Neural Style Transfer for Vector Graphics

Ivan Jarsky[1][a], Valeria Efimova[1][b], Artyom Chebykin[2][c], Viacheslav Shalamov[1][d] and Andrey Filchenkov[1][e]

[1]*ITMO University, Kronverksky Pr. 49, St. Petersburg, Russia*
[2]*SUAI, Bolshaya Morskaya Street 67A, St. Petersburg, Russia*
fi *fi*

Keywords: Vector Graphics, Computer Vision, Neural Style Transfer, DiffVG.

Abstract: Neural style transfer draws researchers' attention, but the interest focuses on bitmap images. Various models have been developed for bitmap image generation both online and offline with arbitrary and pre-trained styles. However, the style transfer between vector images has not almost been considered. Our research shows that applying standard content and style losses insignificantly changes the vector image drawing style because the structure of vector primitives differs a lot from pixels. To handle this problem, we introduce new loss functions. We also develop a new method based on differentiable rasterization that uses these loss functions and can change the color and shape parameters of the content image corresponding to the drawing of the style image. Qualitative experiments demonstrate the effectiveness of the proposed VectorNST method compared with the state-of-the-art neural style transfer approaches for bitmap images and the only existing approach for stylizing vector images, DiffVG. Although the proposed model does not achieve the quality and smoothness of style transfer between bitmap images, we consider our work an important early step in this area. VectorNST code and demo service are available at https://github.com/IzhanVarsky/VectorNST.

1 INTRODUCTION

Style transfer is a task of computer vision aiming to create new visual art objects. Its objective is to synthesize an image, which combines recognizable style patterns of a style image and preserves the subject of a content image.

The pioneering work of Gatys *et al.* (Gatys et al., 2015) in the field of neural style transfer (NST) showed that correlations between image representations extracted from deep neural networks could capture the visual style of an image. Based on this, they proposed the first NST method. Using Gram matrices-based loss functions and training feed-forward neural networks (Li et al., 2017; Ulyanov et al., 2016; Li and Wand, 2016; Johnson et al., 2016), utilizing one model for multiple styles (Dumoulin et al., 2016) and many other essential improvements to the basic method have been proposed. The authors of (Deng et al., 2022) suggested an approach for stylizing images using transformer-based architecture. Contrastive learning strategy is used in the CAST (Zhang et al., 2022) for training style transfer

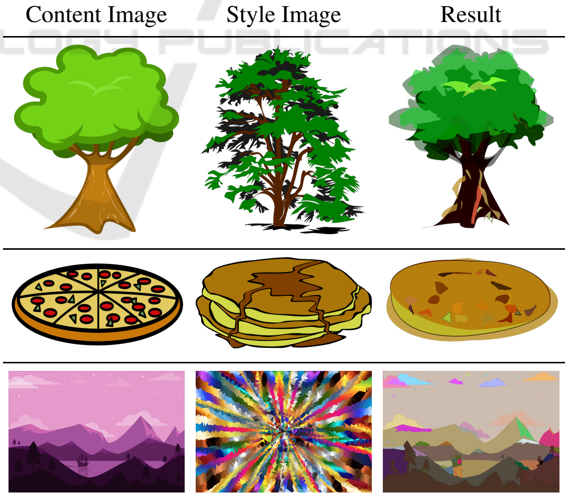

Figure 1: We propose a novel neural style transfer method VectorNST for vector graphics. It takes as inputs a vector content image and some style image and produces a resulting vector image with a style from the style image transferred to the content image.

[a] https://orcid.org/0000-0003-1107-3363
[b] https://orcid.org/0000-0002-5309-2207
[c] https://orcid.org/0009-0002-3163-3727
[d] https://orcid.org/0000-0002-5647-6521
[e] https://orcid.org/0000-0002-1133-8432

Jarsky, I., Efimova, V., Chebykin, A., Shalamov, V. and Filchenkov, A.
Neural Style Transfer for Vector Graphics.
DOI: 10.5220/0012438200003660

In *Proceedings of the 19th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2024) - Volume 3: VISAPP*, pages 686-693
ISBN: 978-989-758-679-8; ISSN: 2184-4321

generator. Dual language-image encoder CLIP (Radford et al., 2021) was used for the image generation and stylization (Kwon and Ye, 2022) using natural language prompts.

One of the main limitations of these methods is that they process bitmap images only of a fixed resolution, which is an essential constraint preventing manipulations with high-resolution images. Image scaling is not applicable for bitmap images without a decrease in quality. Meanwhile, scalability is the feature of vector graphics.

Prior researches tackle the task of vector graphic processing for fonts (Wang and Lian, 2021) and simple graphics such as icons and emoji (Carlier et al., 2020; Reddy et al., 2021), and sketch-like image generation (Frans et al., 2021; Schaldenbrand et al., 2022) using natural language prompts. The study (Efimova et al., 2022) suggests an approach for generating vector images consisting of multiple Bézier curves conditioned by a music track and its emotion.

To the best of our knowledge, the only work that is relevant to NST for vector graphics is the DiffVG method proposed in (Li et al., 2020). However, the authors do not address the NST problem directly, but only provide tools applicable to it. We can thus conclude that the field of vector NST remains majorly untouched.

Two paths exist that lead to styled vector images: (1) rasterize vector input, apply bitmap style transfer algorithms, and then vectorize the result and (2) apply style transfer without directly to the input vector image.

We believe that the second path is preferable due to the following two reasons. First, VGG (Simonyan and Zisserman, 2014) and other backbones that are used for feature extraction are trained on ImageNet, which makes them bitmap-based and unable to classify vector images. Therefore, it is necessary to separately train the network for feature extraction. Second, the first path has a bottleneck: stylized bitmap images must be converted into vector form, which can be done using software algorithms, which produce various artifacts on the image and produce $10-500$ times more curves. A large number of curves makes the vector images difficult to edit. Vectorization approaches without this disadvantage are DiffVG (Li et al., 2020), which produces artifacts on the resulting image, and LIVE (Ma et al., 2022), which is a very resource- and time-consuming. Thus, we consider the raster-then-vectorize approach of generating vector images to be unsuccessful.

Being motivated by this, we decided to find if it is possible to transfer style of a vector image. Our contribution is a novel style transfer method for vector images based on learning how to transform an image via backpropagating contour and perceptual losses through differentiable rasterization transformation, *VectorNST*. Some samples of stylized vector images are presented in Fig. 1.

2 RELATED WORK

2.1 Neural Style Transfer for Raster Graphics

Gatys *et al.* (Gatys et al., 2015) discovered the possibility to separate representations of content and style obtained using a pre-trained CNN (Krizhevsky et al., 2017). They proposed an NST algorithm combining the content of one image with the style of another. It jointly optimizes the loss function responsible for style synthesis and loss for content reconstruction using multiple feature maps from a pre-trained VGG network (Simonyan and Zisserman, 2014). The algorithm starts with random noise and changes pixel values with gradient-based optimization to obtain a stylized image. While producing high-quality results and flexibility, this method is computationally expensive since it requires many forward and backward passes.

To overcome this shortcoming, Johnson *et al.* (Johnson et al., 2016) proposed a feed-forward style transfer network, which synthesizes stylized images in one forward pass; the pre-trained VGG model is used as a loss network. Its performance is similar to the results of Gatys *et al.*, but reduces the inference time. However, the algorithm limitation is that one trained style transfer network can only be used for one style.

Dumoulin *et al.* (Dumoulin et al., 2016) tackled this problem by introducing a conditional style transfer network that can handle multiple styles and is based on a conditional instance normalization algorithm. Defining a specific style requires only trainable parameters of scaling and shifting. Moreover, the latent space of these trainable parameters can be used to interpolate between styles and capture new artistic styles.

To address the problem of high-resolution image generation, Yoo *et al.* (Yoo et al., 2019) proposed an algorithm based on whitening and coloring transforms for the direct change of style representation to match the covariance matrix of content representation. Wavelet Corrected Transmission (WCT2) using Haar wavelet pooling and unpooling allows losing less structural information and maintains the statistical properties of VGG feature space during stylization. It can stylize a 1024×1024 resolution image in

4.7 seconds and obtain a photorealistic result without postprocessing.

A Transformer-based (Vaswani et al., 2017) approach, initially proposed for language processing, can be an alternative to the classic CNN-based methods as it has achieved state-of-the-art results in many computer vision tasks.Park *et al.* (Park and Lee, 2019) proposed the SANet method using the attention mechanism and the identity loss function, which heavily monitors the preservation of image content. However, such an encoder-transfer-decoder architecture cannot handle long-term dependencies, which leads to various distortions and loss of details in a stylized image. Using transformers' ability to handle long-range dependencies, Deng *et al.* (Deng et al., 2022) introduced a transformer-based style transfer framework StyTr^2, which splits content and style images into patches and feeds them into different encoders, and then the transformer decoder stylizes the content sequence according to the style sequence. However, due to the use of a patch-based mechanism, it is difficult to extract and preserve global and local features in a stylized image. Zhang *et al.* (Zhang et al., 2022) presented a framework for style transfer and image style representation based on contrastive learning. Furthermore, style representations are learned directly from image features as well as the global distribution of style. The proposed multi-layer style projector with CNN layers taking as input feature maps from fine-tuned VGG19 encodes the image into a set of codes that are proper guidance for the style transfer generator.

2.2 Vector Graphics

Vector graphics is the most commonly used for various fonts, illustrations, icons, emblems, logos, and other resolution-independent images. Vector graphics is usually declared as a set of primitives such as lines, curves, and circles with many geometric and color attributes.

The most common vector image format is SVG, which is an XML markup text file describing geometric shapes that are mathematically defined by control points. SVG supports many tags and attributes, but the most interesting is the <path> tag, which can be used to describe a shape using Bézier curves. The main advantages of vector graphics are lossless scalability, simplicity, and the memory-efficiency.

Most of the existing methods for neural vector image generation are based on work by Li *et al.* (Li et al., 2020). They introduced the differentiable rasterizer for vector graphics, DiffVG, that allows direct optimization of vector image components such as Bézier curves instead of a matrix of pixels.

On the basis of DiffVG, Frans *et al.* (Frans et al., 2021) introduced the CLIPDraw method that synthesizes vector images conditioned by natural language prompts. CLIPDraw iteratively optimizes a set of RGBA Bézier curves through gradient descent optimizing cosine distance between text encoding and image encoding from the pre-trained CLIP model (Radford et al., 2021). By adjusting text prompts, the model produces different stylized images, which, however, look like sketches rather than pictures. The model performs worse than generative models in high-resolution image generation tasks.

Model-free method for image vectorization, LIVE (Ma et al., 2022) is an approach that offers a completely differentiable way to vectorize bitmap images. Unlike the DiffVG method, which uses random path initialization, LIVE uses an initialization method that determines the best place to add a new path based on the color and size of the component. Although this approach does not use any deep learning model, it implements an iterative image vectorization algorithm, and vectorization of more complex examples requires a lot of resources and takes a long time.

3 METHOD

To develop the style transfer for vector graphics, we use DiffVG to parse vector images and obtain shape parameters: anchor points of vector primitives, shape colors, and line widths. Anchor points are the basis for any vector image, they are used to build curves, which form the figures in the image. Each point is characterized by coordinates $[x, y]$. Also, any curve has a color, it is stored in RGBA format in the interval $[0; 1]$, and a thickness, which is a float number. Unlike style transfer for raster images where only pixel values change, vector images have 3 uncorrelated groups of shape parameters listed above, which can be updated. Changing the parameters of vector primitives is equivalent to transferring the drawing style for bitmap images. Compare NST approaches: in bitmap domain, to transfer style we can only change the color of the pixels in the particular pattern. In vector domain, the drawing style consists of uncorrelated groups of parameters, which can be updated simultaneously or separately.

Based on the above, we aim to develop a model capable of transferring the drawing style of one vector image called a *style image* to another vector image called a *content image* preserving its subject. We do not start the style transfer with a new empty or random vector image, but with a content image, which means that we only change existing shapes and do not create new vector primitives. The method we propose belongs to the iterative optimization methods category, it transfers the style by direct iterative updating shape parameters (Jing et al., 2019). The number of iterations determines the influence of the style image on the result of the style transfer. To allow evaluation of the resulting vector image, it should be rasterized using DiffVG. After that, the similarity between the current image and the style image is measured by the LPIPS method (Zhang et al., 2018) and the similarity between the current image and content image is measured with the Contour Loss. Both LPIPS and the Contour Loss are described in detail in subsection 3.3. The scheme of the method is presented on Fig. 2.

3.1 Differentiable Rasterization with DiffVG

No algorithm exists to compare the similarity between two vector images. However, it is possible to rasterize them and then evaluate their similarity as bitmap images. In this case, rasterization must be performed by a differentiable operator, which is available in DiffVG, allowing thus to apply backpropagation for image updating.

DiffVG is a library which provide functions for reading SVG from source file or parsing SVG from string. The figures and their numerical characteristics, read by this library, are stored as PyTorch tensors, which can be accumulated and transferred to a differentiable rasterization function. The result of this action is a rendered image in RGBA format, stored as a PyTorch tensor. Subsequently, this image can either be saved to disk, or used in further operations - for example, when calculating the loss function. Thus, DiffVG allows to optimize the numerical parameters of the original SVG image using differentiable rasterization.

3.2 Feature Extraction

As a feature extractor, we have chosen the standard VGG-19 network pre-trained on ImageNet. We use the deep embeddings of the 16 convolutional, 5 max pooling, and 16 ReLU activation functions of the 19-layer VGG network[1]. We group these 37 deep embeddings into several intervals by their indices: $[0,4),[9,16),[16,23),[23,30),[30,36)$ (we select features before ReLU) following paper (Zhang et al., 2018). We did not take deep embeddings with indices 4 to 8 because otherwise, it leads to marred contours in the final image.

3.3 Losses

Gatys *et al.* (Gatys et al., 2015) proposed to calculate the style loss based on a Gram matrix, which is effective at representing wide varieties of both natural and non-natural textures. The style loss was designed to capture global statistics but it tosses spatial arrangements, which leads to unsatisfying results for modeling shape parameters and obtaining indecent results for vector images. On contrary, loss evaluation can be done based on the perceptual distance between images. This can be a solution for our task because perceptual losses eliminate the aforementioned drawbacks of the basic method for vector graphics. We introduce our complete loss function:

$$\mathcal{L} = LPIPS(x,y) + \lambda \cdot \mathcal{L}_{contour}(x,z)\,, \tag{1}$$

where $LPIPS$ is the perceptual loss we discuss in detail in the next subsection, $\mathcal{L}_{contour}$ is the regularization on contours we discuss in subsection 3.3.

Learned Perceptual Image Patch Similarity (LPIPS) Metric for Vector Graphics. LPIPS (Zhang et al., 2018) has been used for many computer vision tasks, for example, image restoration and super-resolution. In E-LPIPS (Kettunen et al., 2019), authors proposed to use random transformations before calculating the perceptual similarity between images. After conducting experiments, we found that most of these transformations lead to poorer results for NST for vector graphics. Only the color scale transformation, the coefficient of which is sampled from the standard normal distribution, results in more pleasing colors and smoother contours in the output image.

We use the L_2 term to normalize the feature dimension in all pixels and layers to unit length as it is more stable and computationally effective. Instead of summing L_2 distances between the image activation maps as it was proposed in the original paper (Zhang et al., 2018), we average them to avoid a high range that can cause artifacts in the output image.

Our LPIPS loss implementation is:

$$LPIPS(x,y) = \frac{1}{L}\sum_{l=0}^{L}\frac{1}{H_l W_l C_l}\sum_{h,w}\left\|\hat{x}^l_{hw} - \hat{y}^l_{hw}\right\|_2^2, \tag{2}$$

where $x,y \in R^{1xCxHxW}$ are input images scaled by random channel transformation, L is the number of feature maps used from VGG, (H_l,W_l,C_l) - sizes of height, width and channels in corresponding feature map, (h,w) - indices of height and width, $\hat{x}^l_{hw}$ and $\hat{y}^l_{hw}$ are L_2 normalized feature vectors from feature map l in position (h,w).

The equation illustrates how the distance between style and output images is obtained: we apply the random transformation on both input images, extract and

[1] https://pytorch.org/hub/pytorch_vision_vgg/

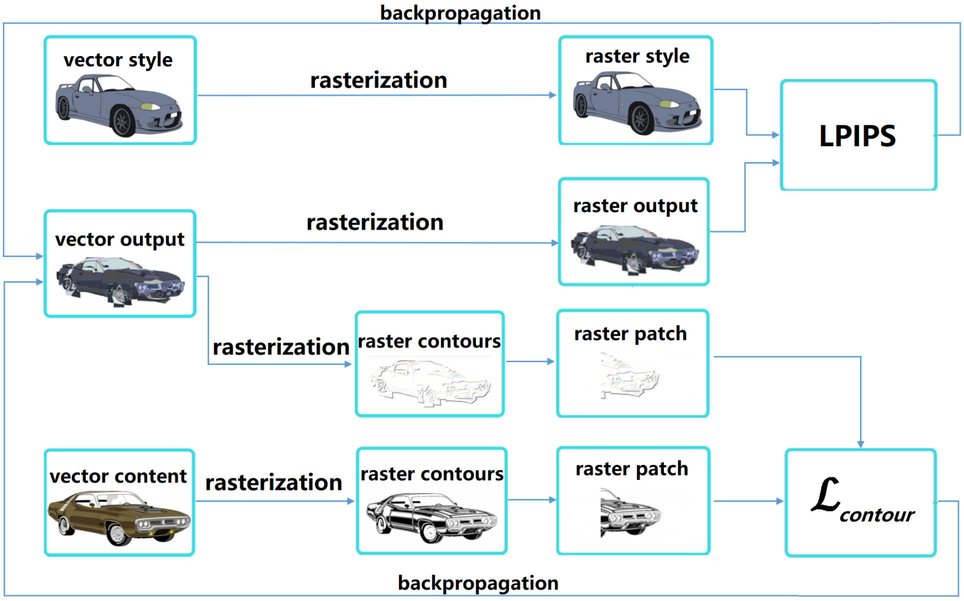

Figure 2: Method overview. We propose a method for real-time style transfer for vector graphics. The optimization consists of two parts. The upper part evaluates the perceptual similarity between the rasterized style image and the output image and aims to convey the style and color of the drawing. The lower part penalizes the differences between the contours of the rasterized content image and the output image to preserve the overall shape of the image.

normalize their features from L layers, and, thus, obtain $\hat{x}^l_{hw}$ and $\hat{y}^l_{hw}$. Then, we compute mean squared L_2 distances, and, finally, we average scores obtained from each layer.

Contour Loss. DiffVG is used to obtain contours of content and current images. We parse them and change the fill and stroke colors of shapes to black and white, respectively. With these vector primitives, we obtain new raster contour images with DiffVG. Then, we crop random patches from both images. We attempted to compute the difference between patches of various size and found that the size of $(W/4, H/4)$ is the most appropriate. After that, we calculate L_1 term:

$$\mathcal{L}_{contour}(x,z) = \frac{1}{n}\sum_{i=1}^{n} |\, x_i - z_i \,|, \qquad (3)$$

where x is the patch of the current image and z is the corresponding patch of the target content image. It forces the input image to respect the target image since the L_1 loss penalizes the distance between them. As a result, it makes images smoother, and more exact and helps obtain sharp outlines.

4 EXPERIMENTS AND RESULTS

In this section, we investigate the behavior of our method and compare it with six other NST methods. We describe the visual differences between the results of these methods, provide the results of a user survey, and estimate the running time of vector methods.

Experiment Setup. We used the Adam optimizer for each of 3 parameter groups used in DiffVG to represent a vector image. The learning rate for color parameters and stroke width was 0.01 and 0.1 correspondingly. Point learning rate lr was chosen depending on the number of shapes in the image, n: if $n \in (0,300), lr = 0.2$; $n \in (300,1000), lr = 0.3$; $n \in (1000,1600), lr = 0.4$; else $lr = 0.8$. Weight of the contour loss $\lambda = 100$.

Methods. We included the following methods in the comparison: (1) DiffVG, the only existing style transfer for vector images, similar to Gatys *et al.*. We selected loss weights following the original implementation[2] : $\lambda_{style} = 500, \lambda_{content} = 1$. (2) Gatys *et al.*, the first and the most widespread method for bitmap images. (3) SANet, (4) StyTr^2, and (5) CAST are three state-of-the-art methods for raster style transfer based on encoder-decoder structure. (6) AttentionedDeepPaint (ADP), a method for sketch colorization conditioned by given style image[3] based on GANs.

Dataset. To assess the quality of the resulting images, we collected a dataset of 500 vector images, mostly sketchy animals, cars, and landscapes from the FreeSVG website[4]. It contains freely distributed SVG

[2] https://github.com/BachiLi/diffvg

[3] https://github.com/ktaebum/AttentionedDeepPaint

[4] https://freesvg.org

files of various domains with no specific focus.

Metrics. Evaluating the results in the field of NST is a sophisticated problem and there is no gold standard by which the best model can be identified. No method can determine how accurately the image style was reproduced, because this task is imprecise, and even a human is often unable to give a correct assessment. Nevertheless, we made attempts to compare the models using style and content losses proposed in the original article by Gatys *et al.*. However, using this approach, we encountered difficulties that did not allow us to make a comparison in this way. Instead, we evaluated generated images by ourselves, involved assessors for quality estimation, and compared the time of inference.

4.1 Visual Comparison

The results of the application of the methods with various style and content image pairs are presented in Fig. 3.

As can be seen from Fig. 3, the Attentioned Deep Paint, SANet, StyTr^2, and CAST methods transfer the style but add a lot of artifacts to the images, while losing content patterns. All raster methods make uniform areas non-uniform. The StyTr^2 method achieves good stylization effects for the owl and hippo images, but at the same time, the stylized images of the tiger and the first landscape contain noticeable artifacts that distort the perception of the content. CAST preserve objects' contours, however, it adds unacceptable extra background.

Although the DiffVG algorithm changes the colors of content images, it blurs the contours or adds distortions, and it cannot convey the style, which is clearly seen in the examples images of a tiger, a hippopotamus, a car, and landscapes. It produces much fewer artifacts, all contours are clear, the pictures are smooth, the color changes (but not everywhere), and the drawing is not transferred, that is, the image content almost does not change.

Our method seeks a trade-off between following the style and freezing the content. It changes the shape and color of vector primitives to preserve the content as much as possible. The sharpness of the contours does not change.

4.2 User Study

We attracted 40 assessors to evaluate the quality of images generated by VectorNST. We conducted a survey asking participants to assess 10 images generated by each method on a scale of 1 to 5 (1 stands for completely inappropriate, 5 stands for the perfect fit). The images were grouped by method without providing any information about the methods. Survey results are presented in Tab. 1.

Table 1: Comparison of survey results to the proposed VectorNST with DiffVG, Gatys *et al.*, StyTR^2, SANet, CAST, and Attentioned Deep Paint.

Method	Score
VectorNST (ours)	0.56 ± 0.04
DiffVG	0.44 ± 0.05
Gatys *et al.*	0.42 ± 0.06
StyTR^2	0.62 ± 0.05
SANet	0.43 ± 0.06
CAST	0.59 ± 0.06
Attentioned Deep Paint	0.11 ± 0.04

Table 2: Timings in seconds. Small stands for 256 × 256 bitmap images and for vector images with a number of shapes less than 100. Medium stands for 512 × 512 bitmap images and for vector images with the number of shapes less between 100 and 700. Big is for bitmap images 1024 × 1024 and greater and for vector images with more than 700 shapes.

Method	Small	Medium	Big
Gatys *et al.*	1.61	4.14	11.59
DiffVG	4.20	26.21	98.57
VectorNST	5.93	33.52	112.10

4.3 Time Comparison

We compare the time required for processing a single image by our method, Gatys *et al.*approach, and its implementation for vector graphics in DiffVG. Because three other methods use pre-trained networks, we excluded them from the comparison.

The speed of Gatys *et al.*depends only on the size of the content image. On the contrast, the speed of our method and DiffVG depend on (1) the content image size (because how many points need to be sampled during rasterization depends on its size); (2) the number of paths (because when creating an image with contours, the number of paths is important and it determines the size of the image during rasterization); (3) the total number of parameters (the sum of the parameters of all three optimizers).

The results of the time comparison can be found in Tab. 2. VectorNST is a bit slower than DiffVG because it spent time on computing the contour loss value. Gatys *et al.*is considerably faster.

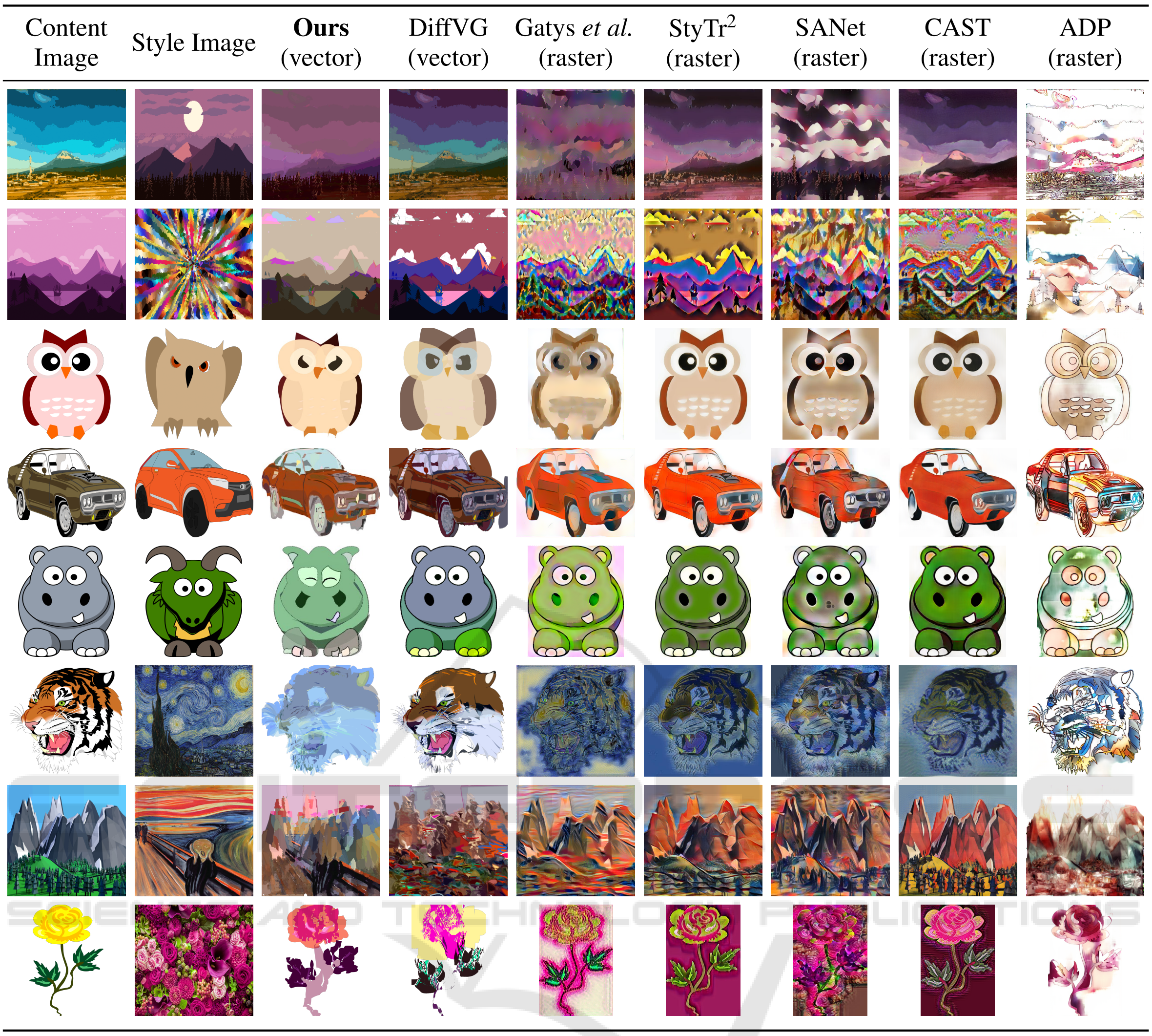

Figure 3: Qualitative comparisons of style transfer results using different methods.

5 CONCLUSION

In this paper, we proposed a novel neural style transfer method for vector graphics, VectorNST, which allows processing illustrations such as sketchy animals, cars, and landscapes. We introduced a loss function consisting of two parts, an adapted LPIPS loss and a contour loss, the latter providing more accurate style transfer and content information preservation. Experimental results demonstrated that our method generates gorgeous stylized vector images and achieves higher human assessment results compared to SANet, Attentioned Deep Paint, and DiffVG methods.

Further improvement of our method would include adding a transformer-based model for more accurate preservation of the vector image contours. Another direction would be to overcome the limitation rooted in DiffVG by making the model capable of changing the input parameters of a number of curves or anchor points via backpropagation. Additionally, future work may include collecting a vector image dataset for improving style transfer inference time as it can be done offline using a pre-trained style network.

ACKNOWLEDGEMENTS

The research was supported by the ITMO University, project 623097 ”Development of libraries containing perspective machine learning methods”.

REFERENCES

Carlier, A., Danelljan, M., Alahi, A., and Timofte, R. (2020). Deepsvg: A hierarchical generative network for vector graphics animation. *Advances in Neural Information Processing Systems*, 33:16351–16361.

Deng, Y., Tang, F., Dong, W., Ma, C., Pan, X., Wang, L., and Xu, C. (2022). Stytr2: Image style transfer with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11326–11336.

Dumoulin, V., Shlens, J., and Kudlur, M. (2016). A learned representation for artistic style. *arXiv preprint arXiv:1610.07629*.

Efimova, V., Jarsky, I., Bizyaev, I., and Filchenkov, A. (2022). Conditional vector graphics generation for music cover images. *arXiv preprint arXiv:2205.07301*.

Frans, K., Soros, L. B., and Witkowski, O. (2021). Clipdraw: Exploring text-to-drawing synthesis through language-image encoders. *arXiv preprint arXiv:2106.14843*.

Gatys, L. A., Ecker, A. S., and Bethge, M. (2015). A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*.

Jing, Y., Yang, Y., Feng, Z., Ye, J., Yu, Y., and Song, M. (2019). Neural style transfer: A review. *IEEE transactions on visualization and computer graphics*, 26(11):3365–3385.

Johnson, J., Alahi, A., and Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer.

Kettunen, M., Härkönen, E., and Lehtinen, J. (2019). E-lpips: robust perceptual image similarity via random transformation ensembles. *arXiv preprint arXiv:1906.03973*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.

Kwon, G. and Ye, J. C. (2022). Clipstyler: Image style transfer with a single text condition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18062–18071.

Li, C. and Wand, M. (2016). 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.

Li, T.-M., Lukáč, M., Gharbi, M., and Ragan-Kelley, J. (2020). Differentiable vector graphics rasterization for editing and learning. *ACM Transactions on Graphics (TOG)*, 39(6):1–15.

Li, Y., Wang, N., Liu, J., and Hou, X. (2017). Demystifying neural style transfer. *arXiv preprint arXiv:1701.01036*.

Ma, X., Zhou, Y., Xu, X., Sun, B., Filev, V., Orlov, N., Fu, Y., and Shi, H. (2022). Towards layer-wise image vectorization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16314–16323.

Park, D. Y. and Lee, K. H. (2019). Arbitrary style transfer with style-attentional networks. In *proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5880–5888.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR.

Reddy, P., Gharbi, M., Lukac, M., and Mitra, N. J. (2021). Im2vec: Synthesizing vector graphics without vector supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7342–7351.

Schaldenbrand, P., Liu, Z., and Oh, J. (2022). Styleclipdraw: Coupling content and style in text-to-drawing translation. *arXiv preprint arXiv:2202.12362*.

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

Ulyanov, D., Lebedev, V., Vedaldi, A., and Lempitsky, V. (2016). Texture networks: Feed-forward synthesis of textures and stylized images. *arXiv preprint arXiv:1603.03417*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Wang, Y. and Lian, Z. (2021). Deepvecfont: synthesizing high-quality vector fonts via dual-modality learning. *ACM Transactions on Graphics (TOG)*, 40(6):1–15.

Yoo, J., Uh, Y., Chun, S., Kang, B., and Ha, J.-W. (2019). Photorealistic style transfer via wavelet transforms. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9036–9045.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018). The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595.

Zhang, Y., Tang, F., Dong, W., Huang, H., Ma, C., Lee, T.-Y., and Xu, C. (2022). Domain enhanced arbitrary image style transfer via contrastive learning. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–8.