Interactive Multi-level Stroke Control for Neural Style Transfer

Max Reimann, Benito Buchheim
Hasso Plattner Institute
University of Potsdam, Germany
max.reimann@hpi.de
benito.buchheim@student.hpi.de

Amir Semmo
Digital Masterpieces GmbH
Potsdam, Germany
amir.semmo@digitalmasterpieces.com

Jürgen Döllner, Matthias Trapp
Hasso Plattner Institute
University of Potsdam, Germany
juergen.doellner@hpi.de
matthias.trapp@hpi.de

***Abstract*—We present *StyleTune*, a mobile app for interactive multi-level control of neural style transfers that facilitates creative adjustments of style elements and enables high output fidelity. In contrast to current mobile neural style transfer apps, *StyleTune* supports users to adjust both the size and orientation of style elements, such as brushstrokes and texture patches, on a global as well as local level. To this end, we propose a novel stroke-adaptive feed-forward style transfer network, that enables control over stroke size and intensity and allows a larger range of edits than current approaches. For additional level-of-control, we propose a network agnostic method for stroke-orientation adjustment by utilizing the rotation-variance of Convolutional Neural Networks (CNNs). To achieve high output fidelity, we further add a patch-based style transfer method that enables users to obtain output resolutions of more than 20 Megapixel (Mpix). Our approach empowers users to create many novel results that are not possible with current mobile neural style transfer apps.**

***Keywords*-neural style transfer, local adjustments, mobile devices, artistic rendering, interaction**

I. Introduction

Machine learning has become of prior interest to both research and end-user applications of image-based artistic rendering [1]. Its usage in mobile expressive rendering setups has particularly increased over the last years, providing essential tools for casual creativity and image filtering [2]. Here, a popular method is to extract the style from one exemplar image and transfer it to a target image or video, thus making a generalized Neural Style Transfer (NST) practicable [3].

NSTs enjoy a great popularity with both users and developers because of its ability to emulate artistic styles without the need to engineer style-specific algorithms. While NSTs have also found their way into several professional tools, such as Photoshop [4], they are mostly limited to resembling "one click solutions". In particular, these implementations are typically constrained to pre-defined styles that can be applied globally to a target image, thus enabling higher-level interaction, but without lower-level control that is often seeked by artists [5] and prosumers of image filtering apps [6]. For instance, no control over perceptual elements of a style such as stroke placement or style granularity are provided, i.e., inherently limiting the degree of artistic expression [7]. Further, existing approaches for low-level control of NSTs generally only considers a univariate adjustment of the stylized outputs, which makes a complex and individual editing impracticable. Furthermore, current mobile apps are quite limited in their output resolutions, which typically stems from inherent hardware limitations.

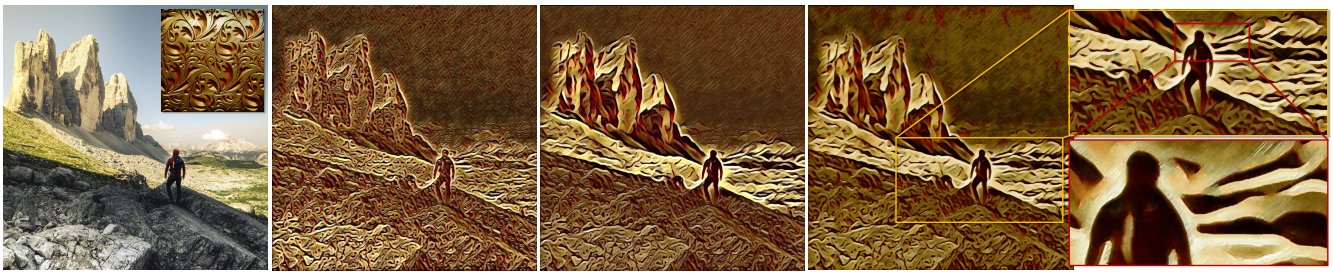
This work presents an approach for multivariate editing of NSTs using a novel feedforward style transfer network that alleviates the aforementioned limitations by incorporating global and local control over style elements, such as their granularity and orientation. We demonstrate the real-world applicability of our concept by presenting a mobile app (*StyleTune*) for interactive editing of NSTs. In addition, our app enables to create high-resolution outputs of the edited results to advance the field of mobile style transfer towards art-directable tools for both casual and professional creativity (Figure 1). To summarize, this paper makes the following contributions:

1) A method for level-of-control interaction to adjust a style's stroke size, intensity, and rotation with a single neural network.
2) A method for efficient upsampling and adaptive global control over NST outputs using a novel two-stream network architecture.
3) An interactive editing pipeline for NSTs on mobile devices, with the ability to achieve high-resolution outputs of 20 Mpix and more.

The remainder of this paper is structured as follows. Section II reviews related and previous work on NST and local-scale adjustable style transfer approaches. Section III describes our method, gives an overview of our system, and outlines implementation aspects. Section IV briefly explains the structure and capabilities of the user interface provided by our mobile app. Section V shows and discusses exemplary results and application examples. Finally, Section VI concludes this paper and provides a prospect on future work.

II. Related Work

NST was introduced in the seminal work of Gatys *et al*. [8], which is based on an iterative optimization of feature statistics of a content, style and target image, extracted using a CNN. The "similarity" of the target image to the style



(a) Content and style images (b) Global transfer (c) Local adjustments (d) Upsampled NST and zoomed versions

Figure 1: Comparison between a global style transfer (b) and a locally-retouched version produced with *StyleTune* (c) based on given content and style images (a). Figure (d) shows a high-resolution result obtained by style-guided upsampling of (c) from 1024×1024 pixels to a resolution of 3200×3200 pixels. (content image from © Shutterstock, used with permission).

is represented as the difference between Gram matrices in style and target image over a set of feature responses of the VGG [9] network. However, the optimization process is computationally intensive and slow, and thus not suited for use in interactive and mobile environments. To approach this issue, several methods of accelerating style transfer have been published—denoted as fast style transfer methods in literature [3]. Generally, these methods train a network to directly inference the stylized output, such as the popular architecture introduced by Johnson *et al*. [10].

Furthermore, feed-forward network architectures can be categorized by their ability to represent different styles. While the approach of Johnson *et al*. [10] can represent a single style per network, follow-up works added the ability to represent multiple styles (Zhang *et al*. [11], Msg Net) or arbitrary styles (Huang *et al*. adaptive Instance Normalization (adaIN) [12]). However, representing more styles with one single network generally represents a trade-off in quality, memory and run-time performance versus single-style networks [13], [14], and thus the single-style-per-network approaches have prevailed in mobile applications as the "gold standard" (e.g., refer to Prisma [15] and Becasso [16]). While the overall goal of most style transfer methods has been to achieve plausible global results without requiring user interaction, several methods allow to adjust perceptual factors of the output in varying degrees, and thus directly or indirectly control semiotic aspects known from artwork production [7]. Gatys *et al*. [17] demonstrate that the iterative approach can adjust the style content trade-off and colorization through weighting different loss terms. Wu *et al*. [18] control stroke orientation by adding an additional direction-aware loss term to the optimization. In the feed-forward network approaches, the controllable factors are either an inherent property of the network or have to be explicitly built into the network architecture. For example, arbitrary NST methods such as adaIN [12] inherently allow adjusting the stroke size by re-scaling the input style image, while this is not possible for single-style transfer methods.

For explicit control, Reimann *et al*. [14] extend a multiple-styles-per-network approach with a consistency loss to enable seamless local combination of different styles. Jing *et al*. [13] introduce an approach for globally and locally adjusting stroke sizes using a multi-branch network architecture that includes a Stroke-Pyramid and explicitly training different style sizes, and Yao *et al*. incorporate self-attention into multi-stroke transfer [19]. Similarly, our approach also extends a feed-forward, single-style-per-network architecture with run-time controls. However, in contrast to previous approaches, it allows a wider range of control for editing and is more efficient with respect to the run-time and memory consumption for large images and large-scale texture marks. We demonstrate how to incorporate our approach and other state-of-the-art NSTs into our pipeline for interactive editing with local guidance and high-resolution upsampling in our *StyleTune* app.

III. Method

A. Preliminary Analysis

Fundamentally, in NST, textural elements of the style such as stroke placement, size, or orientation are not defined explicitly, but implicitly learned by matching the style images Gram-based statistics over extracted features using image recognition networks such as VGG [9]. These stroke textons [20] are micro-structures in the generated image and reflect perceptual style patterns in images stylized by NST. They act as the implicit and entangled representation of several painterly concepts such as stroke/brush-size, -orientation or -intensity that are visible in the style image. In line with previous work [11] [13], controlling strokes is thus referred to adjusting stroke textons, where the visual representation of a stroke entirely depends on the extracted style statistics.

Stroke sizes in the generated output are sensitive to the resolution of the style image, as the Gram-based statistics over the VGG features are themselves not scale-invariant. Resizing the *style image* during training can thus be used to control the stroke size in the output image [13]. Furthermore,

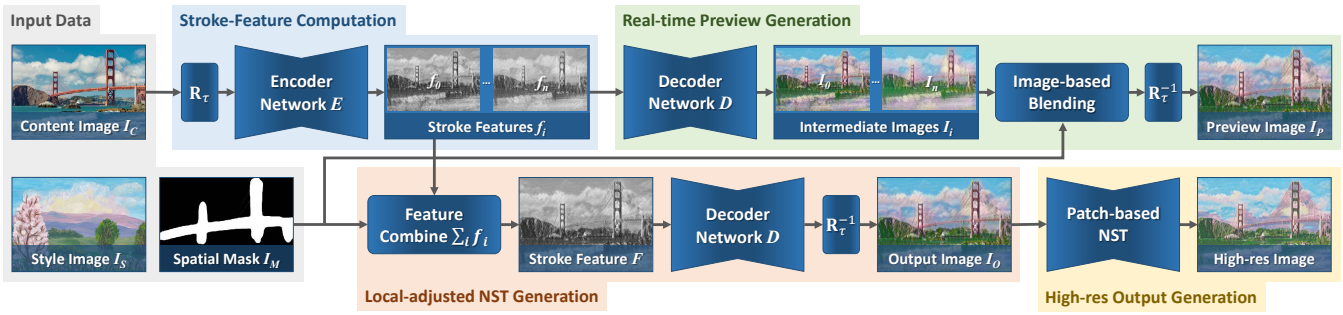


Figure 2: Schematic overview of the processing stages, components, and data flow for the presented approach.

the stroke size also strongly depends on the resolution of the *input image*, as networks, once trained with a style image, produce stroke features on a certain scale dependent on the receptive field of the network. Thus, a smaller image forwarded through the same network will produce larger strokes in the output image as the receptive field of convolutional units in the network is relatively higher. Similarly, convolutional units are also not invariant against rotation of the input, i.e., forwarding the rotated image through an object detection network and then rotating the output back produces different results than forwarding an unrotated image. In the following, we show how we make use of these inherent properties of convolutional neural networks to control multivariate aspects of the style.

B. Method Overview

Figure 2 shows an overview of our method for a controllable NST, which allows control of stroke size, orientation, and intensity on a global and local level. It comprises the following main stages:

Stroke-Feature Computation: Computes stroke features f_i for different stroke granularities based on the rotated (angle τ) content input image I_C using the content encoder network E trained on the style image I_S (Section III-D).

Real-time Preview Generation: To support real-time preview of local-adjustment results I_P, this stage performs image-based blending of the results of decoder network D (Section III-D) applied to all features f_i using the spatial mask image I_M and inverts the rotation subsequently.

Local-adjusted NST Generation: To synthesize a high-quality output image I_O, this stage performs model-space blending by combining the features f_i based on I_M into a unified stroke feature map F that is fed into the decoder D and followed-up with a rotation inversion (Section III-F).

High-resolution Output Generation: To obtain a high-resolution output, this stage performs optional upsampling of I_O using a service-based implementation of patch-based upsampling [21].

C. Parameter Mapping

Our approach combines the controllability of the stroke size and style intensity parameter (Figure 3) by mapping them to different input modalities of the proposed network. Stroke-orientation control on the other hand can be achieved through transformation of the input image and does not need any architecture considerations.

1) Stroke-Size Control: We make use of the scale dependency of receptive fields (Section III-A) in our architecture to scale stroke sizes (Figure 3). The naive approach of downscaling the image, applying a conventional fast style transfer network, and upscaling the image using a super resolution upscaling approach can achieve control over stroke sizes, as Jing *et al.* [13] show. However, the output loses sharpness and details. Therefore, the key idea of our network is to combine dynamic input scaling with extracted high-frequency details from the high-resolution input image. The stroke size parameter λ_S effectively controls the receptive field size of the stroke branch (Section III-D).

2) Stroke-Intensity Control: Based on instance normalization, Dumoulin *et al.* [22] proposed a Conditional Instance Normalization (CIN) layer that, instead of learning affine parameters, learns a set of parameters β and γ for each style. In their approach, conditioning on a style is achieved as follows:

$$z = \gamma_s \left(\frac{x - \mu(x)}{\sigma(x)} \right) + \beta_s$$

where $\mu(x)$ and $\sigma(x)$ refer to the mean and standard deviation taken across the spatial dimensions of input features x. The NST network incorporates multiple CIN layers, and sets β_s and γ_s according to the chosen style s.

Huang *et al.* [12] further show, that the parameters can be directly predicted from an input style image, without requiring to train the corresponding style, yielding an adaIN layer. We adapt the parametrization of α and β as a form of general controllability that can map continuous input variables to desired network outputs. Similar to Babaeizadeh *et al.* [23], we map style intensity (Figure 3) via the CIN layers, where the set of all CIN parameters $\Phi = \{\alpha, \beta\}$ is predicted using a linear regression $\Phi = W\lambda_I + b$, where λ_I

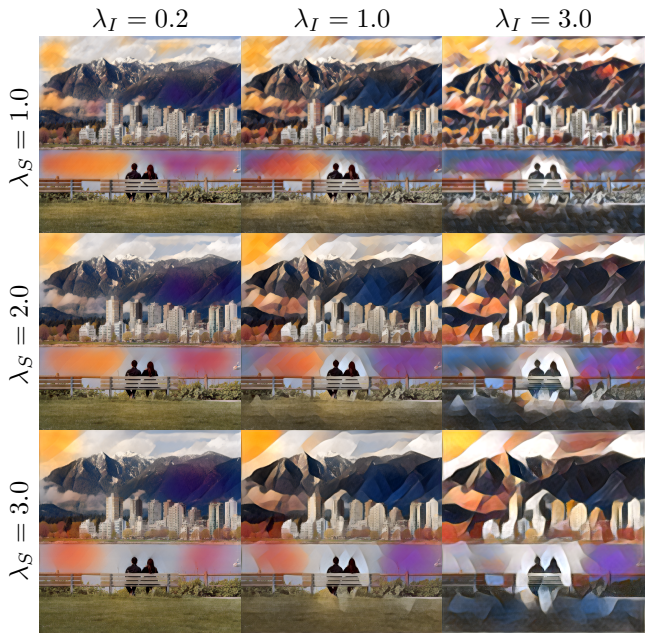


Figure 3: Variations of stroke size λ_S and style intensity λ_I.

is the input style intensity and W, b are learnable weights obtained using a fully-connected layer.

D. Network Architecture

Figure 4 shows our proposed adjustable NST network architecture. The network consists of a two-branch architecture using a Resnet-like [24] structure with residual-blocks. The dynamic/low-resolution branch is based on the fast style transfer architecture [10], with instance normalization replaced by CIN layers, and is responsible for learning the stylization operation. The high-resolution branch consists of a relatively light-weight set of layers for extracting high-frequency details from the input image. Both branches are fused together by channel concatenation before decoding the final output image. The input stroke size λ_S controls the downsampling factor on the input image, with the dynamic upsampling operation used to upsample back to the original resolution before merging features. For local editing, the encoder part E of the network is used to produce features f_i in different style scales which are blended together based on a spatial mask I_M during feature merging (c.f. Figure 2) and then jointly decoded by the network decoder D.

E. Network Training

We train our proposed network on the MS-COCO dataset [25] for 4 epochs using the Adam optimizer [26]. Images are cropped and resized to 512×512 pixels, and the pre-trained VGG-19 [9] network is used as the loss network, using style and content loss as defined by Gatys *et al.* [8], for which $relu1\_1, relu2\_1, relu3\_1, relu4\_1, relu5\_1$ are used as the style layers and $relu4\_2$ is used as the content

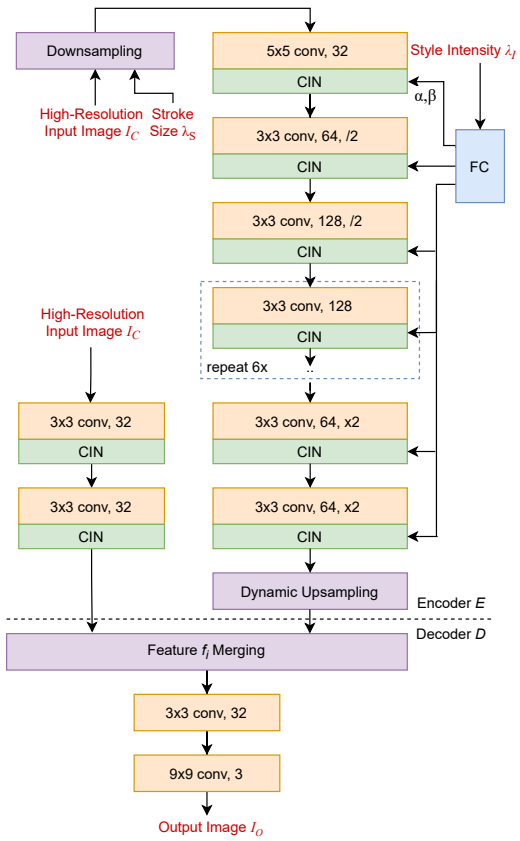


Figure 4: Adjustable NST network architecture. The encoder network E consists of a high-resolution processing branch (left) and a low-resolution branch (right), that produce features f_i for a certain stroke size. The decoder D consists of a feature merging and decoding operation. The stroke scale λ_S is used as the downsampling factor of the input image for the low-resolution branch. For each residual block (depicted in orange), kernel size, channel number, and spatial up/down sampling are shown. Residual blocks use CIN operations with parameters α, β controlled through a regression from the style-intensity parameter λ_I.

layer. During training, the input image for the low resolution branch is cycled through different factors for downsampling. We set the downsampling factors to 2 and 4 during training. During inference time, the downsampling factor can be chosen from a continuous range even if these input sizes were not observed during training. The style intensity parameter is randomly sampled from a uniform distribution $\lambda_I \in U(0, 1)$ with the corresponding style weight set to the same value.

F. Stroke Orientation

We control stroke orientation by taking advantage of the property that convolutional units are not invariant with respect to input rotation, as described in Section III-A. In particular, using the stroke rotation factor τ, a stroke in the

(a) Content/Style (b) $\tau = 0°$ rotation (c) $\tau = 45°$ rotation

Figure 5: Change of stroke orientation by content rotation.

output is rotated as follows:

$$f_i = E\left[\text{pad}\left(\mathbf{R}_\tau I_C\right)\right]$$
$$I_O = \text{crop}\left[\mathbf{R}_\tau^{-1} D\left(\sum_i f_i\right)\right]$$

where the input image I_C is rotated by τ degrees using a rotation matrix $\mathbf{R}_\tau$ and then padded back to a rectangular shape (i.e., filling whitespace resulting from the rotation) using reflection padding. The image is then fed through the encoder E to obtain stroke features f_i, which can then be blended according to a spatial mask and fed to the decoder D to obtain the output image. The rotation is then inverted and the resulting image is cropped to the original extent to obtain a stylized image where the strokes are rotated by τ (Figure 5). This method of stroke orientation adjustment is agnostic of network architecture and can be applied using any feed-forward NST method.

G. Implementation Aspects

Training and model preparation is implemented in python and using PyTorch [27]. The respective models are converted to CoreML [28] and weights are quantized to 16 bit in a pre-processing step for their use on mobile devices supporting iOS. The app implementation is based on Apple's Swift, CoreML (MTLGPUFamilyApple5), and Metal Application Programming Interfaces (APIs) for Graphics Processing Unit (GPU)-based processing of neural networks and rendering techniques, and implements the respective functionality for local adjustments. The results can be transferred to a service-based implementation of patch-based upsampling [21].

IV. User Interface

Figure 6 shows screenshots of our application illustrating the three step process to create the final output image. Thereby, the interactive image editing and enhancement workflow comprises the following steps and interface components to facilitate edits on different levels-of-control [5].

1) Selection of Content and Style Images: After selecting and loading or capturing an input content image I_C, the user is required to select an style image I_S from a list of available/trained styles (Figure 6a). Upon style selection, the application applies the respective NST model to the content image and presents a preview I_P of the resulting style transfer in real-time. This enables a user to quickly browse the available styles and to decide for a basic style for his subsequent edits.

2) Adjustment of Stroke Granularity and Rotation: After selecting a model for the global transfer, the user can control the global stroke size λ_S interactively using a slider. To allow for further control, we present the user an additional optional editing view with global style settings that enables the interactive adjustment of stoke size and rotation (Figure 6b). To implement such interactive exploration of the parameters (e.g., using a slider), results are pre-generated when entering the edit view. Pre-computation incurs a brief loading time, depending on the number of pre-computed stroke-size levels. Furthermore, using painting brush metaphors to manipulate the spatial mask I_M, the user can then locally apply different stroke-size and orientation edits. The stroke levels are blended in image space to retain interactivity during the brush process and then merged in feature space of the neural network to create seamless stroke transitions on demand (Figure 6c).

3) High-resolution Image Export: In a final step, the composition can be exported at a very high spatial resolution using patch-based upsampling [21]. After the upsampled image is received, the user can explore the result using pan and zoom gestures. If the user is satisfied with the result, the application allows storing and sharing respectively.

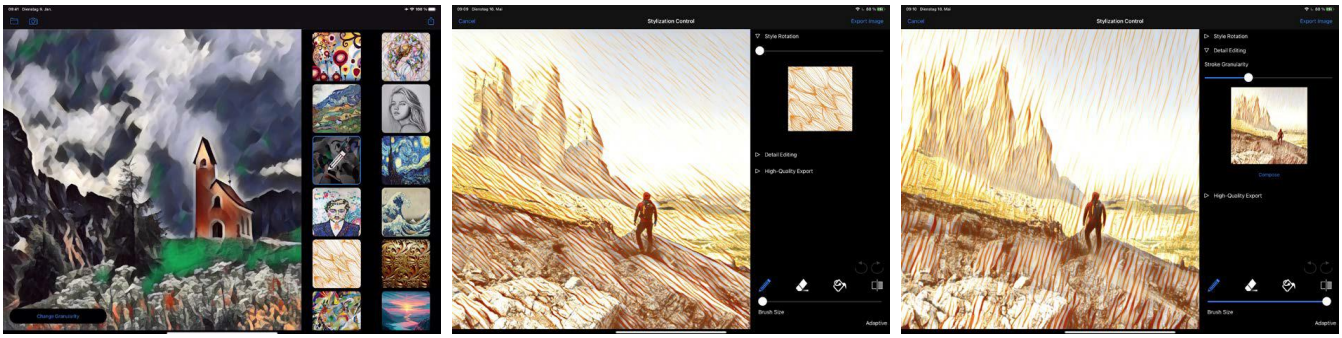
V. Results

A. Qualitative Comparison

Figure 7 shows comparison results of our architecture to the architecture of Jing *et al.* [13] as it is the most similar to our work in representing stroke size control for a single style in one feedforward style transfer network. In addition to being able to control further elements of the style such as its intensity, our adjustable architecture allows to represent a greater range of stroke size edits. This is especially true for higher resolution images, as style transfer networks are often trained on small resolutions images (usually 256×256 pixels) to reduce the computational cost, resulting in very fine and subtle style elements during inference on large images (i.e., edge length 1024 and above). Our architecture mitigates this by downsampling the input image in the stylization branch while retaining high-frequency details in the high-resolution branch. Furthermore, our architecture is able to represent consistent stroke sizes in different output image resolutions – if the input image resolution doubles the stroke size can be equally doubled. This can be important in an application when working on a lower resolution preview image and exporting to a high-resolution final image.

B. Exemplary Results

Figure 8 shows exemplary results obtained using *StyleTune* to locally retouch the hair and face of the person. Using less



(a) Content and style image selection (b) Style transfer without stroke rotation. (c) Local retouches with stroke rotation.

Figure 6: Screenshots of *StyleTune*: After selecting a style, it can be rotated globally. The user can apply different style sizes to different parts of the image using *brush metaphors* (all content images from © Shutterstock, used with permission).

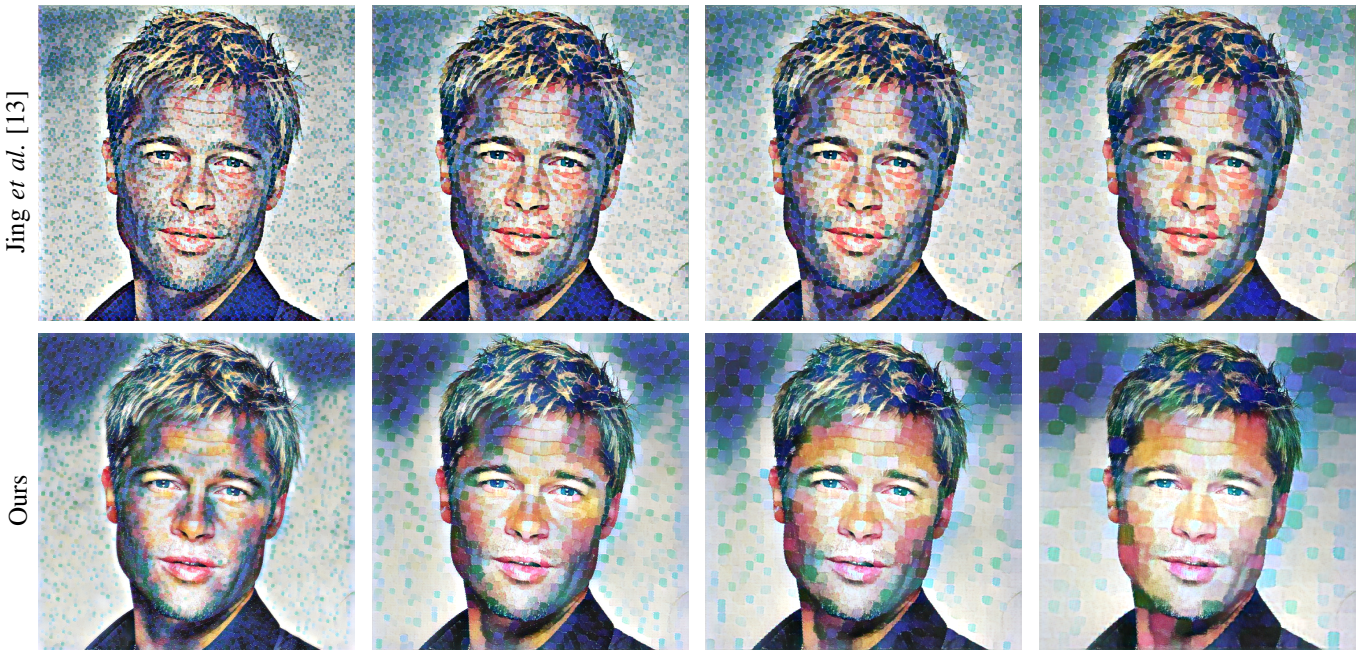


Figure 7: Comparing global stroke size adjustment from lowest to highest level. Our approach can represent a higher range of stroke sizes than the adaptive network of Jing *et al.* [13], as (in theory) arbitrarily large stroke sizes λ_I can be set. In practice, very large stroke sizes tend to lose sharpness, as the style-branch resolution decreases.

obstructive lines in the face yield a visually more pleasing result and the lighter color tone directs the gaze to the center of the image. Furthermore, oriented strokes along flow lines of the hair in Figure 8c improves the visual separation of foreground and background and lends the image more depth. The tools provided via *StyleTune* can thus be used to art-direct semiotic aspects of the style with local guidance, a step towards *semiotics-based loss functions* [7].

C. Performance Considerations

The performance of the proposed architecture depends mainly on the stroke-size setting, as larger strokes are generated by running through a downsampled style branch. As Figure 9 shows, for stroke sizes $\lambda_S \geq 2.0$, our network is faster than the adaptive network of Jing *et al.* [13] and the similarly performing network of Johnson *et al.* [10]. For comparability to the performance of other NST methods, the timings are shown for a desktop system similar to [3].

For execution on mobile devices, we tested our application using an iPad Pro 3$^{\text{rd}}$ generation equipped with an Apple A12X Bionic and 4 GB Random Access Memory (RAM). We use input images of 1024×1024 pixels resolution. The application of global style transfer and the image-based blending operation run in real-time without notable latency.

(a) Content and Mask (b) Global Transfer and Style (c) Local Adjustments (d) Global Transfer and Style (e) Local Adjustments

Figure 8: Comparison between global style transfer and locally retouched versions produced using *StyleTune*.

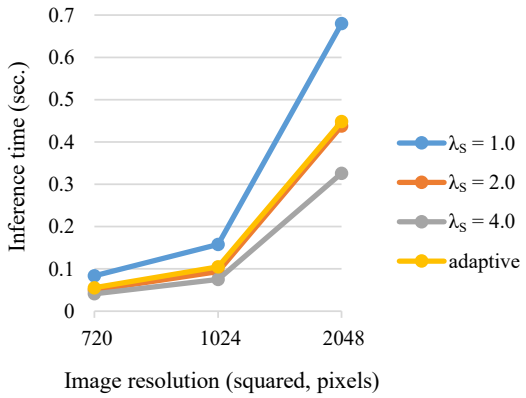


Figure 9: Performance comparison for different stroke sizes of our adjustable architecture and the adaptive network [13]. Tests were performed on desktop PC using a Nvidia GTX 1080Ti GPU and averaged over 100 processed images.

The style-encoder network for stroke-size pre-generation required approx. 5 sec for 10 stroke sizes and the decoder network performing model-space blending approx. 3 sec. A lower number of pre-computed strokes can be sufficient for many styles when editing locally, however for adjusting the global granularity and orientation sliders, 10 levels and more empirically provide a visually smooth transition between scales. The optional server-based upsampling requires 1 min for 256 Mpix. The overall on-device memory usage comprises approx. 900 MB.

D. Limitations

While *StyleTune* enables more degrees of artistic freedom for style transfer, there are still some limitations to overcome. Our proposed architecture achieves a greater range of flexibility for global edits, however, strokes from one stroke-size level are generally not consistently placed at the same location in other stroke-size levels. This represents a limitation on local editing scenarios where strokes are expected to consistently flow between different stroke sizes.

To remedy this, we also implement the adaptive stroke architecture of Jing *et al.* [13] in *StyleTune* as an option for fine detail control. Further, before locally applying edits, there is a brush pre-generation step for every stroke size, which incurs a loading time that is dependent on the number of brush sizes and any stroke orientation changes require a new round of pre-computation for stroke size previews. Finally, the patch-based upsampling step alters global appearance in ways that may not be intended by the user and has a high execution time.

VI. Conclusions & Future Work

This paper introduces an approach for multivariate control over fast style transfer. Our method is the first to enable control over stroke size, style intensity and stroke orientation with a single model. We demonstrate the real-world applicability of our idea by implementing a mobile app for finegrained global and local control over these aspects using our proposed network. Our app implements an editing pipeline to enable both interactive adjusting and retouching of results as well as very high resolution exports using style-guided upsampling. Our work is a step towards making NST a useful tool for art-directed image stylization for casual and professional users, however, still some limitations remain to be addressed.

As future work, we plan to further explore the integration of different style scales into patch-based upsampling by adding a neural representation of stroke size and implement the upsampling method on mobile devices directly, to provide interactive feedback by continuously updating the visual results during editing.

Acknowledgment

This work was partially funded by the German Federal Ministry of Education and Research (BMBF) through grants 01IS18092 (“mdViPro”) and 01IS19006 (“KI-LAB-ITSE”).

References

[1] A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Nießner *et al.*, “State of the Art on Neural Rendering,” in *Computer Graphics Forum*, vol. 39, no. 2. Wiley Online Library, 2020, pp. 701–727.

[2] G. Amato, M. Behrmann, F. Bimbot, B. Caramiaux, F. Falchi, A. Garcia, J. Geurts, J. Gibert, G. Gravier, H. Holken *et al.*, "AI in the media and creative industries," *arXiv preprint arXiv:1905.04175*, 2019.

[3] Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, and M. Song, "Neural Style Transfer: A Review," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 11, pp. 3365–3385, 2020.

[4] D. Dapkus, "How to transfer styles to images with Adobe Photoshop." [Online]. Available: https://creativecloud.adobe.com/de/discover/article/how-to-transfer-styles-to-images-with-adobe-photoshop

[5] T. Isenberg, "Interactive NPAR: What Type of Tools Should We Create?" in *Proc. NPAR*, ser. Expressive '16. Goslar, DEU: Eurographics Association, 2016, p. 89–96.

[6] M. Klingbeil, S. Pasewaldt, A. Semmo, and J. Döllner, "Challenges in User Experience Design of Image Filtering Apps," ser. Proceedings SIGGRAPH ASIA Mobile Graphics and Interactive Applications. New York: ACM, 2017.

[7] A. Semmo, T. Isenberg, and J. Döllner, "Neural Style Transfer: A Paradigm Shift for Image-based Artistic Rendering?" ser. Proceedings International Symposium on Non-Photorealistic Animation and Rendering (NPAR). New York: ACM, 7 2017, pp. 5:1–5:13.

[8] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image Style Transfer Using Convolutional Neural Networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Jun. 2016, pp. 2414–2423.

[9] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *3rd International Conference on Learning Representations, ICLR 2015*, San Diego, CA, USA, May 2015.

[10] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution," in *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II*, ser. Lecture Notes in Computer Science, vol. 9906. Springer, 2016, pp. 694–711.

[11] H. Zhang and K. Dana, "Multi-style Generative Network for Real-Time Transfer," in *Computer Vision – ECCV 2018 Workshops*. Springer International Publishing, 2019, pp. 349–365.

[12] X. Huang and S. Belongie, "Arbitrary Style Transfer in Real-Time With Adaptive Instance Normalization," in *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE Computer Society, Oct 2017, pp. 1510–1519.

[13] Y. Jing, Y. Liu, Y. Yang, Z. Feng, Y. Yu, D. Tao, and M. Song, "Stroke Controllable Fast Style Transfer with Adaptive Receptive Fields," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018, pp. 244–260.

[14] M. Reimann, M. Klingbeil, S. Pasewaldt, A. Semmo, M. Trapp, and J. Döllner, "MaeSTrO: A Mobile App for Style Transfer Orchestration Using Neural Networks," in *2018 International Conference on Cyberworlds, CW 2018, Singapore, October 3-5, 2018*. IEEE Computer Society, 2018, pp. 9–16.

[15] A. Moiseenkov, O. Poyaganov, I. Frolov, and A. Usoltsev, "Prisma," https://prisma-ai.com/, 2021, version: 4.3.4.

[16] S. Pasewaldt, A. Semmo, J. Döllner, and F. Schlegel, "BeCasso: Artistic Image Processing and Editing on Mobile Devices," in *SIGGRAPH ASIA 2016, Macao, December 5-8, 2016 - Mobile Graphics and Interactive Applications*. ACM, 2016, p. 14:1.

[17] L. A. Gatys, A. S. Ecker, M. Bethge, A. Hertzmann, and E. Shechtman, "Controlling Perceptual Factors in Neural Style Transfer," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 3730–3738.

[18] H. Wu, Z. Sun, Y. Zhang, and Q. Li, "Direction-aware neural style transfer with texture enhancement," *Neurocomputing*, vol. 370, pp. 39–55, 2019.

[19] Y. Yao, J. Ren, X. Xie, W. Liu, Y. Liu, and J. Wang, "Attention-aware multi-stroke style transfer."

[20] S.-C. Zhu, C.-E. Guo, Y. Wang, and Z. Xu, "What are textons?" *International Journal of Computer Vision*, vol. 62, no. 1, pp. 121–143, 2005.

[21] O. Texler, J. Fišer, M. Lukáč, J. Lu, E. Shechtman, and D. Sýkora, "Enhancing Neural Style Transfer Using Patch-Based Synthesis," in *Proc. NPAR*, ser. Expressive '19. Goslar, DEU: Eurographics Association, 2019, p. 43–50.

[22] V. Dumoulin, J. Shlens, and M. Kudlur, "A Learned Representation For Artistic Style," *ICLR*, 2017.

[23] M. Babaeizadeh and G. Ghiasi, "Adjustable Real-time Style Transfer," in *8th International Conference on Learning Representations, ICLR 2020*, Apr. 2020.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[25] 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 *Proc. ECCV*. Cham, Switzerland: Springer International, 2014, pp. 740–755.

[26] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[27] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[28] O. Marques, *Machine Learning with Core ML*. Cham: Springer International Publishing, 2020, pp. 29–40.

[29] A. Semmo, T. Dürschmid, M. Trapp, M. Klingbeil, J. Döllner, and S. Pasewaldt, "Interactive Image Filtering with Multiple Levels-of-control on Mobile Devices," in *Proceedings SIGGRAPH ASIA Mobile Graphics and Interactive Applications (MGIA)*. New York: ACM, 2016, pp. 2:1–2:8.

[30] S. Bakhshi, D. A. Shamma, L. Kennedy, and E. Gilbert, "Why we filter our photos and how it impacts engagement," in *Proceedings of the Ninth International Conference on Web and Social Media ICWSM*. AAAI Press, 2015, pp. 12–21.