Rethinking Style Transfer: From Pixels to Parameterized Brushstrokes

Dmytro Kotovenko* Matthias Wright* Arthur Heimbrecht Björn Ommer
IWR, Heidelberg Collaboratory for Image Processing, Heidelberg University

Abstract

There have been many successful implementations of neural style transfer in recent years. In most of these works, the stylization process is confined to the pixel domain. However, we argue that this representation is unnatural because paintings usually consist of brushstrokes rather than pixels. We propose a method to stylize images by optimizing parameterized brushstrokes instead of pixels and further introduce a simple differentiable rendering mechanism.
Our approach significantly improves visual quality and enables additional control over the stylization process such as controlling the flow of brushstrokes through user input.
We provide qualitative and quantitative evaluations that show the efficacy of the proposed parameterized representation. Code is available at `https://github.com/CompVis/brushstroke-parameterized-style-transfer`.

1. Introduction

Style and texture transfer have been research topics for decades [17, 9]. More recently, the seminal work by Gatys et al. [11] reformulated style transfer as the synthesis of an image combining content of one image with style of another image. Since then, a plethora of approaches have explored different aspects of the original problem. There are papers on feed-forward architectures [23, 48], universal feed-forward models [18, 32, 33, 31], disentanglement of style and content [44, 28, 29], ultra-resolution models [50], meta-learning techniques [45, 56], and video style transfer [4]. Yet, the initial approach suggested by Gatys et al. [11] remains one of the best in terms of image quality, especially in the artistic style transfer scenario, with one style image and one content image.

Recent works have advanced the field of style transfer and produced impressive results by introducing novel losses [34, 43, 44], adopting more suitable architectures [23, 48, 18, 32], imposing regularizations on the final image and intermediate latent representation [44, 28, 29, 47],

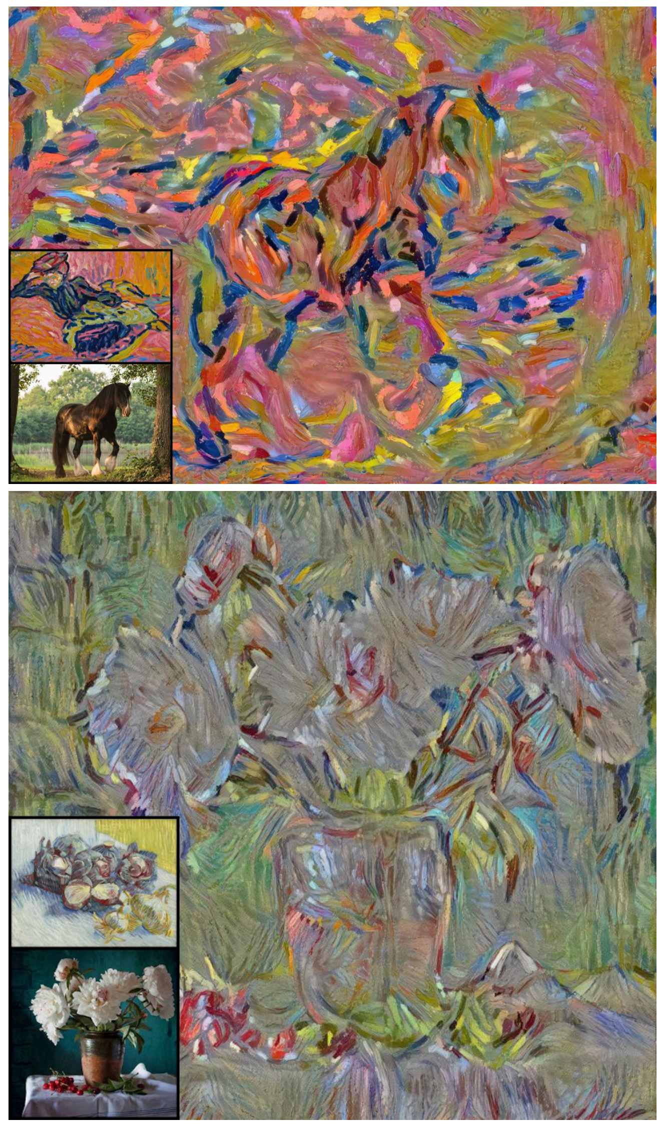


Figure 1: Stylization results. Top artwork: "Girl on a Divan" by Ernst Ludwig Kirchner. Bottom artwork: "Red Cabbages and Onions" by Vincent van Gogh.

and even using different training paradigms [45, 56]. However, they share a key commonality: the stylization pro-

*Both authors contributed equally to this work.

cess is confined to the pixel domain, almost as if style transfer is a special case of image-to-image translation [21, 58, 51, 36, 19, 37, 6, 7, 25]. We argue that the pixel representation is unnatural for the task of artistic style transfer: artists compose their paintings with brushstrokes, not with individual pixels. While position, color, shape, placement and interaction of brushstrokes play an important role in the creation of an artwork, small irregularities appearing on the pixel level like bristle marks, canvas texture or pigments are to some extent arbitrary and random.

With this in mind, we take a step back and rethink the original approach by suggesting a representation that inherently aligns with these characteristics by design.
Just like learning to walk in the reinforcement learning setting starts with defining the set of constraints and degrees of freedom for individual joints, we restrict our representation to a collection of brushstrokes instead of pixels. Specifically, we parameterize a brushstroke with a Bézier curve and additional parameters for color, width, and location.
To map these parameterized brushstrokes into the pixel domain we propose a lightweight, explicit, differentiable renderer which serves as a mapping between brushstroke parameters and pixels. Thus, this reparameterization can be seamlessly combined with other style transfer approaches. One crucial property that this rendering mechanism offers is a spatial relocation ability of groups of pixels. Standard optimization on the pixel level cannot directly move pixels across the image - instead it dims pixels in one area and highlights them in another area. Our model, however, parameterizes brushstrokes with location and shape, thus moving brushstrokes becomes a more natural transformation.

We validate the effectiveness of this reparameterization by coupling the renderer with the model Gatys et al. [11] have suggested, see Fig. 4. We show that this simple shift of representation along with our rendering mechanism can outperform modern style transfer approaches in terms of stylization quality. This is measured using 1) the deception rate - how similar is the stylized image to the style of an artist 2) human deception rate - whether a human subject can distinguish cropouts of real artworks from cropouts of our stylization. In addition, we illustrate that the brushstroke representation offers more control. A user can control brushstrokes, change the flow of strokes in a neighbourhood.

We further conduct experiments on reconstructions of an image using our rendering mechanism. Huang et al. [20] train a neural network that successively fits colored quadratic Bézier curves (brushstrokes) that approximate a target image. Our renderer can be applied to this task as well. It achieves almost 2 times smaller mean squared error (MSE) in the pixel space for a large number of strokes (1000 strokes) and 20% smaller MSE using 200 strokes.

2. Related Work

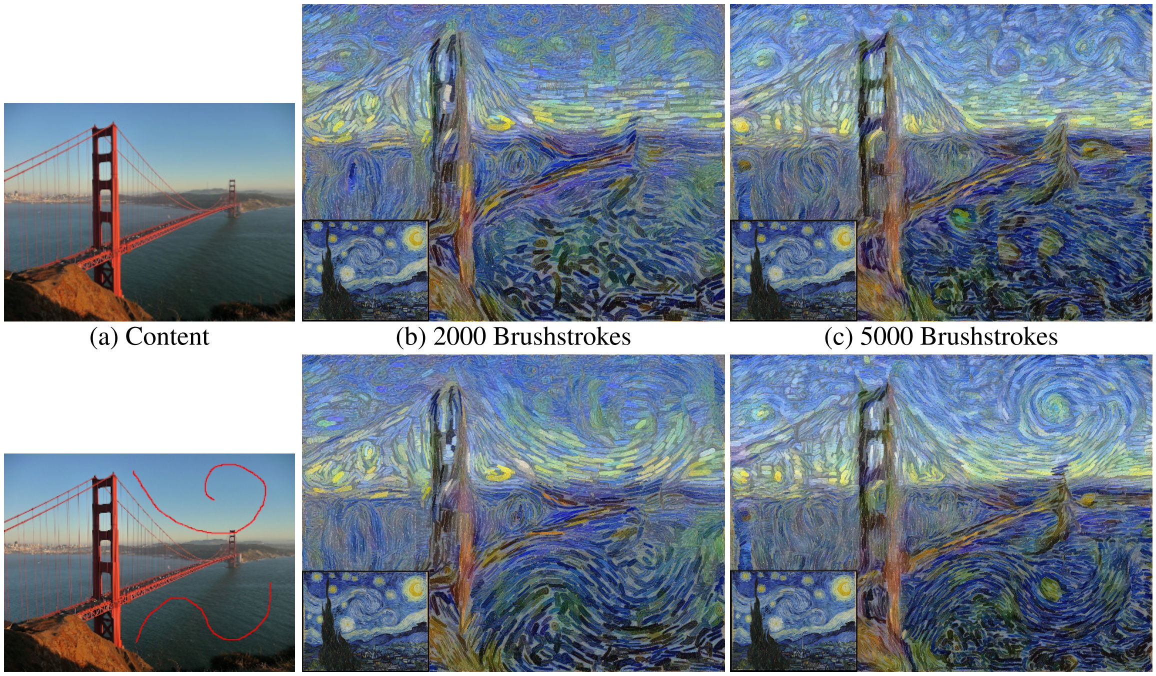
Style Transfer. Initially, Efros and Freeman [9] performed texture synthesis and transfer using image quilting and Hertzmann et al. [17] used a pair of images - one being a filtered version of the other - to learn a filter, which can then be applied to a new image. Wang et al. [49] introduced a method for synthesizing directional textures. Besides that, there are works studying shape and morphology of images [54, 40, 39].
More recently, Gatys et al. [11] proposed an iterative method for combining the content of one image with the style of another by jointly minimizing content and style losses, where the content loss compares the features of a pretrained VGG network [46] and the style loss compares the feature correlations as given by the Gram matrices.

Several works [23, 48] have proposed feed-forward networks to approximate the optimization problem posed by Gatys et al. [11] for a fixed style image.
Li et al. [34] showed that matching the Gram matrices of feature maps corresponds to minimizing the Maximum Mean Discrepancy with the second order polynomial kernel and also proposed alternative style representations to the Gram matrix such as mean and variance. Dumoulin et al. [8] introduced conditional instance normalization, which enables the model to learn multiple styles. Huang and Belongie [18] performed arbitrary real-time style transfer by training a feed-forward network to align the channel-wise mean and standard deviation of the VGG features of a content image to match those of a given style image. Li et al. [32] extend this approach by replacing the moment matching between the encoder and decoder with whitening and colouring transformations.

Li et al. [33] propose a closed-form solution for photorealistic image stylization and Li et al. [31] learn linear transformations for fast arbitrary style transfer. Sanakoyeu et al. [44] and Kotovenko et al. [28] propose a style-aware content loss, which also has been used for disentanglement of style and content [29].

Another line of work draws on meta learning to handle the trade-off between speed, flexibility, and quality [45, 56]. Wang et al. [50] incorporate model compression to enable ultra-resolution style transfer [50], Xia et al. perform photorealistic style transfer using local affine transforms [53], Chang et al. [3] employ domain-specific mappings for style transfer, Chiu and Gurari [5] propose an iterative and analytical solution to the style transfer problem, and Kim et al. [26] suggest a method for deformable style transfer that is not restricted to a particular domain. Yim et al. [55] introduce filter style transfer, Wang et al. [52] propose deep feature perturbation, Svoboda et al. [47] perform style transfer with a custom graph convolutional layer, and Chen et al. [4] employ optical flow to stylize videos.

Stroke Based Rendering. Stroke based rendering



(a) Content (b) 2000 Brushstrokes (c) 5000 Brushstrokes

(d) Content with User Input (e) 2000 Brushstrokes and User Input (f) 5000 Brushstrokes and User Input

Figure 2: A user can draw curves on the content image and thus control the flow of the brushstrokes in the stylized image. Note that for the stylizations with user input we also used (a) as content image. The control is imposed on the brushstroke parameters, not the pixels. Images in the middle column are synthesized using 2000 brushstrokes and images in the right column are synthesized with 5000 brushstrokes. See supplementary for more experiments.

(SBR) aims to represent an image as a collection of parameterized strokes or other shapes that can be explicitly defined by a finite set of parameters. In accordance with other non-photorealistic rendering techniques, the goal is not to reconstruct but rather to render the image into an artistic style. Early works include an interactive method by Haeberli [14], where the program follows the cursor across the canvas, obtains a color by point sampling the source image, and then paints a brush of that color. Hertzmann [15] extended this line of research by proposing an automated algorithm that takes a source image and a list of brush sizes, and then paints a series of layers, one for each brush size, on a canvas in order to recreate the source image with a hand-painted appearance. Similar approaches employ segmentation [12] or relaxation [16]. SBR methods are not constrained to static images and have also been used to transform ordinary video segments into animations that possess a hand-painted appearance [35].

Brush Stroke Extraction. Conversely to SBR methods, there have been attempts to detect and extract brush strokes from a given painting. These methods generally utilize edge detection and clustering-based segmentation [30] or other classical computer vision techniques [1, 42] and have been used to analyze paintings.

Drawing Networks. Recent work relies on neural networks to predict brush stroke parameters that approximate a given image, using a variety of architectures and training paradigms. These range from supervised training of feed-forward and recurrent architectures [13, 57, 41] to deep reinforcement learning, using recurrent [10, 22, 38] and feed-forward models [20]. Note that our work is orthogonal to this line of research because we focus on performing style transfer on the level of parameterized brushstrokes.

3. Background

In the original style transfer formulation, Gatys et al. [11] propose an iterative method for combining the content of one image with the style of another by jointly minimizing content and style losses. The content loss is the Euclidean distance between the rendered image I_r and the content image I_c in the VGG feature space:

$$\mathcal{L}_{\text{content}} = ||\phi_l(I_{\text{r}}) - \phi_l(I_{\text{c}})||_2, \tag{1}$$

where $\phi_l(\cdot)$ denotes the l-th layer of the VGG-19 network. The style loss is defined as:

$$\mathcal{L}_{\text{style}} = \sum_{l=0}^{L} w_l E_l \tag{2}$$

Gatys et al. Ours

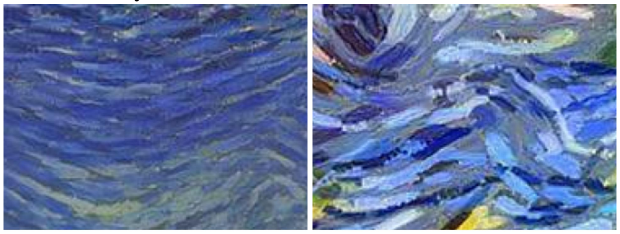


Figure 3: For Gatys et al. [11], the pixels are adjusted to match the brushstroke pattern. In our approach, the brushstroke pattern is occurring by design. Style image: "Starry Night" by Vincent van Gogh. Content image: original image of Tuebingen from the paper [11]. Same region of the sky is cropped.

with

$$E_l = \frac{1}{N_l^2 M_l^2} ||G_r^l - G_s^l||_F \quad (3)$$

where G_r^l and G_s^l are the Gram matrices of I_{r} and I_{c} respectively, computed from the l-th layer of the VGG-19 network.

4. Approach

The method by Gatys et al. [11] adjusts each pixel individually to minimize the content and style losses. However, artworks generally consist of brushstrokes, not pixels. Instead of optimizing on pixels, we therefore optimize directly on parameterized brushstrokes, using the same content and style losses defined in Eq. 1 and Eq. 2, respectively. See Fig. 4 for an overview of our method and Fig. 3 for a comparison of the synthesized brushstroke patterns.

Our brushstrokes are parameterized by location, color, width, and shape. The shape of a brushstroke is modelled as a quadratic Bézier curve [41, 10, 20], which can be parameterized by:

$$\mathbf{B}(t) = (1-t)^2\mathbf{P}_0 + 2(1-t)t\mathbf{P}_1 + t^2\mathbf{P}_2 \,, 0 \leq t \leq 1. \quad (4)$$

A key difficulty here is to find an efficient and differentiable mapping from the brushstroke parameter space into the pixel domain. To this end, we propose a mechanism to construct this mapping explicitly. See Sec. 4.2 for details.

Using our rendering mechanism we can backpropagate gradients from the style and content losses through the rendered pixels directly to the brushstroke parameters.

After the optimization is finished, we render the optimized brushstroke parameters to obtain an image I and then apply the standard Gatys et al. [11] approach on the pixel level using I_s as style image and I as content image. This final step blends the brushstrokes together and adds some texture. Fig. 7 shows the effect of this pixel optimization.

4.1. Implementation Details

Similar to Gatys et al. [11], we use layers "conv4_2" and "conv5_2" for the content loss and layers "conv1_1", "conv2_1", "conv3_1", "conv4_1", and "conv5_1" for the style loss.

We use Adam [27] with learning rate 0.1 for optimization. Similar to Johnson et al. [23], we employ a total variation regularization.

4.2. Differentiable Renderer

Nowadays, generative models have reached unmatched image quality on a variety of datasets [24, 2]. Thus, our first attempt to generate brushstrokes followed this line of work. We generated a dataset of brushstrokes simulated in the FluidPaint environment [1] and trained a network inspired by StyleGAN [24] to generate images conditioned on brushstroke parameters. Despite achieving satisfactory visual quality, the main limitation of this approach is that it is memory-intensive and can not be efficiently scaled to process a large number of brushstrokes in parallel. This is critical for us since our method relies on an iterative optimization procedure.

Therefore, instead of training a neural network to generate brushstrokes, we explicitly construct a differentiable function which transforms a collection of brushstrokes parameterized by location, shape, width and color into pixel values on a canvas. Formally, the renderer is a function:

$$\mathcal{R} : \mathbb{R}^{N \times F} \to \mathbb{R}^{H \times W \times 3}, \quad (5)$$

where N denotes the number of brushstrokes, F the number of brushstroke parameters (12 in our case), and H and W are the height and width of the image to render. This renderer requires less memory and is also not constrained by the limitations of a brushstroke dataset.

4.2.1 Motivation and Idea

Before explaining how our render works, let us start with a simple example. Assume we have a flat disk parameterized with color, radius, and location (1, 1, and 2 scalars respectively) and we want to draw it on a canvas. For the sake of brevity, we assume our images are grayscale but the algorithm trivially generalizes to the RGB space. A grayscale image is a 2D matrix of pixel values. First, we need to decide for every pixel whether or not it belongs to the disk. For this, we simply subtract the disk location from each pixel coordinate and compute the L_2 norm to obtain distances D from each pixel to the disk center. Now we have to check if the distance D is smaller then the radius to get a binary mask M. To incorporate color, it suffices to multiply the mask by a color value.

[1] https://david.li/paint/

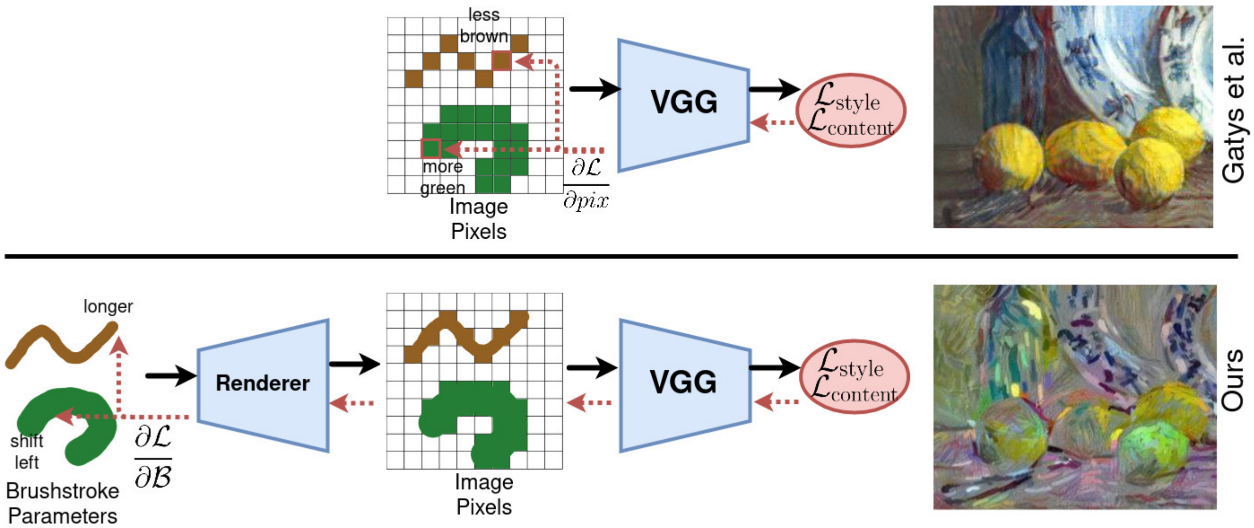


Figure 4: Comparison of our method (bottom row) with Gatys et al. [11] (top row). Gatys et al. [11] optimize pixels to minimize style and content loss. We directly optimize parameters of the brushstrokes. To do that we have designed a differentiable rendering mechanism that maps brushstrokes onto the canvas. Each brushstroke is parameterized by color, location, width and shape. Brushstroke parameters are updated by gradient backpropagation (red, dashed arrows).

If we have two disks, we simply repeat the procedure above for each disk separately and obtain two separate images with disks, namely $I_1, I_2 \in \mathbb{R}^{H\times W\times 3}$. Now, how do we blend I_1, I_2 together? If they do not overlap we can sum the pixel values across disks $I_1 + I_2$. However, if the disks overlap, adding them together will produce artifacts. Therefore, in the overlapping regions, we will assign each pixel to the nearest disk. This can be done by computing the distances $D_1,\ D_2 \in \mathbb{R}^{H\times W}$ from each pixel to each disk center and determine for every pixel the closer disk. We call this object an assignment matrix $A := \{1 \text{ if } D_1 \le D_2,\ 0 \text{ otherwise}\} \in \mathbb{R}^{H\times W}$. Now the final image I can be computed using the matrices I_1, I_2 and A: $I := I_1 * A + I_2 * (\mathbf{1} - A)$. The assignment matrix A naturally generalizes to N objects:

$$A(i,j,n) := \begin{cases} 1 & \text{if } D_n(i,j) < D_k(i,j)\ \forall k \neq n, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

It indicates which object is the nearest to the coordinate (i, j). The final image computation for N images of disks $I_1, .., I_N$ then corresponds to:

$$I(i,j) := \sum_{n=1}^{N} I_n(i,j) * A(i,j,n) \tag{7}$$

Hence, the final image is computed by the weighted sum of renderings weighted according to the assignment matrix A. Both the assignment matrix and the individual renderings $I_1, ..., I_N$ originate from the distance matrices $D_1, .., D_N$ from each pixel location to the object. Indeed, to render a single object we take its distance matrix, threshold with radius/width and multiply by a color value. The assignment matrix is an indicator function of the smallest distance across distances $D_1, .., D_N$. Thus, the matrix of distances is a cornerstone of our approach. We can effectively render any object for which we can compute the distances from each pixel to the object.

Our initial goal was to render brushstrokes. To render a disk we take a distance matrix D, get a mask of points that are closer than the radius and multiply this mask by a color value. The same holds for a Bézier curve.
First, we compute a matrix of distances to the curve $D_{\mathbf{B}}$ (matrix of distances from every point in a 2D image to the nearest point on the Bézier curve).
Then, we mask points that are closer than the brushstroke width and multiply them by a color value. We approximate the distance from a point p to a Bézier curve by sampling S equidistant points $p'_1, .., p'_S$ along the curve and computing the minimum pairwise distance between p and $p'_1, .., p'_S$. Note that there exists an analytical solution of this distance for a quadratic Bézier curve, however, the approximated distance allows the use of arbitrary parametric curves.
In the final step, we can compute the individual renderings of brushstrokes and the assignment matrix as in Eq. 6 and blend them together into the final rendering with Eq. 7.

For the sake of clarity, we have left out two important details in the above explanation.
First, the renderer should be differentiable, yet, the compu-

tation of the assignment matrix and the masking operation are both discontinuous. To alleviate this problem, we implement a masking operation with a sigmoid function. To make the assignment matrix computation differentiable we replace it with a softmax operation with high temperature. Second, the computation of distances between every brushstroke and every pixel on the canvas is computationally expensive, memory-intensive and also redundant because a brushstroke only affects the nearby area of the canvas. Therefore, we limit the computation of distances from a pixel to all the brushstrokes to only the K nearest brushstrokes, see Sec. 3.2 of the supplementary.

Algorithm 1: Renderer

Input: Brushtroke parameters $\mathcal{B} = \{B_1, B_2, ..., B_N\}$, temperature parameter t, number samples per curve S

Output: Image $I \in \mathbb{R}^{H \times W \times 3}$

init $\mathcal{C} \in \mathbb{R}^{H \times W \times 2}$; `// coordinates tensor,` $\mathcal{C}(x, y) = (x, y)$

init tensor of brushtrokes colors $c_{strokes}$ from $\mathcal{B}$ parameters ; `// shape=[N, 3]`

init tensor of brushtrokes widths $w_{strokes}$ from $\mathcal{B}$ parameters ; `// shape=[N]`

sample S points $t \in [0; 1]$ sample points t at each brushtroke

$\mathcal{B}_{sampled} := \{\text{compute } B_i(t_j) \text{ with } Eq.4 \, |\forall i, j\}$; `// shape=[N,S,2]`

$D_(x, y, n, s) := ||\mathcal{C}(x, y) - \mathcal{B}_{sampled}(n, s)||_2$; `// Distances from each sampled point on a stroke to each coordinate, shape=[H,W,N,S]`

$D_{strokes} := min(D, axis = 4)$; `// distance from a coordinate` x, y `to the nearest point on a curve. shape=[H, W, N]`

$M_{strokes} := \mathbf{sigm}(\mathbf{max}(t \cdot ||w_{strokes} - D_{strokes}||_2, axis = 4)$; `// mask of each stroke, shape=[H, W, N]`

$I_{strokes} := M_{strokes} \cdot c_{strokes}$; `// rendering of each stroke, shape=[H, W, N, 3]`

$A := \mathbf{softmax}(t \cdot D_{strokes}, axis = 3)$; `// assignment, shape=[H, W, N]`

$I := \mathbf{einsum}('xyn, xync- > xyc', A, I)$; `// final rendering, see Eq.7`

See Alg.1. The supplementary contains additional technical details of the implementation.

5. Experiments

5.1. Deception Rate

To evaluate the quality of the stylization we use a deception rate proposed by Sanakoyeu et al. [44]. The method is based on a network trained to classify paintings into artists.

Table 1: (Left) Deception score. Wikiart test gives the accuracy on real artworks from the test set. Photos correspond to the content images used by each of the methods for style transfer. (Right) Human deception rate. The probability of labeling randomly sampled crop out of a specified class as real. Both scores are averaged over 8 styles.

Method	Mean deception score ↑	Mean human deception rate ↑
AdaIN [18]	0.08	0.035
WCT [32]	0.11	0.091
Gatys et al. [11]	0.389	0.139
AST [44]	0.451	0.146
Ours	**0.588**	**0.268**
Wikiart test	0.687	-
Photos	0.002	-

The deception rate is the fraction of stylized images that the network has assigned to the artist, whose artwork has been used for stylization. However, a high deception score indicates high similarity to the target image. But this metric does not indicate how plausible a stylized image is. To measure this quality we conduct the following experiment: we show to a human subject four crop outs. Each one can be either taken from a real artwork or from a generated image. The task is to detect all real crop outs. The experiment is conducted with 10 human subjects, each participant evaluates 200+ tuples. Fake images are randomly sampled from one of three methods: ours, Gatys et al. [11], and AST [44]. For each method we report the proportion of ranking this image as real, see Tab. 1.

5.2. Differentiable Renderer

We compare our simple explicitly constructed renderer to the rendering mechanism proposed by Huang et al. [20]. Our approach is slower, but it requires no pretraining on specific datasets as opposed to Huang et al. [20]. We achieve 20% lower mean squared error (MSE) using 200 strokes, and 49% lower MSE on 1000 strokes. The comparison has been conducted on the CelebA dataset. See Fig. 6 for a visual comparison.

5.3. Fitting Brushstrokes to Artwork

We can fit brushstrokes not only to a photograph but also to an artwork. This procedure is useful if we want to study the distribution of brushstrokes in an artwork. It has been shown by Li et al. [30] that this information may be helpful to detect forgeries and analyze the style of an artist. In Fig. 5 we show reconstructions of "Self-Portrait" by Vincent van Gogh obtained using our renderer.

We additionally trained a neural network that receives brushstroke parameters as input and generates the corresponding brushstrokes. The network employs an architecture inspired by StyleGAN [24] and was trained on a dataset obtained using the FluidPaint environment. The brushstroke

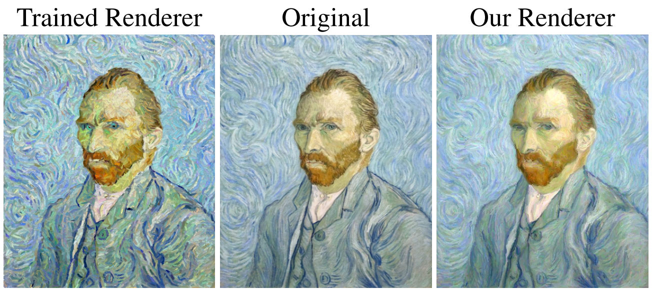


Figure 5: Reconstructions of "Self-Portrait" by Vincent van Gogh using our brushstroke renderer and a trained renderer. In either case we use 10.000 brushstrokes.

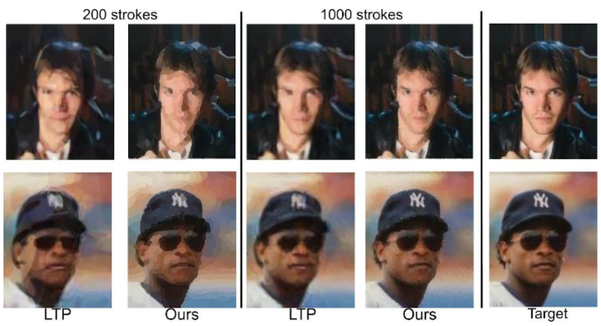
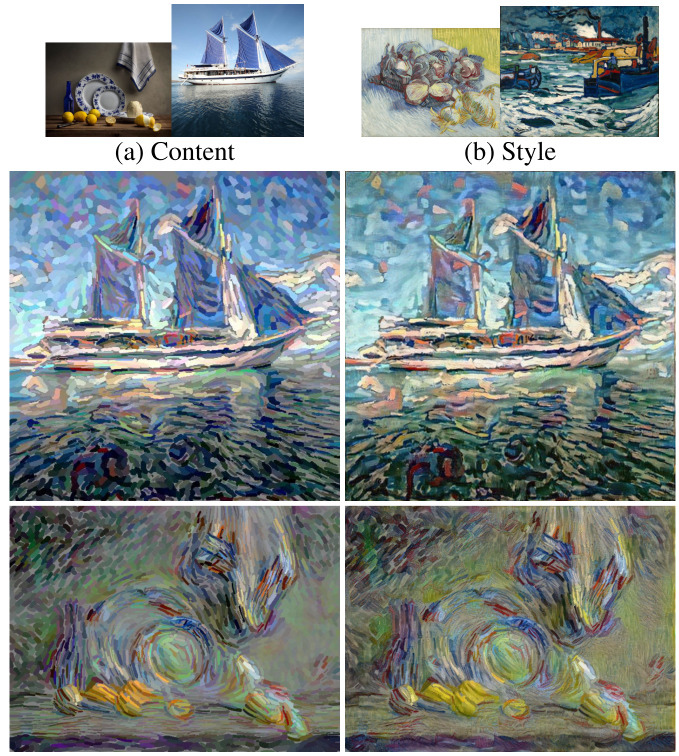


Figure 6: Comparison to the Learning to Paint (LTP) by Huang et al. [20] on the image reconstruction task. Our method directly minimizes l_2 distance between the input target image and image rendered as a collection of brushstrokes. Using our renderer we achieve 20% lower Mean Squared Error (MSE) for 200 strokes and 49% lower MSE for 1000 strokes. Please zoom in for details.

parameterization is as described in this paper. The trained renderer yields results comparable to our simple renderer but requires more precise hyperparameter tuning an takes more time to optimize on. Since the trained renderer is based on the StyleGAN [24] architecture, it consumes much more memory and thus fitting hundreds or thousands of brushstrokes cannot be run in parallel. In Fig. 5 we present results of our renderer and the trained renderer. See supplementary for more details.

5.4. Controlling Brushstrokes

To highlight the additional control our brushstroke representation enables over the stylization process, we show how users can control the flow of brushstrokes in the stylized image, see Fig. 2. A user can draw arbitrary curves on the content image and the brushstrokes in the stylized image will follow these curves. This can be achieved by adding a simple projection loss that enforces brushstrokes along the drawn paths to align with the tangent vectors of the paths. See Sec. 2 of the supplementary for details. Fig. 2 further shows the effect the number of brushstrokes has on the stylization.



(c) Before Pixel Optimization (d) After Pixel Optimization

Figure 7: The effect of the pixel optimization. Brushstrokes are blended together and texture is added. Zoom in for details.

6. Conclusion and Future Work

In this paper, we have proposed to switch the representation for style transfer from pixels to parameterized brushstrokes. We argue that the latter representation is more natural for artistic style transfer and show how it benefits the visual quality of the stylizations and enables additional control.

We have further introduced an explicit rendering mechanism and show that it can be applied even beyond the field of style transfer.

A limitation of our approach is that it performs best for artistic styles where brushstrokes are clearly visible. This can potentially be alleviated with more sophisticated brushstroke blending procedures and should be investigated in future endeavors.

7. Acknowledgments

This work has been supported in part by the German Research Foundation (DFG) within project 421703927.

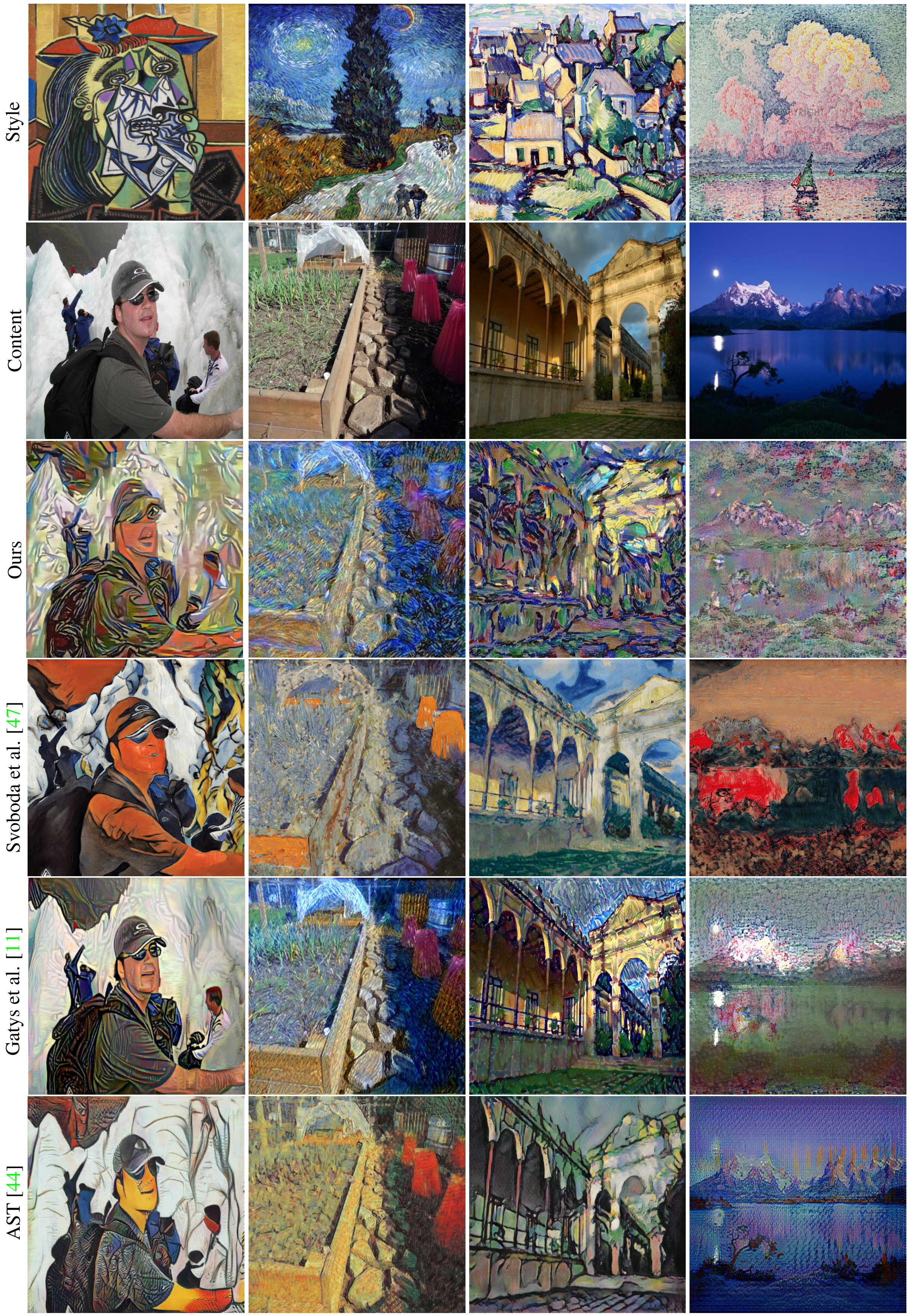


Figure 8: Comparison with other methods for images used by Svoboda et al. [47] and AST [44], please zoom in for details. See supplementary for full size images.

References

[1] Igor E. Berezhnoy, Eric O. Postma, and H. Jaap Herik. Automatic extraction of brushstroke orientation from paintings. *Mach. Vision Appl.*, 20(1):1–9, Jan. 2009.

[2] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[3] Hsin-Yu Chang, Zhixiang Wang, and Yung-Yu Chuang. Domain-specific mappings for generative adversarial style transfer. In *Eur. Conf. Comput. Vis.*, 2020.

[4] Xinghao Chen, Yiman Zhang, Yunhe Wang, Han Shu, Chunjing Xu, and Chang Xu. Optical flow distillation: Towards efficient and stable video style transfer. In *Eur. Conf. Comput. Vis.*, 2020.

[5] Tai-Yin Chiu and Danna Gurari. Iterative feature transformation for fast and versatile universal style transfer. In *Eur. Conf. Comput. Vis.*, 2020.

[6] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[7] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[8] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. In *Int. Conf. Learn. Represent.*, 2017.

[9] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, page 341–346, New York, NY, USA, 2001. Association for Computing Machinery.

[10] Yaroslav Ganin, Tejas Kulkarni, Igor Babuschkin, S.M. Ali Eslami, and Oriol Vinyals. Synthesizing programs for images using reinforced adversarial learning. 2018.

[11] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, June 2016.

[12] Bruce Gooch, Greg Coombe, and Peter Shirley. Artistic vision: Painterly rendering using computer vision techniques. In *Proceedings of the 2nd International Symposium on Non-Photorealistic Animation and Rendering*, NPAR '02, page 83–ff, New York, NY, USA, 2002. Association for Computing Machinery.

[13] David Ha and Douglas Eck. A neural representation of sketch drawings. In *Int. Conf. Learn. Represent.*, 2018.

[14] Paul Haeberli. Paint by numbers: Abstract image representations. SIGGRAPH '90, page 207–214, New York, NY, USA, 1990. Association for Computing Machinery.

[15] Aaron Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. In *SIGGRAPH '98*, 1998.

[16] Aaron Hertzmann. Paint by relaxation. In *Computer Graphics International 2001*, CGI '01, page 47–54, USA, 2001. IEEE Computer Society.

[17] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. SIGGRAPH '01, page 327–340, New York, NY, USA, 2001. Association for Computing Machinery.

[18] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Int. Conf. Comput. Vis.*, 2017.

[19] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *Eur. Conf. Comput. Vis.*, 2018.

[20] Zhewei Huang, Wen Heng, and Shuchang Zhou. Learning to paint with model-based deep reinforcement learning. 2019.

[21] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial nets. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[22] Biao Jia, Chen Fang, Jonathan Brandt, Byungmoon Kim, and D. Manocha. Paintbot: A reinforcement learning approach for natural media painting. *ArXiv*, abs/1904.02201, 2019.

[23] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Eur. Conf. Comput. Vis.*, 2016.

[24] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[25] Junho Kim, Minjae Kim, Hyeonwoo Kang, and Kwang Hee Lee. U-gat-it: Unsupervised generative attentional networks with adaptive layer-instance normalization for image-to-image translation. In *Int. Conf. Learn. Represent.*, 2020.

[26] Sunnie S. Y. Kim, Nicholas Kolkin, Jason Salavon, and Gregory Shakhnarovich. Deformable style transfer. In *Eur. Conf. Comput. Vis.*, 2020.

[27] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Int. Conf. Learn. Represent.*, 2014.

[28] Dmytro Kotovenko, Artsiom Sanakoyeu, Sabine Lang, Pingchuan Ma, and Björn Ommer. Using a transformation content block for image style transfer. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[29] Dmytro Kotovenko, Artsiom Sanakoyeu, Sabine Lang, and Björn Ommer. Content and style disentanglement for artistic style transfer. In *Int. Conf. Comput. Vis.*, 2019.

[30] Jia Li, Lei Yao, Ella Hendriks, and James Z. Wang. Rhythmic brushstrokes distinguish van gogh from his contemporaries: Findings via automated brushstroke extraction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(6):1159–1176, 2012.

[31] Xueting Li, Sifei Liu, Jan Kautz, and Ming-Hsuan Yang. Learning linear transformations for fast arbitrary style transfer. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[32] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. In *Adv. Neural Inform. Process. Syst.*, 2017.

[33] Yijun Li, Ming-Yu Liu, Xueting Li, Ming-Hsuan Yang, and Jan Kautz. A closed-form solution to photorealistic image stylization. In *Eur. Conf. Comput. Vis.*, 2018.

[34] Yanghao Li, Naiyan Wang, Jiaying Liu, and Xiaodi Hou. Demystifying neural style transfer. In *IJCAI*, page 2230–2236. AAAI Press, 2017.

[35] Peter Litwinowicz. Processing images and video for an impressionist effect. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, page 407–414, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[36] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Adv. Neural Inform. Process. Syst.*, 2017.

[37] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation. In *Int. Conf. Comput. Vis.*, 2019.

[38] John F. J. Mellor, Eunbyung Park, Yaroslav Ganin, I. Babuschkin, T. Kulkarni, Dan Rosenbaum, Andy Ballard, T. Weber, Oriol Vinyals, and S. Eslami. Unsupervised doodling and painting with improved spiral. *ArXiv*, abs/1910.01007, 2019.

[39] Antonio Monroy, Peter Bell, and Björn Ommer. Morphological analysis for investigating artistic images. *Image and Vision Computing*, 32(6):414–423, 2014.

[40] Antonio Monroy, Bernd Carqué, and Björn Ommer. Reconstructing the drawing process of reproductions from medieval images. In *2011 18th IEEE International Conference on Image Processing*, pages 2917–2920, 2011.

[41] Reiichiro Nakano. Neural painters: A learned differentiable constraint for generating brushstroke paintings. *ArXiv*, abs/1904.08410, 2019.

[42] Tieta Putri, Ramakrishnan Mukundan, and Kourosh Neshatian. Artistic style characterization of vincent van gogh s paintings using extracted features from visible brush strokes. In *Conference on Pattern Recognition Applications and Methods (ICPRAM)*, 2017.

[43] Eric Risser, Pierre Wilmot, and Connelly Barnes. Stable and controllable neural texture synthesis and style transfer using histogram losses. *ArXiv*, abs/1701.08893, 2017.

[44] Artsiom Sanakoyeu, Dmytro Kotovenko, Sabine Lang, and Björn Ommer. A style-aware content loss for real-time hd style transfer. In *Eur. Conf. Comput. Vis.*, 2018.

[45] Falong Shen, Shuicheng Yan, and Gang. Zeng. Neural style transfer via meta networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8061–8069, 2018.

[46] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Int. Conf. Learn. Represent.*, 2015.

[47] Jan Svoboda, Asha Anoosheh, Christian Osendorfer, and Jonathan Masci. Two-stage peer-regularized feature recombination for arbitrary image style transfer. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[48] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. volume 48 of *Proceedings of Machine Learning Research*, pages 1349–1357, New York, New York, USA, 20–22 Jun 2016.

[49] Bin Wang, Wenping Wang, Huaiping Yang, and Jiaguang Sun. Efficient example-based painting and synthesis of 2d directional texture. *IEEE Transactions on Visualization and Computer Graphics*, 10(3):266–277, May 2004.

[50] Huan Wang, Yijun Li, Yuehai Wang, Haoji Hu, and Ming-Hsuan Yang. Collaborative distillation for ultra-resolution universal style transfer. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[51] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[52] Zhizhong Wang, Lei Zhao, Haibo Chen, Lihong Qiu, Qihang Mo, Sihuan Lin, Wei Xing, and Dongming Lu. Diversified arbitrary style transfer via deep feature perturbation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[53] Xide Xia, Meng Zhang, Tianfan Xue, Zheng Sun, Hui Fang, Brian Kulis, and Jiawen Chen. Joint bilateral learning for real-time universal photorealistic style transfer. In *Eur. Conf. Comput. Vis.*, 2020.

[54] Pradeep Yarlagadda and Björn Ommer. From meaningful contours to discriminative object shape. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012*, pages 766–779, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[55] Jonghwa Yim, Jisung Yoo, Won joon Do, Beomsu Kim, and Jihwan Choe. Filter style transfer between photos. In *Eur. Conf. Comput. Vis.*, 2020.

[56] Chi Zhang, Yixin Zhu, and Song-Chun Zhu. Metastyle: Three-way trade-off among speed, flexibility, and quality in neural style transfer. In *AAAI*, 2019.

[57] Ningyuan Zheng, Y. Jiang, and Ding jiang Huang. Strokenet: A neural painting environment. In *Int. Conf. Learn. Represent.*, 2019.

[58] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Int. Conf. Comput. Vis.*, 2017.