**Stroke Controllable Fast Style Transfer with Adaptive Receptive Fields**

Yongcheng Jing† Yang Liu† Yezhou Yang‡ Zunlei Feng† Yizhou Yu† Mingli Song††
† Microsoft Visual Perception Laboratory, College of Computer Science and Technology, Zhejiang University
‡ School of Computing, Informatics, and Decision Systems Engineering, Arizona State University

{ycjing,lyng.95,zunleifeng,brooksong}@zju.edu.cn yz.yang@asu.edu yizhouy@acm.org

**Abstract**

Recently, in the community of Neural Style Transfer, several algorithms are proposed to transfer an artistic style in real-time, which is known as Fast Style Transfer. However, controlling the stroke size in stylized results still remains an open challenge. To achieve controllable stroke sizes, several attempts were made including training multiple models and resizing the input image in a variety of scales, respectively. However, their results are not promising regarding the efficiency and quality. In this paper, we present a stroke controllable style transfer network that incorporates different stroke sizes into one single model. Firstly, by analyzing the factors that influence the stroke size, we adopt the idea that both the receptive field and the style image scale should be taken into consideration for most cases. Then we propose a StrokePyramid module to endow the network with adaptive receptive fields, and two training strategies to achieve faster convergence and augment new stroke sizes upon a trained model respectively. Finally, by combining the proposed runtime control techniques, our network can produce distinct stroke sizes in different output images or different spatial regions within the same output image. The experimental results demonstrate that with almost the same number of parameters as the previous Fast Style Transfer algorithm, our network can transfer an artistic style in a stroke controllable manner.

1. **Introduction**

Rendering a photograph with a given artwork style has been a long-standing research topic [12, 29, 27, 13]. Conventionally, the task of style transfer is usually studied as a generalization of texture synthesis [6, 8, 7]. Based on the recent progress in visual texture modelling [9], Gatys et al. firstly propose an algorithm that exploits Convolutional Neural Network (CNN) to recombine the content of a given photograph and the style of an artwork, and reconstruct a visually plausible stylized image, which is known as the process of Neural Style Transfer [10]. Since the seminal work of Gatys et al., Neural Style Transfer has been attracting wide attention from both academia and industry [19, 23, 26]. However, the algorithm of Gatys et al. is based on iterative image optimizations and it requires a slow optimization process for each pair of content and style. To tackle this issue, several algorithms are proposed to speed up the style transfer process, which is called Fast Style Transfer in the literature [11, 25].

Currently, there are three categories of Fast Style Transfer, namely Per-Style-Per-Model (PSPM) [30, 31, 15, 20], Multiple-Style-Per-Model (MSPM) [5, 35, 21, 2] and
Arbitrary-Style-Per-Model (ASPM) [14, 22]. The gist of the PSPM is to train a feed-forward style-specific generator and produces a corresponding stylized result with a forward pass. The MSPM improves the efficiency by further incorporating multiple styles into one single generator. The ASPM tries to transfer an arbitrary style through only one single model.

For these Fast Style Transfer algorithms, there is a trade-off between efficiency and quality [14, 22]. In terms of the quality, the PSPM is usually regarded to produce more appealing stylized results [31, 14]. However, the PSPM is not flexible in terms of controlling perceptual factors (e.g., style-content tradeoff, color control, spatial control). Among these perceptual factors, strokes are one of the most important geometric primitives to characterize an artwork, as is shown in Figure 1. In reality, for the same texture, different artists have their own way to place different sizes of strokes as a reflection of their unique “styles” (e.g., Monet and Pollock). To achieve different stroke sizes with the PSPM, one possible solution is to train multiple models, which is time and space consuming. Another solution is to resize the input image to different scales, which will inevitably hurt the quality of stylization.

In this paper, we propose a stroke controllable Fast Style Transfer algorithm that can incorporate multiple stroke sizes into one single model. Firstly, by analyzing the factors that influence the stroke size in stylized results, we adopt the idea that both the receptive field and the style image scale should be considered for most cases. Based on this idea, we propose a StrokePyramid module to endow the network with adaptive receptive fields and different stroke sizes are learned with different receptive fields. Then a progressive training strategy is introduced to make the network converge faster, and an incremental training strategy is presented to learn new stroke sizes upon a trained model. Finally, by combining two proposed runtime control techniques which are stroke interpolation and spatial stroke control, our network can produce distinct stroke sizes in different output or different spatial regions within the same output image.

In summary, the contributions of this work are threefold:

- We present two runtime control techniques to empower the network with the ability of producing more diverse stroke sizes in different output images and distinct stroke sizes in different spatial regions within the same output image.

2. Related Work

Controlling perceptual factors in Fast Style Transfer. Stroke size control belongs to the domain of controlling perceptual factors during stylization. In this field, several significant works are recently presented. However, there are few efforts devoted to controlling stroke size during Fast Style Transfer. In [11], Gatys et al. study the color control and spatial control for Fast Style Transfer. Lu et al. further extend Gatys et al.’s work to meaningful spatial control by incorporating semantic content, achieving the so-called Fast Semantic Style Transfer [25]. Another related work is Wang et al.’s algorithm which aims to learn large brush strokes for high-resolution images [32]. They find that current Fast Style Transfer algorithms fail to paint large strokes in high-resolution images and propose a coarse-to-fine architecture to solve this problem. Note that the work in [32] is intrinsically different from this paper as one single pre-trained model in [32] still produces one stroke size for the same input image.

Regulating receptive field in neural network. The receptive field is one of the basic concepts in convolutional neural networks, which refers to a region of the input image that one neuron is responsive to. It can affect the performance of the networks and becomes a critical issue in many tasks (e.g., semantic segmentation, image parsing). To regulate the receptive field, [34] proposes the operation of dilated convolution (also called atrous convolution in [3]), which supports the expansion of receptive field by setting different dilation values. Another work in [4] further proposes a deformable convolution which augments the sampling locations in regular convolution with additional offsets. Furthermore, Wei et al. [33] propose a learning-based receptive field regulating method which is to inflate or shrink feature maps automatically on the basis of the learned knowledge.

3. Pre-analysis

Before our pre-analysis, we model the concept of the stroke size firstly. Consider an image in style transfer as a composition of a series of small stroke textons, which are referred as the fundamental geometric micro-structures in images [16, 36]. The stroke size of an image can be defined as the average scale of the composed stroke textons.

In the deep neural network based Fast Style Transfer, three factors are found to influence the stroke size, namely
the whole stroke texton in each region, which influences the stroke texton, the kernels can only learn to paint a part of receptive field in a generative network is smaller than the size of regions, as shown in Figure 3. In particular, when the paint almost the same size of stroke textons in the same size with the size of receptive field. Therefore, given two different to paint a pre-defined size of stroke textons in each region the generative network as teaching the convolutional kernels.

To explain this result, we interpret the training process of the pre-trained VGG loss network, there is no viable image. When the stroke texton is much larger than the receptive field size in the generative network also has influence on the stroke size. In Figure 2, we change the receptive field size in the generative network should generally be considered for stroke size control. As the style image is not high-resolution in most cases, the influence of the receptive field in the loss network is not considered in this work.

4. Proposed Approach

4.1. Problem Formulation

Assume that \( T_i \in \mathbb{T} \) denotes the stroke size of an image, \( \mathbb{T} \) denotes the set of all stroke sizes, and \( I^{T_i} \) represents an image \( I \) with the stroke size \( T_i \). The problem studied in this paper is to incorporate different stroke sizes \( T_i \) \( \in \mathbb{T} \) into the feed-forward fast neural style transfer model. Firstly, we formulate the feed-forward stylization process as:

\[
g(I_c) = I_o, \quad I_o \sim p(I_o|I_c, I_s),
\]

where \( g \) is the trained generator. And the target statistic \( p(I_o) \) of the output image \( I_o \) is characterized by two components, which are the semantic content statistics derived from the input image \( I_c \), and the visual style statistics derived from the style image \( I_s \).

Our feed-forward style transfer process for producing multiple stroke sizes can then be modeled as:

\[
g'(I_c, T_i) = I_o^{T_i}, \quad I_o^{T_i} \sim p(I_o^{T_i}|I_c, I_s, T_i) \quad (T_i \in \mathbb{T}).
\]

We aim to enable one single generator \( g' \) to produce stylized results with multiple stroke sizes \( T_i \in \mathbb{T} \) for the same content image \( I_c \).

4.2. Network Architecture

Based on the analysis in Section 3, to incorporate different stroke sizes into one single model, we propose to design a network with adaptive receptive fields and each receptive field is used to learn a corresponding size of stroke. The
The stroke decoder module takes the feature maps from the first few layers in the network and is shared among different stroke branches to learn both the semantic content and the basic appearances of a style. With different receptive fields, the network learns to paint strokes with different sizes. In particular, to better preserve the desired size of strokes, larger strokes are learned with larger receptive fields, as is explained in Section 3. During the testing phase, given a signal which indicates the desired size of strokes, larger strokes are automatically adapted in the generative network. With the desired stroke size:}

\[
D = \sum_{s} \left( \|I - I_s\|_2 \right) = I_o^{T_s},
\]

Figure 5. Details of our architecture. An image with size 1024 × 1024 is taken as our example input.

where \(F_{B_{s_i}}\) is the output feature map of the branch \(B_{s_i}\) in the StrokePyramid, which corresponds to the stroke size \(T_i\). All the stroke features from the StrokePyramid need to go through the gating function and then be fed into the stroke decoder \(Dec\) to be decoded into the output result \(I_o^{T_s}\) with the desired stroke size:

\[
Dec(\sum_{s} G(F_{B_{s_i}})) = I_o^{T_s}.
\]

4.3. Loss Function

Semantic loss. The semantic loss is defined to preserve the semantic information in the content image, which is formulated as the Euclidean distance between the content image \(I_c\) and the output stylized image \(I_o\) in the feature space of the VGG network [10].
Assume that $F^l(I) \in \mathbb{R}^{C \times H \times W}$ represents the feature map at layer $l$ in VGG network with a given image $I$, where $C$, $H$, and $W$ denote the number of channels, the height and width of the feature map respectively. The semantic content loss is then defined as:

$$L_c = \sum_{l \in \{l_c\}} \| F^l(I_c) - F^l(I_o) \|^2. \tag{5}$$

where $\{l_c\}$ represents the set of VGG layers used to compute the content loss.

**Stroke loss.** The visual style statistics can be well represented by the correlations between filter responses of the style image $I_s$ in different layers of pre-trained VGG network. These feature correlations can be obtained by computing the Gram matrix over the feature map at a certain layer in VGG network. As the gram-based statistic is scale-sensitive, representations of different stroke sizes can be obtained by simply resizing the given style image.

By reshaping $F^l(I)$ into $F^l(I') \in \mathbb{R}^{C \times (H \times W)}$, the Gram matrix $G(F^l(I')) \in \mathbb{R}^{C \times C}$ over feature map $F^l(I)$ can be computed as:

$$G(F^l(I_s')) = [F^l(I_s')][F^l(I_s')]^T. \tag{6}$$

The stroke loss for size $T_k$ can be therefore defined as:

$$L_{T_k} = \sum_{l \in \{l_s\}} \| G(R(I_s, T_k))' - G(F^l(I_o^{B_{s_k}})') \|^2, \tag{7}$$

where $R$ represents the function that resizes the style image to an appropriate scale according to the desired stroke size $T_k$, and $I_o^{B_{s_k}}$ represents the output of the $k$-th stroke branch. $\{l_s\}$ is the set of VGG layers used to calculate the style loss.

The total loss for stroke branch $B_{s_k}$ can then be written as:

$$L_{B_{s_k}} = \alpha L_c + \beta L_{T_k} + \gamma L_{tv}, \tag{8}$$

where $\alpha$, $\beta$ and $\gamma$ are balancing factors. $L_{tv}$ is a total variation regularization loss to encourage the smoothness in the generated images.

**4.4. Training Strategies**

**Progressive training.** To train different stroke branches in one single network, we propose a progressive training strategy. This training strategy stems from the intuition that the training of the latter stroke branch benefits from the knowledge of the previously learned branches. Taken this into consideration, the network learns different stroke sizes with different stroke branches progressively. Assume that the number of the stroke sizes to be learned is $K$. For every $K$ iterations, the network firstly updates the first stroke branch in order to learn the smallest size of stroke. Then, based on the learned knowledge of the first branch, the network uses the second stroke branch to learn the second stroke size with a corresponding scale of the style image. In particular, since the second stroke branch grows the convolutional filters on the basis of the first stroke branch, the updated components in the previous iteration are also adjusted. Similarly, the following stroke branches are updated with the same progressive process. In the next $K$ iterations, the network repeats the above progressive process, since we need to ensure that the network preserves the previously learned stroke sizes.

**Incremental training.** We also propose a flexible incremental training strategy to efficiently augment new stroke sizes upon a trained model. Given a new desired stroke size, instead of learning from scratch, our algorithm incrementally learns the new stroke size by adding one single layer as a new stroke branch in the StrokePyramid. The position of the augmented layer depends on the previously learned stroke sizes and their corresponding receptive fields. By fixing other network components and only updating the augmented layer, the network learns to paint a new size of strokes on the basis of the previously learned stroke features and thus can reach convergence quickly.

**5. Experiment**

**5.1. Implementation Details**

Our proposed network is trained on MS-COCO dataset [24]. All the images are cropped and resized to $512 \times 512$ pixels before training. We adopt the Adam optimizer [18] during training. The pre-trained VGG-19 network [28] is selected as the loss network and \{relu1_1, relu2_1, relu3_1, relu4_1, relu5_1\} are used as the style layers and \{relu1_2, relu2_2, relu3_2, relu4_2, relu5_2\} are used as the content layer. The number of initially learned stroke sizes are set to 3 by
default. For a fair comparison, the parameters of the existing algorithms are set to be the default values according to their published literature. We implement our algorithm based on Tensorflow [1].

5.2. Qualitative Evaluation

Sample results of our algorithm and two aforementioned possible solutions are shown in Figure 6 (the generator for Figure 6(a) is trained using [15]). Our algorithm achieves comparative results with the first possible solution in Figure 6(a) regarding the quality while preserving the flexibility of the second possible solution in Figure 6(b). Figure 7 shows sample results of our algorithm and other Fast Style Transfer algorithms. More results can be found in the supplementary material1. For results from the algorithm of Johnson et al. [15] and Ulyanov et al. [31], training a separate scale-specific generator for each stroke size is adopted, which is the first aforementioned possible solution of stroke size control. Totally three generators are trained using [15] and [31] for each style in Figure 7. It can be noticed that our

1https://youtu.be/UNG38tddMSMg
algorithm achieves competitive results against [15, 31] but exploits only one single pre-trained model. To achieve visually plausible results with $K$ stroke sizes, [15, 31] need to train $K$ corresponding stroke-specific models, which brings additional time cost. In contrast, our algorithm needs less training time due to our progressive training strategy. The algorithm of Huang and Belongie [14] and the algorithm of Li et al. [22] belong to the category of ASPM algorithms and are able to transfer an arbitrary style through one single model. Therefore, these two algorithms do not need to train a style-specific generator in advance and can control the stroke size during stylization by just feeding different scales of style images. However, [14] is not effective at producing some fine textures in some styles (Figure 7, the fourth row). Although [22] captures finer textures, the details are not well preserved in some cases (Figure 7, the fifth row).

5.3. Quantitative Evaluation

In terms of the quantitative evaluation, we focus on three evaluation metrics in this section, which are: training curves during jointly training and incremental training; average content and style loss for test content images; training time for our single model and corresponding generating time for results with different stroke sizes.

Training curve analysis. To demonstrate the effectiveness of our progressive training strategy, we record the stroke losses when learning several sizes of strokes progressively and learning different strokes individually. The result is shown in Figure 8. The reported loss values were averaged over 15 randomly selected batches of content images. It can be observed that the network which progressively learns multiple stroke sizes converges relatively faster than the one which learns only one single stroke size individually. The result indicates that during progressive training, the latter stroke branch benefits from the learned knowledge of the previous branches, and can even improve the training of previous branches through a shared network component in turn. To validate our stroke incremental training strategy, we present both the training curves of the incremental learning and learning from scratch in Figure 9. While achieving comparable stylization quality, incrementally learning a stroke can significantly speed up the training process compared to learning from scratch.

Average loss analysis. To measure how well the loss function is minimized, we compare the average content and style loss of our algorithm with other style transfer methods. For a fair comparison, the loss functions of different algorithms are kept the same. The recorded values are averaged over 100 content images and 5 style images. For each style, we calculate the average loss of the three stroke sizes. As shown in Figure 10, the average style loss of our
algorithm is similar to [31], and our average content loss is slightly lower than [31]. This indicates that our algorithm is comparable to [31] regarding the ability to minimize the loss function.

**Speed analysis.** Fully training one single model with three stroke sizes takes about 2 hours on a single NVIDIA Quadro M6000. For generating time, it takes averagely 0.09 seconds to stylize an image with size $1024 \times 1024$ on the same GPU using our algorithm. Since our network architecture is similar with [15, 31] but with a shorter path for some stroke sizes, our algorithm can be on average faster than [15, 31] and further faster than Huang and Belongie’s algorithm and Li et al.’s algorithm according to the speed analysis in [14, 22].

5.4. Runtime Controls

**Stroke Interpolation.** By interpolating between the output feature maps in the StrokePyramid, our algorithm can achieve arbitrary intermediate stroke sizes. Given a content image $I_c$, we assume that $\mathcal{F}_{B_{sm}}$ and $\mathcal{F}_{B_{sn}}$ are two output feature maps in the StrokePyramid, which can be decoded into the stylized results with two stroke sizes $I_{T_m}$ and $I_{T_n}$ respectively. The interpolated feature $\mathcal{F}_{B_\tilde{s}}$ can then be calculated as:

$$\mathcal{F}_{B_\tilde{s}} = a\mathcal{F}_{B_{sm}} + (1 - a)\mathcal{F}_{B_{sn}}$$ (9)

By changing the value of $a$ and feeding the obtained $\mathcal{F}_{B_{\tilde{s}}}$ into the stroke decoder module, stylized results with arbitrary intermediate stroke sizes $I_{T_\tilde{o}}$ can be produced, as shown in Figure 11.

**Spatial Stroke Control.** Previously, in the community of Fast Style Transfer, stylized results usually have almost the same stroke size across the whole image, which is impractical in the real case. As is shown in Figure 12, our algorithm supports mixed stroke sizes in different spatial regions and in this way, the contrast information in the content image can be enhanced. Our spatial stroke control is achieved by feeding masked content image through different corresponding stroke branches and then combining these stylized results. The mask can be obtained either by manual labelling or forwarding the content image through a pre-trained semantic segmentation network.

6. Conclusions

In this paper, we present a Fast Style Transfer deep network that allows a flexible control for the stroke size during stylization. By using almost the same number of parameters as the previous Fast Style Transfer algorithm, our network is capable of learning multiple stroke sizes. The main idea behind our technique is proposing a StrokePyramid module to endow the network with adaptive receptive
fields and the network can learn to paint different stroke sizes with the corresponding size of the receptive field. By adopting the proposed progressive training strategy, our network achieves faster convergence and through incremental training strategy, new stroke sizes can be augmented in a trained mode. Finally, our network can produce distinct stroke sizes in different output images or different spatial regions within the same output image. Embedded with other existing perceptual factor controlling strategies, our work takes a step towards breaking the tradeoff between the flexibility and efficiency in Fast Style Transfer. Our future work will focus on exploring the influence of other factors on the stroke size. Another research direction is to apply the idea of the StrokePyramid into the MSPM so as to efficiently learn multiple stroke sizes for multiple styles.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467, 2016.

[2] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua. Stylebank: An explicit representation for neural image style transfer. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017. 1

[3] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets. Atrous Convolution, and Fully Connected CRFs. arXiv, 2016. 2

[4] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. arXiv preprint arXiv:1703.06211, 2017. 2

[5] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. In International Conference on Learning Representations, 2017. 1

[6] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pages 341–346. ACM, 2001. 1

[7] M. Elad and P. Milanfar. Style transfer via texture synthesis. IEEE Transactions on Image Processing, 26(5):2338–2351, 2017. 1

[8] O. Frigo, N. Sabater, J. Delon, and P. Hellier. Split and match: Example-based adaptive patch sampling for unsupervised style transfer. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 553–561, 2016. 1

[9] L. A. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In Advances in Neural Information Processing Systems, pages 262–270, 2015. 1

[10] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2414–2423, 2016. 1, 3, 4

[11] L. A. Gatys, A. S. Ecker, M. Bethge, A. Hertzmann, and E. Shechtman. Controlling perceptual factors in neural style transfer. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017. 1, 2, 3

[12] B. Gooch and A. Gooch. Non-photorealistic rendering. A. K. Peters, Ltd., Natick, MA, USA, 2001. 1

[13] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pages 327–340. ACM, 2001. 1

[14] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In Proceedings of the IEEE International Conference on Computer Vision, 2017. 2, 6, 7, 8

[15] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In European Conference on Computer Vision, pages 694–711, 2016. 1, 4, 6, 7, 8

[16] B. Julesz et al. Textons, the elements of texture perception, and their interactions. Nature, 290(5802):91–97, 1981. 2

[17] J. Kim, J. Kwon Lee, and K. Mu Lee. Accurate image super-resolution using very deep convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1646–1654, 2016. 5

[18] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In International Conference on Learning Representations, 2015. 5

[19] C. Li and M. Wand. Combining markov random fields and convolutional neural networks for image synthesis. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2479–2486, 2016. 1

[20] C. Li and M. Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In European Conference on Computer Vision, pages 702–716, 2016. 1

[21] Y. Li, F. Chen, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Diversified texture synthesis with feed-forward networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017. 1, 6

[22] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Universal style transfer via feature transforms. In Advances in Neural Information Processing Systems, 2017. 2, 7, 8

[23] Y. Li, N. Wang, J. Liu, and X. Hou. Demystifying neural style transfer. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, pages 2230–2236, 2017. 1, 3

[24] 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 European conference on computer vision, pages 740–755. Springer, 2014. 5

[25] M. Lu, H. Zhao, A. Yao, F. Xu, Y. Chen, and L. Zhang. Decoder network over lightweight reconstructed feature for fast semantic style transfer. In Proceedings of the IEEE International Conference on Computer Vision, 2017. 1, 2

[26] I. Prisma Labs. Prisma: Turn memories into art using artificial intelligence, 2016. 1
[27] P. Rosin and J. Collomosse. *Image and video-based artistic stylisation*, volume 42. Springer Science & Business Media, 2012.

[28] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[29] T. Strothotte and S. Schlechtweg. *Non-photorealistic computer graphics: modeling, rendering, and animation*. Morgan Kaufmann, 2002.

[30] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. In *International Conference on Machine Learning*, pages 1349–1357, 2016.

[31] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[32] X. Wang, G. Oxholm, D. Zhang, and Y.-F. Wang. Multimodal transfer: A hierarchical deep convolutional neural network for fast artistic style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[33] Z. Wei, Y. Sun, J. Wang, H. Lai, and S. Liu. Learning adaptive receptive fields for deep image parsing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2434–2442, 2017.

[34] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations*, 2016.

[35] H. Zhang and K. Dana. Multi-style generative network for real-time transfer. *ArXiv e-prints*, Mar. 2017.

[36] S.-C. Zhu, C.-E. Guo, Y. Wang, and Z. Xu. What are textons? *International Journal of Computer Vision*, 62(1):121–143, 2005.