are shown as round charts in Figure 8. We find the proposed algorithm is preferred for its stylization effect 50% of the times, which is two-times more frequent than the second best method of Luan et al. [21]. For the user study on photorealism, our method is preferred 62% of the time, which significantly outperforms the competing algorithms.

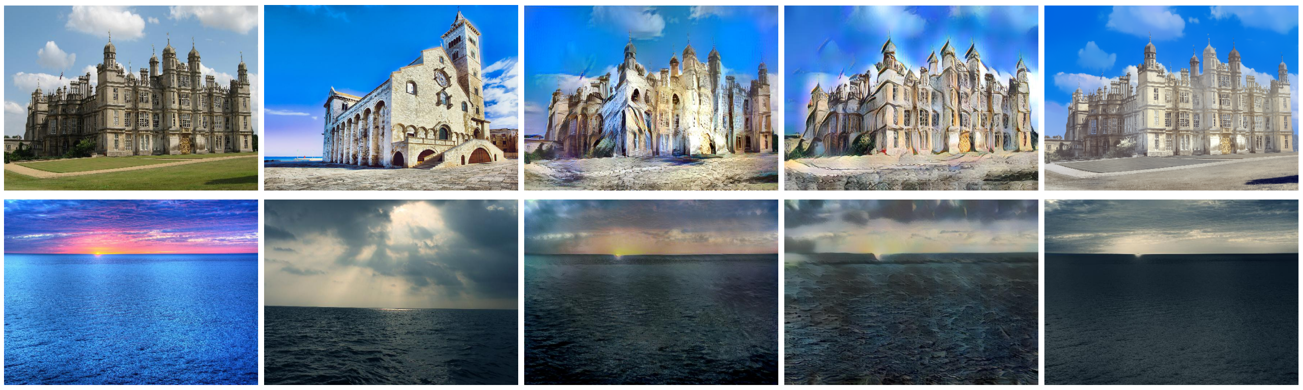
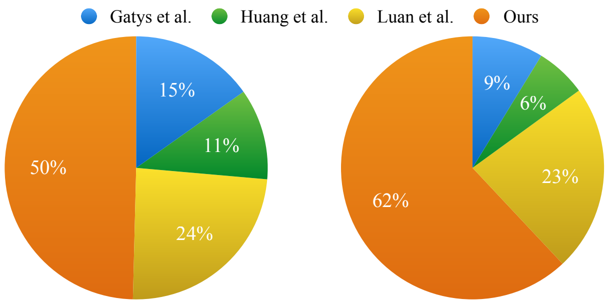

(a) Content (b) Style (c) Gatys et al. [6] (d) Huang et al. [9] (e) Ours

Figure 7: Comparison with artistic stylization methods. Note the severe distortions on the building and sea in (c) and (d). Our results do not contain structural artifacts and exhibit much better photorealism.



(a) Stylization effects (b) Photorealism

Figure 8: User study results. The proposed algorithm is compared to three competing methods on their stylization effects and photorealism. The numbers in the charts indicate the percentage of users favoring a particular algorithm.

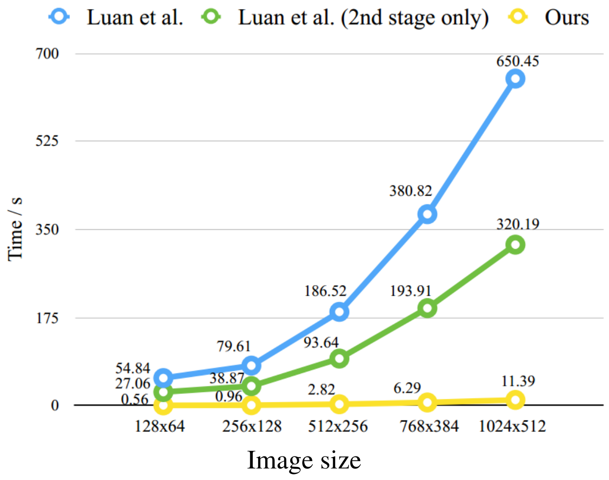


Figure 9: Run-time comparison. We compare the run-time of the proposed method to that of Luan et al. [21] and its second-stage-only variant. The experiments are conducted using a PC with an NVIDIA Titan X Pascal GPU.

**Speed.** We compare our method to the state-of-the-art method [21] in terms of run-time. The method of Luan et al. [21] stylizes a photo by solving two optimization problems in sequence. The first one is to obtain an initial stylized result by solving the optimization problem in the neural style transfer algorithm [6]. The second one is to refine the initial result by adding another regularization term to the optimization objective function of the neural style transfer algorithm. We report the run-time of solving both optimization problems and the run-time of solving the second optimization problem alone. We resize the content images in the benchmark dataset to different sizes and report the average run-time for each image size. The results are shown in Figure 9. For 1K image size, our algorithm takes 11.39s in average, which is much faster than the 650.45s achieved by [21]. For images in small resolution, our algorithm can stylize an image within one second. Overall, our method is 60 times faster that the method of Luan et al. [21].

**WCT versus PhotoWCT.** We compare the proposed algorithm with a variant where the PhotoWCT step is replaced by the WCT [17]. Again, we conduct two user studies for the comparison where one compares stylization effects while the other compares photorealism as described earlier. The result shows that the proposed algorithm is favored over its variant for better stylization 83% of the times and favored for better photorealism 88% of the times.

**Sensitivity analysis on $\lambda$.** In the photorealistic smoothing step, the $\lambda$ balances between the smoothness term and fitting term in (4). A smaller $\lambda$ renders smoother results, while a larger $\lambda$ renders results that are more faithfulness to the queries (the PhotoWCT result). Figure 10 shows results of

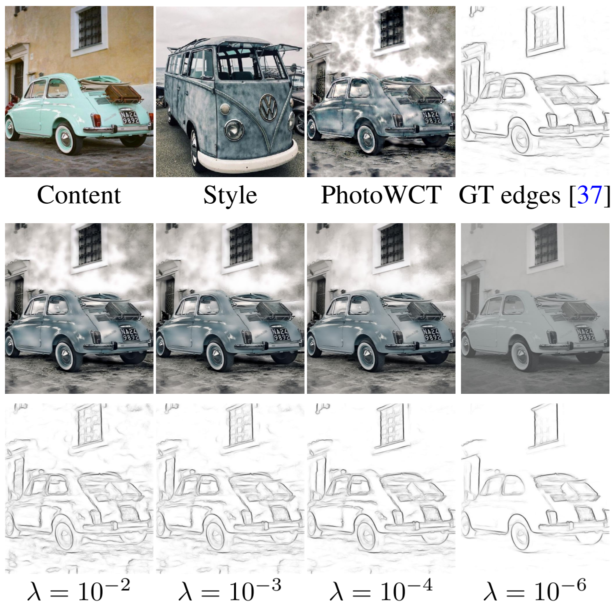


Figure 10: Visualization of the effect of different $\lambda$ in the photorealistic smoothing step.

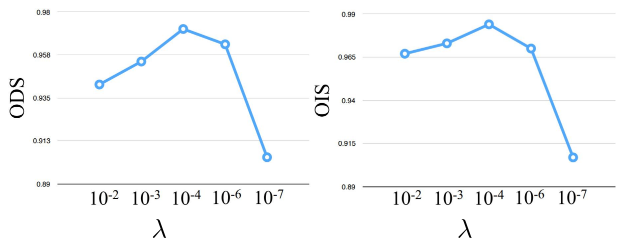


Figure 11: Quantitative stylization performance under different $\lambda$. A higher ODS or OIS score means a more photorealistic stylization. See text for further details.

using different $\lambda$ value. In general, decreasing $\lambda$ helps remove artifacts and hence improves photorealism. However, if $\lambda$ is too small, the output image tends to be over-smoothed. In order to find the optimal $\lambda$, we perform a grid search. We use the similarity between the boundary maps extracted from stylized and original content photos as the criteria since object boundaries should remain the same despite the stylization [2]. We employ the HED method [37] for boundary detection and use two standard boundary detection metrics: ODS and OIS. A higher ODS or OIS score means a stylized photo better preserves the content in the original photo. The average scores over the benchmark dataset are shown in Figure 11. Based on the results, we use $\lambda = 10^{-4}$ in all the experiments.

**Alternative smoothing techniques.** In Figure 12, we compare our photorealistic smoothing step with two alternative approaches. In the first approach, we use the PhotoWCT-stylized photo as the initial solution for solving the second optimization problem in the method of Luan et al. [21]. The result is shown in Figure 12(e). We find that this approach leads to noticeable artifacts as the color on the road is distorted. In the second approach, we use the method of Mechrez et al. [22], which refines stylized results by matching the gradients in the stylized photo to those in the content photo. As shown in Figure 12(f), we find this approach performs well for removing structural distortions on boundaries but does not remove visual artifacts. In contrast, as shown in Figure 12(d), our method generates more photorealistic results with an efficient closed-form solution.

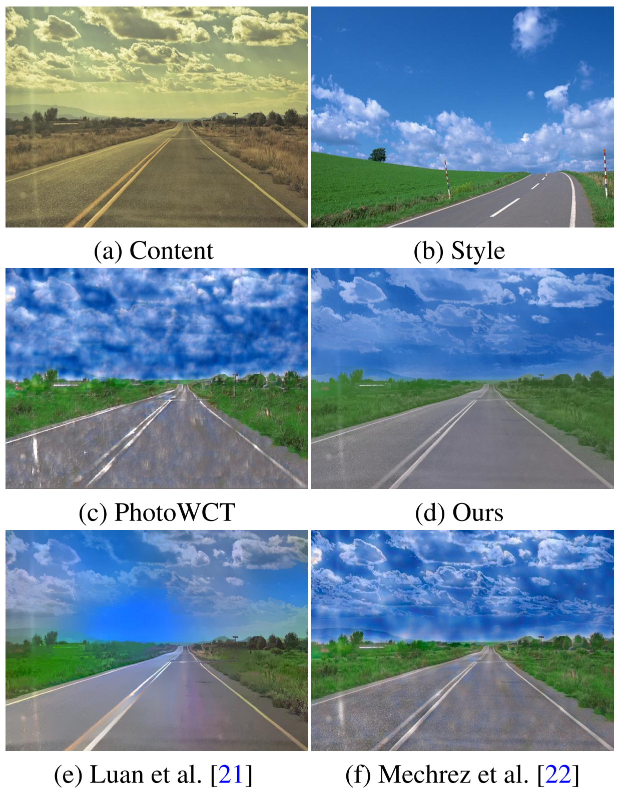


Figure 12: Comparison between our photorealistic smoothing and other refinement methods (d)-(f). We refine the PhotoWCT result (c) in order to encourage spatially consistent stylization for better photorealism.

## 5. Conclusion

We propose a novel fast photorealistic image style transfer algorithm. It consists of a stylization step and a photorealistic smoothing step. Both steps have a closed-form solution. Experimental results show that our algorithm generates results that are twice more favored and runs 60 times faster than the state-of-the-art method.

## References

[1] S. Bae, S. Paris, and F. Durand. Two-scale tone management for photographic look. *ACM Transactions on Graphics*, 25(3):637–645, 2006. 1, 2

[2] F. Cutzu, R. Hammoud, and A. Leykin. Estimating the photorealism of images: Distinguishing paintings from photographs. In *CVPR*, 2003. 8

[3] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. In *ICLR*, 2017. 2

[4] D. Freedman and P. Kisilev. Object-to-object color transfer: Optimal flows and smsp transformations. In *CVPR*, 2010. 2

[5] L. A. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *NIPS*, 2015. 1, 2

[6] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016. 1, 2, 5, 7

[7] L. A. Gatys, A. S. Ecker, M. Bethge, A. Hertzmann, and E. Shechtman. Controlling perceptual factors in neural style transfer. In *CVPR*, 2017. 5, 10

[8] G. Ghiasi, H. Lee, M. Kudlur, V. Dumoulin, and J. Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network. In *BMVC*, 2017. 2

[9] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 2, 5, 7

[10] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 2

[11] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 2, 5

[12] P.-Y. Laffont, Z. Ren, X. Tao, C. Qian, and J. Hays. Transient attributes for high-level understanding and editing of outdoor scenes. *ACM Transactions on Graphics*, 33(4):149, 2014. 1, 2

[13] A. Levin, D. Lischinski, and Y. Weiss. A closed-form solution to natural image matting. *PAMI*, 30(2):228–242, 2008. 5

[14] C. Li and M. Wand. Combining markov random fields and convolutional neural networks for image synthesis. In *CVPR*, 2016. 2

[15] S. Li, X. Xu, L. Nie, and T.-S. Chua. Laplacian-steered neural style transfer. In *ACM MM*, 2017. 2

[16] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Diversified texture synthesis with feed-forward networks. In *CVPR*, 2017. 2

[17] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Universal style transfer via feature transforms. In *NIPS*, 2017. 2, 3, 4, 5, 7, 10

[18] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 5

[19] M.-Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. In *NIPS*, 2017. 2

[20] M.-Y. Liu and O. Tuzel. Coupled generative adversarial networks. In *NIPS*, 2016. 2

[21] F. Luan, S. Paris, E. Shechtman, and K. Bala. Deep photo style transfer. In *CVPR*, 2017. 1, 2, 5, 6, 7, 8

[22] R. Mechrez, E. Shechtman, and L. Zelnik-Manor. Photorealistic style transfer with screened poisson equation. In *BMVC*, 2017. 2, 8

[23] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015. 3

[24] F. Pitié, A. C. Kokaram, and R. Dahyot. N-dimensional probability density function transfer and its application to color transfer. In *ICCV*, 2005. 1, 2, 5, 6

[25] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley. Color transfer between images. *IEEE Computer graphics and applications*, 21(5):34–41, 2001. 1, 2

[26] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 2000. 5

[27] Y. Shih, S. Paris, C. Barnes, W. T. Freeman, and F. Durand. Style transfer for headshot portraits. In *SIGGRAPH*, 2014. 1, 2

[28] Y. Shih, S. Paris, F. Durand, and W. T. Freeman. Data-driven hallucination of different times of day from a single outdoor photo. In *SIGGRAPH*, 2013. 2

[29] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, 2017. 2

[30] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 3, 5, 10

[31] K. Sunkavalli, M. K. Johnson, W. Matusik, and H. Pfister. Multi-scale image harmonization. *ACM Transactions on Graphics*, 29(4):125, 2010. 1

[32] Y. Taigman, A. Polyak, and L. Wolf. Unsupervised cross-domain image generation. In *ICLR*, 2017. 2

[33] Y.-H. Tsai, X. Shen, Z. Lin, K. Sunkavalli, and M.-H. Yang. Sky is not the limit: Semantic-aware sky replacement. *ACM Transactions on Graphics*, 35(4):149, 2016. 2

[34] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *ICML*, 2016. 2

[35] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. *arXiv preprint arXiv:1711.11585*, 2017. 2

[36] F. Wu, W. Dong, Y. Kong, X. Mei, J.-C. Paul, and X. Zhang. Content-based colour transfer. In *Computer Graphics Forum*, volume 32, pages 190–203, 2013. 2

[37] S. Xie and Z. Tu. Holistically-nested edge detection. In *ICCV*, 2015. 8

[38] C. Yang, L. Zhang, H. Lu, X. Ruan, and M.-H. Yang. Saliency detection via graph-based manifold ranking. In *CVPR*, 2013. 4

[39] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 3

[40] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *NIPS*, 2005. 5

[41] J. Zhao, M. Mathieu, R. Goroshin, and Y. LeCun. Stacked what-where auto-encoders. In *ICLR Workshop*, 2016. 3

[42] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schölkopf. Ranking on data manifolds. In *NIPS*, 2004. 4

[43] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017. 2

## A. Semantic Label Map

In order to obtain a better stylization result and give users control to decide the content–style correspondences, our stylization algorithm supports spatial control [7] through using semantic label maps. Labels in the same color represent the corresponding regions in the content and style photos. Note that we do not require precise label maps. Our photorealistic smoothing step, which employs pixel affinities to encourage consistent stylizations, can naturally handle inaccurate semantic labels along object boundaries. This could greatly reduce the labeling burden for users. Figure 13 shows a comparison between using coarse and precise semantic label maps. Results in (e) and (f) show that using the coarse label map can achieve nearly the same stylization performance as using the precise label map.

## B. Multi-level Stylization Strategy

We present the details of our PhotoWCT stylization step, which consists of a reconstruction auto-encoder with unpooling layers, and a pair of feature transforms ($P_C$, $P_S$). The encoder is made of the first few layers of the VGG-19 [30] network. We perform the feature transforms on the features extracted by the encoder.

As suggested in the WCT paper [17], it is advantageous to match features across different levels in the VGG-19 encoder to fully capture the characteristics of the style. We hence adopt a similar strategy for the PhotoWCT. Specifically, we train four decoders for image reconstruction. They are responsible for inverting features extracted from $conv1\_1$, $conv2\_1$, $conv3\_1$ and $conv4\_1$ layer of VGG-19, respectively. With the four encoders, we have a set of 4 auto-encoder networks, which corresponds to a set of 4 PhotoWCT transforms. We apply the transform from the one with the highest feature representation to stylize the content image. The result stylized image is then passed to the transform with the second highest feature representation as shown in Figure 14. Note that the decoders are trained separately and they do not share weights.

Table 2–1 show the detailed configurations of VGG encoder (up to the conv4_1 layer) and the decoders. We use the following abbreviation for ease of presentation: N=Filter number, K=Filter size, S=Stride.

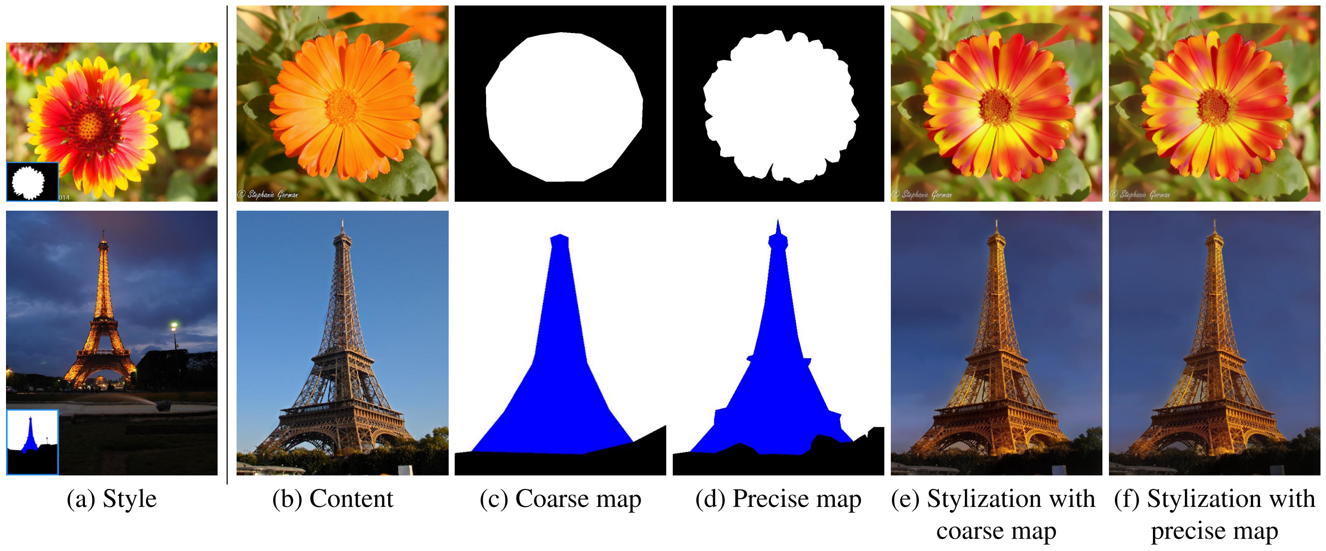

(a) Style
(b) Content
(c) Coarse map
(d) Precise map
(e) Stylization with coarse map
(f) Stylization with precise map

Figure 13: Comparisons of stylization results of using coarse and precise label maps.

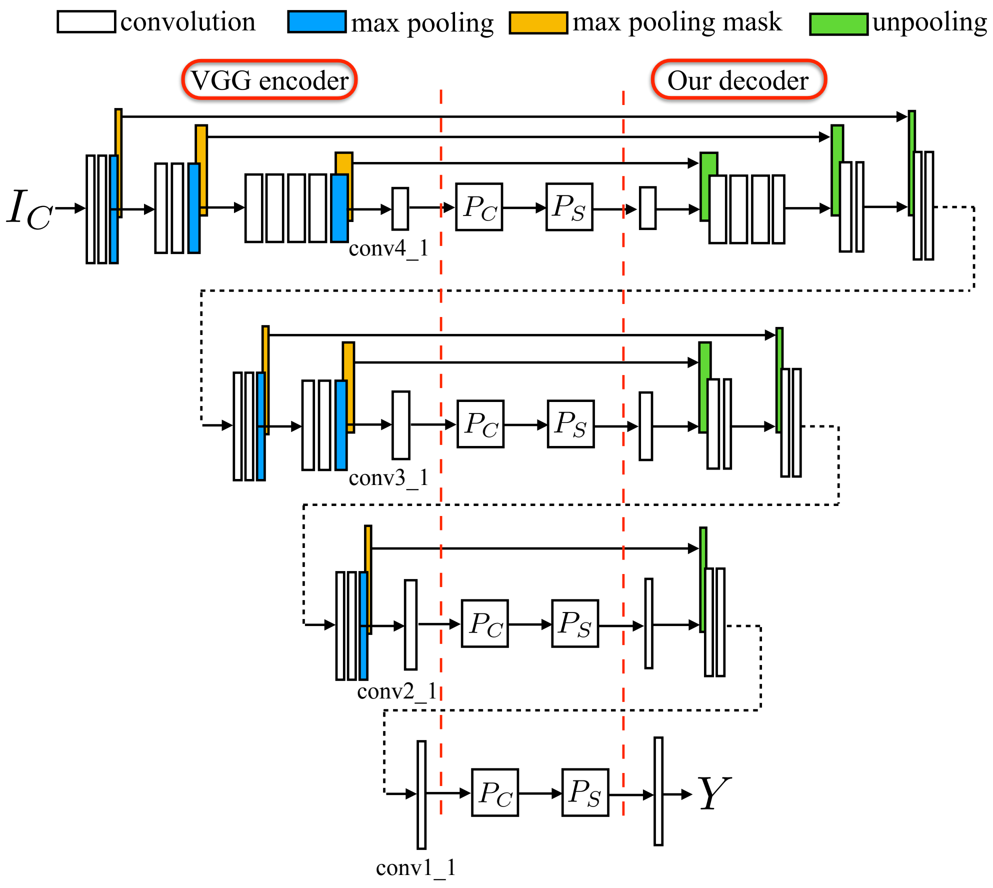


Figure 14: Multi-level stylization strategy with the PhotoWCT transform.

Table 1: Details of the decoders.

| Layer Name | Specification | Decoder 1 | Decoder 2 | Decoder 3 | Decoder 4 |
|---|---|---|---|---|---|
| $inv-conv4\_1$ | Conv (N256, K3, S1), ReLU | v | | | |
| | MaxUnpooling (K2, S2) | v | | | |
| $inv-conv3\_4$ | Conv (N256, K3, S1), ReLU | v | | | |
| $inv-conv3\_3$ | Conv (N256, K3, S1), ReLU | v | | | |
| $inv-conv3\_2$ | Conv (N256, K3, S1), ReLU | v | | | |
| $inv-conv3\_1$ | Conv (N128, K3, S1), ReLU | v | v | | |
| | MaxUnpooling (K2, S2) | v | v | | |
| $inv-conv2\_2$ | Conv (N128, K3, S1), ReLU | v | v | | |
| $inv-conv2\_1$ | Conv (N64, K3, S1), ReLU | v | v | v | |
| | MaxUnpooling (K2, S2) | v | v | v | |
| $inv-conv1\_2$ | Conv (N64, K3, S1), ReLU | v | v | v | |
| $inv-conv1\_1$ | Conv (N3, K3, S1) | v | v | v | v |

Table 2: Details of the VGG-19 encoder.

| Layer Name | Specification (up to $conv4\_1$) |
|---|---|
| $conv1\_1$ | Conv (N64, K3, S1), ReLU |
| $conv1\_2$ | Conv (N64, K3, S1), ReLU |
| | MaxPooling (K2, S2) |
| $conv2\_1$ | Conv (N128, K3, S1), ReLU |
| $conv2\_2$ | Conv (N128, K3, S1), ReLU |
| | MaxPooling (K2, S2) |
| $conv3\_1$ | Conv (N256, K3, S1), ReLU |
| $conv3\_2$ | Conv (N256, K3, S1), ReLU |
| $conv3\_3$ | Conv (N256, K3, S1), ReLU |
| $conv3\_4$ | Conv (N256, K3, S1), ReLU |
| | MaxPooling (K2, S2) |
| $conv4\_1$ | Conv (N512, K3, S1), ReLU |