The Visual Computer
https://doi.org/10.1007/s00371-022-02518-x

ORIGINAL ARTICLE

Controlling strokes in fast neural style transfer using content transforms

Max Reimann[1] · Benito Buchheim[1] · Amir Semmo[2] · Jürgen Döllner[1] · Matthias Trapp[1]

Accepted: 3 May 2022

Abstract
Fast style transfer methods have recently gained popularity in art-related applications as they make a generalized real-time stylization of images practicable. However, they are mostly limited to one-shot stylizations concerning the interactive adjustment of style elements. In particular, the expressive control over stroke sizes or stroke orientations remains an open challenge. To this end, we propose a novel stroke-adjustable fast style transfer network that enables simultaneous control over the stroke size and intensity, and allows a wider range of expressive editing than current approaches by utilizing the scale-variance of convolutional neural networks. Furthermore, we introduce a network-agnostic approach for style-element editing by applying reversible input transformations that can adjust strokes in the stylized output. At this, stroke orientations can be adjusted, and warping-based effects can be applied to stylistic elements, such as swirls or waves. To demonstrate the real-world applicability of our approach, we present *StyleTune*, a mobile app for interactive editing of neural style transfers at multiple levels of control. Our app allows stroke adjustments on a global and local level. It furthermore implements an on-device patch-based upsampling step that enables users to achieve results with high output fidelity and resolutions of more than 20 megapixels. Our approach allows users to art-direct their creations and achieve results that are not possible with current style transfer applications.

1 Introduction

Image-based artistic rendering methods stylize images with expressive stylistic effects [37], often resembling a real-world artistic style. Typically, these methods have been engineered in the form of style-specific algorithms [24]. With the advent of deep network-based learning, however, automatically learning new styles by example has enabled impressive results [18,40]. Therefore, machine learning-based techniques for image stylization have received significant attention in both research and end-user targeted applications. In particular, they have become essential tools in mobile applications for easy-to-use, casual creativity-targeted image editing and filtering [1].

In this domain, Neural Style Transfers (NSTs) have gained large popularity, primarily due to their ability to transfer artistic styles of a reference image to target images, and to perform on resource-constrained mobile devices. NST was introduced as a non-interactive, one-shot stylization technique, mainly due to its underlying CNN being a black-box. While NST has been applied both in casual and professional image-editing applications such as Photoshop [7], they have thus been mostly limited to so-called one-click solutions. At this, pre-trained styles are typically applied uniformly to the input image without providing lower-level control that goes beyond one-shot stylizations, thus limiting the artistic freedom and expression often sought by artists [16,37] and prosumers of image stylization applications [22]. So far, only a few approaches for interactive low-level control over NST have been proposed, and these often only consider univariate style-element adjustments, thus making composable and individualistic editing workflows impracticable. In particular, current applications lack sophisticated control over perceptual elements of a style, such as stroke directions or stroke granularities.

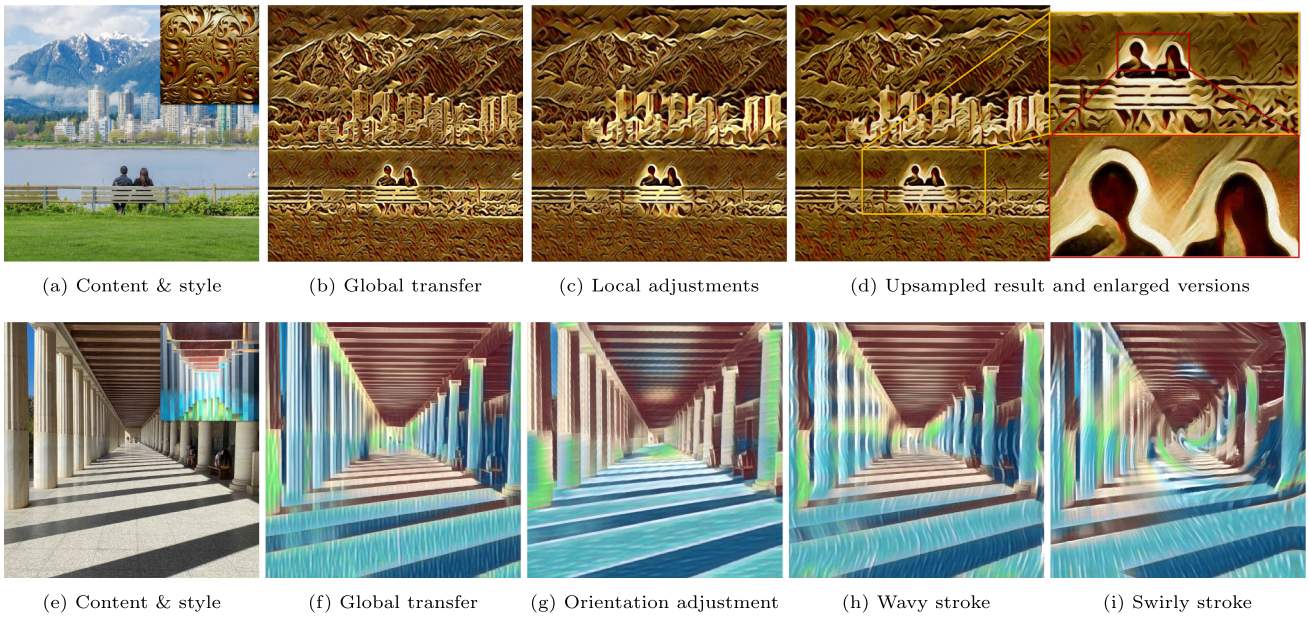
✉ Max Reimann
max.reimann@hpi.uni-potsdam.de

Amir Semmo
amir.semmo@digitalmasterpieces.com

[1] Hasso Plattner Institute, University of Potsdam, Potsdam, Germany

[2] DigitalMasterpieces GmbH, Potsdam, Germany

Published online: 08 June 2022



(a) Content & style (b) Global transfer (c) Local adjustments (d) Upsampled result and enlarged versions

(e) Content & style (f) Global transfer (g) Orientation adjustment (h) Wavy stroke (i) Swirly stroke

Fig. 1 Multi-level adjustable style transfer. A global stylization **b** is locally adjusted by increasing stroke size and intensity (**c**). A high-fidelity version **d** is then generated by style-guided upsampling of **c** by a factor of 3 to a resolution of 3200 × 3200 pixels. Reversible content transformations are used to adjust stroke orientation (**g**), and can induce warp-based effects such as wavy (**h**) or swirly (**i**) stroke patterns. (content image (**a**) © Shutterstock, used with permission)

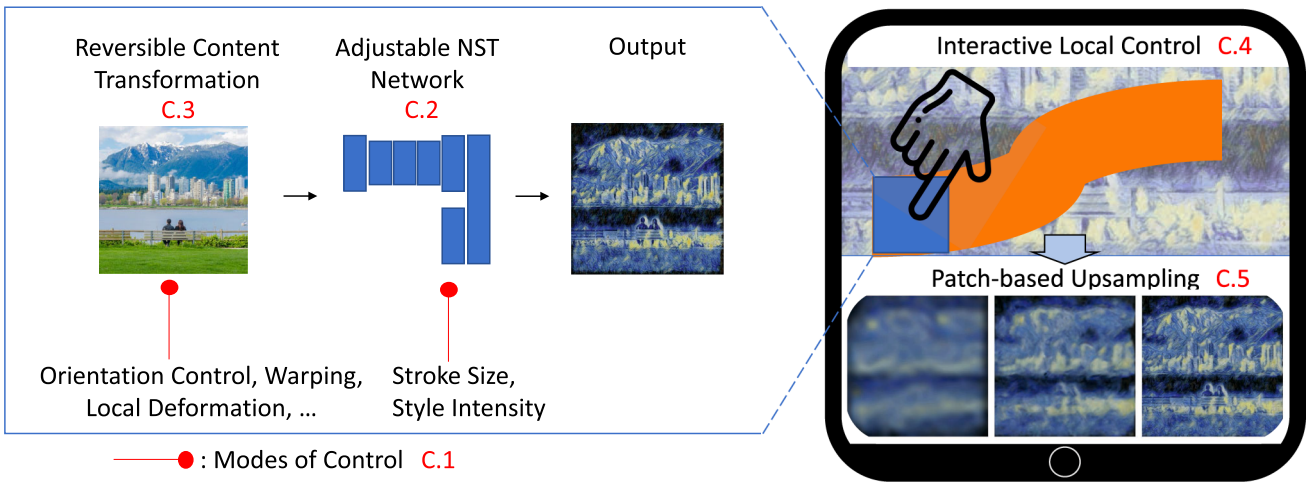


Fig. 2 Overview of our system and contributions. Our method introduces several modes of NST control through content transformations and an adjustable NST network. The interactivity of our method is demonstrated by a mobile application that achieves high-fidelity results by combining local control with patch-based upsampling

In this work, we present an approach[1] for multivariate editing of NSTs using a novel feedforward style transfer network for *global* and *local* control over style elements. Using this approach, style elements such as stroke granularity, orientation, or intensity can be interactively adjusted and composed, which essentially evolve NSTs into art-directable image-stylization tools. Furthermore, we introduce the concept of *reversible* input transformations that enable complex and individualistic image style element adjustments (e.g., element warping) without the need for architecture adjustments or network retraining (Fig. 1). We validate our concept by presenting *StyleTune*, a mobile app for interactive editing of NSTs. In addition to interactive multivariate control, our app implements patch-based upsampling of edited results to create high-resolution output renditions using on-device

[1] https://github.com/MaxReimann/stroke-adjustable-nst-transforms.

processing (Fig. 1). An overview of the above system, components and contributions is shown in Fig. 2.

To summarize, this paper makes the following contributions:

- C.1 A method for controlling multivariate aspects of style transfer, such as stroke orientation, stroke size, or style intensity with a single neural network.
- C.2 A novel two-stream NST network architecture for efficient and highly variable stroke-size control.
- C.3 A general, network architecture-independent method for global and local style element control in feedforward NSTs using reversible input transformations.
- C.4 A mobile demonstrator app for interactive level-of-control editing of multivariate NSTs.
- C.5 An implementation of on-device processing for patch-based upsampling of style transfer results for high-fidelity outputs of 20 Megapixels (Mpixs).

This article represents an extension to our Cyberworlds 2021 conference paper [34]. In addition to the contributions C.1, C.2 and C.4 introduced in [34], we extend this work by generalized reversible transformations (C.3—see Sect. 4) and a discussion of an on-device implementation of patch-based upsampling as a case study (C.5—see Sect. 5.3). Furthermore, this article provides an extended evaluation (Sects. 7.1 and 7.3) including an ablation study for the network architecture (Sect. 3.5).

The remainder of this paper is structured as follows. Section 2 reviews related and previous work on controllable NST approaches. Section 3 describes our adjustable network architecture and its modalities of control. Section 4 introduces the concept of reversible editing transformations. Section 5 gives an overview of our interactive editing pipeline, and outlines implementation aspects of our mobile app. Section 6 briefly explains the structure and capabilities of the user interface provided by our mobile app. Section 7 shows and discusses exemplary results and application examples. Finally, Sect. 8 concludes this paper and provides a prospect on future work.

2 Related work

NST, introduced in the seminal work of Gatys et al. [10], optimizes feature statistics of a target image to match content and style statistics which are extracted from the activations of a Convolutional Neural Network (CNN). In particular, the Gram matrix over a set of features of the VGG [38] network is used as a statistical representation of style, and the difference between target and style image statistics is minimized during optimization. However, this optimization process is computationally expensive and thus unsuitable for interactive and mobile environments. To alleviate this limitation, several approaches for fast style transfer have been published [18]. In general, these methods train an image transformation CNN to directly generate outputs in a forward pass of the network, such as the popular fast style transfer network architecture introduced by Johnson et al. [19]. While this approach is only able to represent a single style per trained network, follow-up works introduced architectures that are able to represent multiple styles [48] or arbitrary styles [13,15,25] using a single network.

However, increasing the style-representation capacity of a network generally represents a trade-off in quality, memory, and run-time performance versus single-style networks [17,35]. In mobile applications, single-style-per-network approaches have thus remained the most suitable and prevalent network architecture (e.g., refer to Prisma [31] and Becasso [32]). While most style transfer methods seek to achieve plausible results on a global level without requiring user interaction, several approaches enable control over perceptual elements of the output to a varying degree [18]. An overarching goal has been to either directly or indirectly control semiotic aspects originating from artwork production [37]. Gatys et al. [10] demonstrate that adjusting the weighting of loss terms in the optimization method can control the style content trade-off, and adding a histogram loss term can achieve a form of color control [11]. Wu et al. [43] add a direction-aware loss term to control the orientation of strokes.

While these optimization-based approaches can flexibly be extended to other forms of control by adding further loss terms, control in feedforward network-based approaches is generally less flexible. It is either achieved by explicitly adapting the network architecture to allow for a particular degree of control or by taking advantage of an inherent property of the network. For example, arbitrary style transfer networks such as adaptive Instance Normalization (adaIN) [15], inherently allow control over stroke sizes by resizing the input style image, whereas the stroke size is fixed in single-style networks [19].

For explicit control over stroke sizes, Jing et al. [17] propose a multibranch network architecture that is trained on multiple, discrete stroke sizes that are then encoded in a stroke pyramid in the network. Yang et al. [45] extend the stroke pyramid to arbitrary style transfer. Yao et al. [46] propose an arbitrary style transfer method that furthermore uses an attention-based mechanism for multi-stroke transfer. Reimann et al. [36] achieve interactive control over the spatial application of different styles using a multi-style network with an additional consistency regularizer loss. Similarly, our approach also extends the architecture of a feedforward style transfer network with runtime controls. However, in contrast to previous approaches, our architecture enables multiple, complementary modes of control and the combina-

tion thereof while being more efficient with respect to runtime and memory consumption.

3 Stroke-adjustable network

We propose an adjustable style transfer network for simultaneous control of stroke size and style intensity. Our network is trained to reproduce a single style to achieve both high quality and performance on mobile devices. In the following, we analyze the underlying principles of our method and give details on the architecture and training approach.

3.1 Preliminary analysis

One of the key differences between NST and algorithmic-based stylization techniques is that elements of the style, such as textures and strokes, are not explicitly defined but rather implicitly learned from matching Gram matrices in the VGG [38] feature space. The generated microstructures that reflect perceptual elements of the style are denoted as stroke textons [49]. They serve as an entangled and implicit representation of several painterly concepts, such as brush size, orientation, or applied pressure, which underlie the artistic formation process of the input style image. Controlling strokes thus refers to the local adjustment of stroke textons [48] [17], for example, through affine transformations, deformations, or intensity changes. The visual representation of strokes is determined solely by the learned style statistics.

Previous work on stroke size control in NST [17] uses the scale sensitivity of CNNs to encode different stroke sizes by using multiple scaled versions of a style image during training. As the receptive field size of neurons in the VGG loss network varies with input resolution, the extracted feature statistics reflect the stroke texton scale of the style image. Similarly, also the receptive field of the style transfer network influences the stroke size; running inference on larger images will thus result in smaller strokes in the output.

Similar to the scale variance, CNNs are also not invariant against the rotation of the input. For example, forwarding a rotated image through an image recognition network can yield different results than using the unrotated input, as neurons have been trained to activate on a particular orientation. Analogously, when passing a rotated image through a style transfer network and rotating the output back to its original orientation, strokes in the output will appear to be rotated by the same amount. Our proposed method makes use of the scale- and rotation-sensitivity of CNNs to control multivariate aspects of feedforward style transfer.

3.2 Parameter mapping

Stroke size and style intensity control are combined in the proposed network architecture by mapping the parameters to different input modalities. We present this parameter mapping (contribution C.1) in the following. Stroke adjustments from reversible transformations (Sect. 4), such as stroke orientation control, on the other hand, are network-independent and thus do not require any architecture considerations.

3.2.1 Stroke-size control

Our architecture makes use of the scale dependency of receptive fields (Sect. 3.1) by predicting on dynamic input resolutions dependent on the desired stroke size. While the naïve approach of downsampling an image according to a stroke size factor λ_S, applying a style transfer and upsampling back to the original resolution works in principle, the output loses details and sharpness, as Jing et al. [17] show in an ablation study. The key idea (C.2) of our proposed architecture is therefore to combine dynamic input scaling with high-frequency details that are extracted from the original input image (refer to Sect. 3.3). The stroke branch hereby operates on the input—dynamically downscaled by a user-defined factor λ_S—to effectively control the size of the receptive field, while the high-frequency branch input resolution remains fixed. Larger downscaling λ_S generate larger, more prominent strokes in the output image (Fig. 3).

3.2.2 Style-intensity control

The style intensity parameter is mapped to feature normalization layers parameters, similar to approaches for arbitrary style transfer based on instance normalization [8,15]. Specifically, Dumoulin et al. [8] proposed a Conditional Instance Normalization (CIN) layer that learns separate parameters β_s and γ_s for each style s:

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

where $\mu(x)$ and $\sigma(x)$ refer to the mean and standard deviation taken across the spatial dimensions of input features x. These CIN layers are used throughout the network, β_s and γ_s are then selected at runtime to generate a particular style. Further, Huang et al. [15] show that these parameters can be adaptively extracted from an arbitrary style image at inference time, without requiring to previously train the specific style, yielding adaIN layer. Keras et al. [20] further show that the adaIN generalizes to a wider range of tasks as its parameterization can be used in Generative Adversarial Networks (GANs) to control facial feature generation.

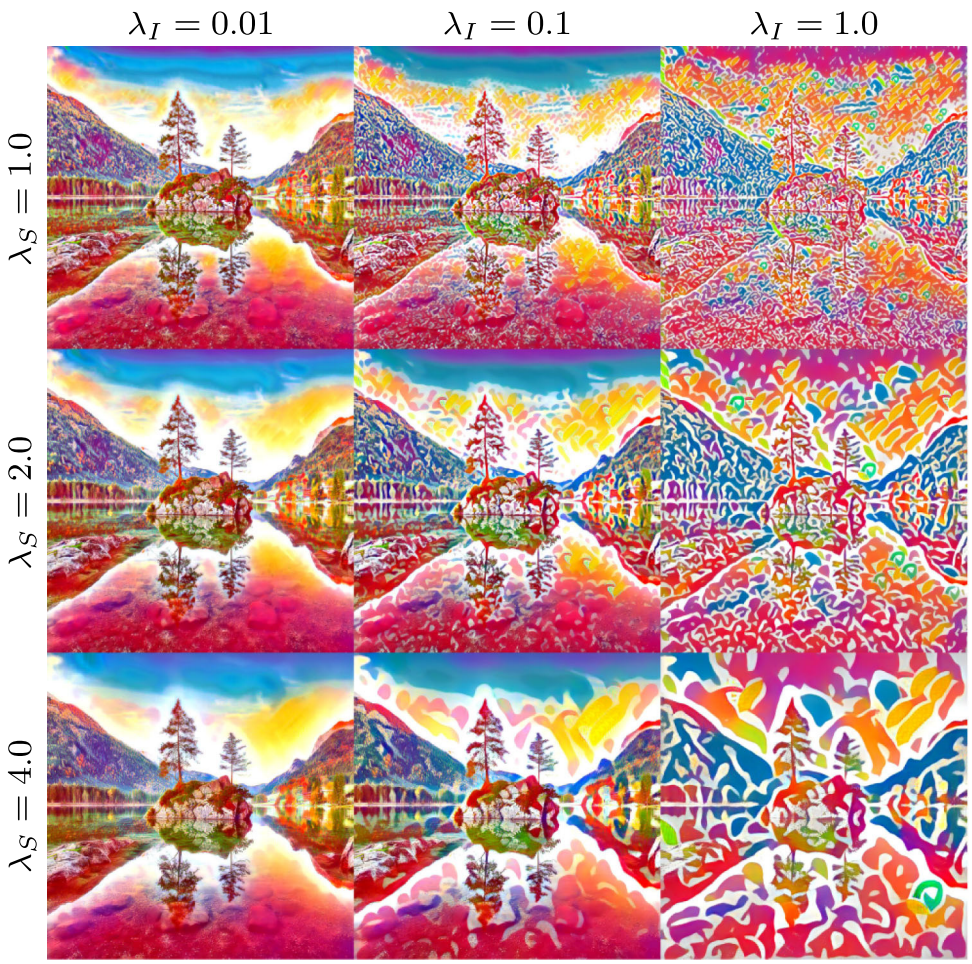


Fig. 3 Results obtained by variations of stroke size λ_S and style intensity λ_I parameters

To represent style intensity in the network, similar to Babaeizadeh et al. [2], we predict the set of all CIN parameters $\Phi = \{\alpha_1, \beta_1, \alpha_2, \beta_2, \ldots\}$ from a given style intensity factor λ_I. Hereby, Φ is linearly regressed as:

$$\Phi = W\lambda_I + b,$$

where weights W and biases b learned during network training. Increasing the style intensity $\lambda_I \in [0, 1]$ generally yields more visible texture marks, stronger abstraction of content and more prominent strokes in the output (Fig. 3).

3.3 Network architecture

Our proposed adjustable NST network architecture is shown in Fig. 4. The network encoder consists of two branches, a dynamic/low resolution branch for learning the stylization operation and a high-resolution branch for high-frequency detail extraction, the outputs of both branches are then combined in a feature merging module. The low-resolution input branch is based on the fast style transfer architecture of Johnson et al. [19] using residual-blocks [14], where instance normalization layers are replaced by CIN layers for style intensity control. Inputs to the low-resolution branch are downsampled by the stroke size factor λ_S, outputs of the branch are then dynamically upsampled back to the original resolution. The branch that receives the high-resolution image consists of a set of relatively lightweight layers that extract high-frequency details. Outputs of both branches are merged together by concatenating features and passing them through two convolutional layers. While more sophisticated, bilateral methods of merging the information streams are possible, they do not necessarily yield better results, as we show in an ablation study (Sect. 3.5).

3.4 Network training

During network training, we use the style loss $\mathcal{L}_s$ and content loss $\mathcal{L}_c$ as defined by Gatys et al. [10] over layers of a VGG-19 network [38] pre-trained on ImageNet. The network is trained on the MS-COCO dataset [28] (cropped and rescaled to 512 × 512 pixels) for 4 epochs using the Adam optimizer [21]. To learn different stroke sizes, the stroke size factor λ_s that determines the downsampling of the input

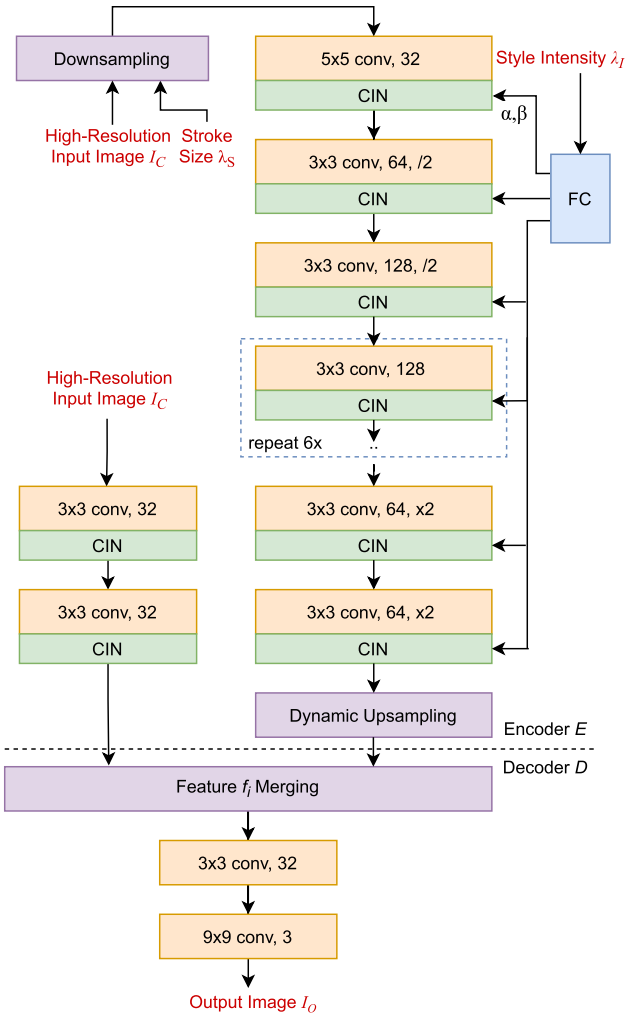


Fig. 4 Adjustable style transfer network architecture. The encoder E processes a low-resolution version—downsampled by stroke size λ_S—and high-resolution version of the input content image I_C in two separate branches. Resulting features f_i are merged in the decoder D. Residual blocks (depicted in orange, showing kernel size, channels and scaling factor) use CIN layers, where normalization constants α, β are predicted from the style-intensity factor λ_I using a fully-connected layer

image is alternated between factor 2 and 4 and the corresponding style image is scaled by the same factor. While the network only observes discrete scaling steps during training, λ_s can be chosen from a continuous range during inference. Furthermore, style intensity is sampled from a uniform distribution $\lambda_I \in U(0, 1)$ during training, and the style loss term is weighted by the same amount, i.e., $\mathcal{L} = \mathcal{L}_c + \lambda_I \mathcal{L}_s$. Model training is implemented using PyTorch [33].

3.5 Feature merging ablation

Merging high-resolution and low-resolution information streams can be formulated as learning joint bilateral upsampling [23], i.e., high-resolution image features act as local detail guidance to the outputs of the stylization branch. Several methods for joint information filtering in CNNs have been published, such as deep joint filtering [26] or Deep Guided Filters (DGF) [44], which is also evaluated on photographic style transfer, as well as pixel adaptive convolution [39]. To compare to our baseline, we trained our network using these three methods in the feature merging module, respectively. Only DGF were able to execute at an acceptable resolution for style transfer ($\geq 1024^2$ pixels), especially with respect to the memory limitations of mobile devices. In Fig. 5, we compare baseline concatenation to the DGF [44]. The convolutional DGF_b^c variant is applied with the high-resolution image as guidance (Fig. 5b) and the style image as guidance (Fig. 5c). We also compare to the DGF^c variant with learned guidance maps (Fig. 5d). Although the DGF upsampling methods can transfer colors and coarse structures from the candy style image, they are not able to produce complex style elements. Given the additional runtime penalty they incur, we thus find that simple concatenation of features (Fig. 5a) yields results with more expressiveness than sophisticated bilateral upsampling methods such as DGF.

4 Reversible transformations

As CNNs are not invariant against input transformations such as scale and rotation, these could be used as a mechanism to manipulate style transfers, as discussed in Sect. 3.1. Specifically, we hypothesize that there is a set of *reversible* geometric image transformations that can modulate stroke textons by application to the input image and reversal in the style transfer output (contribution C.3). Editing using reversible transformations is formally defined as

$$I_O = R^{-1} T(R I_C),$$

where T is a style transfer network and R is the reversible transformation. Note that T is not constrained to the network architecture introduced in Sect. 3 as it relies solely on image manipulation to the network input and output images, and can thus be applied to *any* feedforward style transfer network. Modulation of outputs is only possible if the images transformed using R create differing neuron responses in T compared to their untransformed inputs. For instance, for affine global transformations R, the neuron responses can only be affected by rotation, scaling and shearing, while translation or mirroring, on the other hand, is of no effect due to the shift invariance of CNNs. In addition to affine transformations, nonlinear global and local transformations of structure can be used for output modulation. As such, in the following we further describe stroke rotation as an exam-

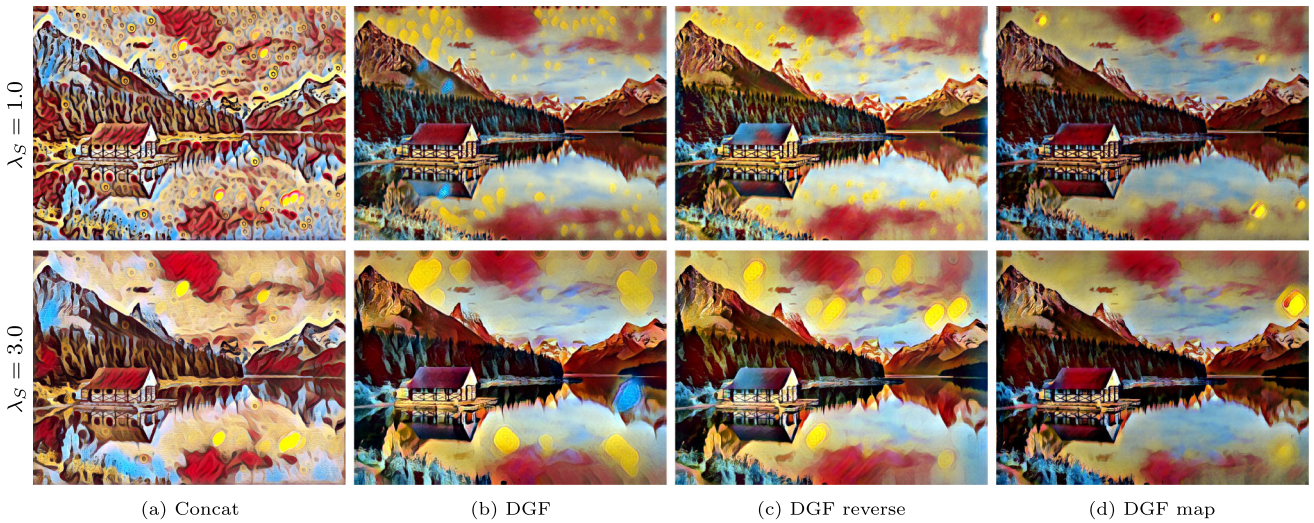


Fig. 5 Effect of different branch-merging operations on scale-adjusted transfer by the Candy style example

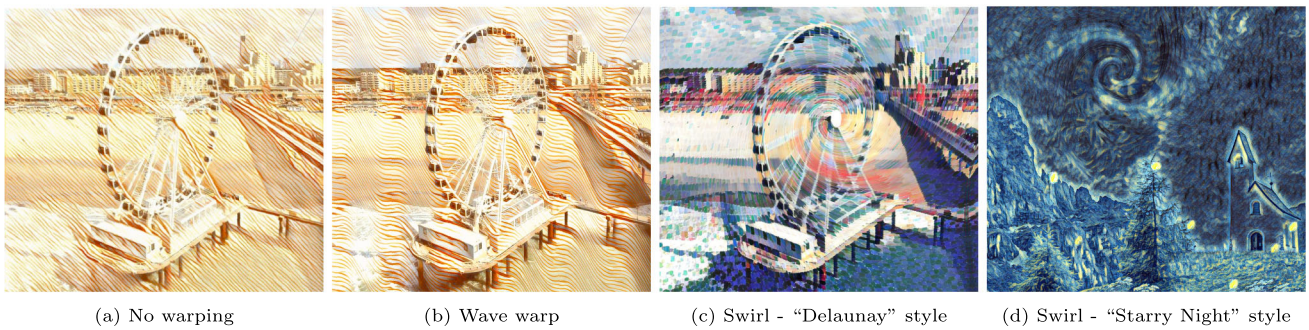


Fig. 6 Reversible global warping can be used for stroke manipulation. In **b**, wavy strokes are created by reversible sinusoidal warping. In **c**, **d**, a stroke swirl is created by using a reversible swirl warp

Fig. 7 Stroke orientation change by a reversible content rotation

ple of a global affine transformation and global and local (non-parametric) image warpings as examples of reversible local structure transformations (Fig. 6).

4.1 Stroke orientation

As CNNs are not rotationally invariant, reversible input rotations can be used to adjust stroke texton orientations. Thus, the reversible transformation R is a rotation matrix $\mathbf{R}_\tau$ around angle τ, with cropping and padding applied as necessary. To achieve a local orientation editing given our adjustable network (Sect. 3), strokes in the output image can be oriented with respect to angles τ_i for i different local orientation edits as follows:

$$\begin{aligned} f_i &= E\left[\text{pad}\left(\mathbf{R}_{\tau_i} I_\text{C}\right)\right] \\ I_\text{O} &= \text{crop}\left[D\left(\textstyle\sum_i \mathbf{R}_{\tau_i}^{-1} f_i\right)\right], \end{aligned}$$

where the input image I_C is rotated by τ_i degrees using $\mathbf{R}_{\tau_i}$. Whitespace resulting from the rotation is filled using reflection padding. To obtain stroke features f_i for a particular rotation, this image is then passed through the encoder E. The induced rotation by τ is then reverted ($\hat{f}_i = \mathbf{R}_{\tau_i}^{-1} f_i$). Features $\hat{f}_i$ of different stroke orientations can then be blended using a spatial mask and passed through the decoder D to obtain a stylized image I_t, the result is cropped back to the original extent. In the resulting image I_O, stroke textons are thus oriented by angle τ (Fig. 7).

Fig. 8 Perspective adjustment of style elements using thin-plate spline warping. Users can adjust the warp using landmark mappings (green dots)

4.2 Global warping

A global pixel mapping is a form of image warping, that relates the source image coordinates u, v to destination coordinates x, y using a global, parametric function R.

For example, a sinusoidal wave function can be applied to the style elements (c.f. Fig. 6b) using the following mapping:

$$x(u,v) = u + a\sin\left(\frac{2\pi v}{p}\right) + 2s\pi \quad \text{and} \quad y(u,v) = v,$$

where a is the amplitude, p is the period, and s is the phase shift. Interpolation of parameters over time can be used to create animated outputs. Inversion of the mapping is achieved by $x^{-1}(u,v) = 2u - x(u,v)$. Similarly, other mapping functions such as swirl (c.f. Fig. 6c) can be used to transform style elements. Warping functions thus posess a high degree of controllability over geometric arrangement of strokes. Using our proposed adjustable network (Sect. 3) as the style transfer network T in the reversible transformation furthermore enables an additional level of control over strokes through stroke size and intensity adjustment (λ_s and λ_I). For this, both the high-resolution and low-resolution input images are transformed by the warping function.

4.3 Local (non-parametric) warping

While global mappings can generate a range of interesting effects, they are limited to predefined function mappings and cannot be adopted by the user to transform local image content, e.g., based on landmarks. For this, we implement non-parametric warping using thin-plate spline interpolation as proposed by Bookstein et al. [5], which can be controlled given source and destination landmarks as shown in Fig. 8. As thin-plate splines are generally not invertible analytically, we use an iterative solver [12] based on Newton's method for inverting the transform. However, not every warp is reversible; if the information loss is too high (e.g., many pixels folded onto a single pixel), solving the inverse fails. In this case, the last landmark added by the user is reverted, and a warning is shown.

5 Interactive NST control on mobile devices

We integrate the previously introduced concepts for multi-level stroke control into a holistic pipeline for interactive style transfer. Figure 9 shows an overview of the components and data flow. To demonstrate the applicability of this approach in an end-user-focused context, we implement the *StyleTune* app for interactive style transfer editing (contribution C.4). It enables control over strokes configurations in NST on a global and local level, in particular over stroke size, orientation, intensity, and local deformations using the previously described approaches.

5.1 Overview of processing stages

The editing and processing pipeline in *StyleTune* (Fig. 9) comprises the following main stages:

Stroke-Feature Computation: Stroke features f_i are computed for different stroke parameters based on the content input image I_C using the encoder network E trained on the style image I_S. Before passing I_C through the network, a reversible transformation R can be applied (e.g., rotation by angle τ).

Real-time Preview Generation: To enable interactive local adjustments, a real-time preview I_P is generated by image-based blending of intermediate results I_i obtained from the decoder network D according to the spatial mask I_M. Any applied input transformations R are inverted (R^{-1}) after blending.

Local-adjusted NST Generation: To seamlessly blend between several different stroke-size edits, model-space blending of features f_i based on I_M is performed, The resulting stroke feature map $\mathcal{F}$ is then jointly decoded by D and again followed up by transformation inversions R^{-1}.

High-resolution Upsampling: Optionally, high-resolution outputs can be obtained through patch-based upsampling [41] of I_O. The upsampling step is executed on-device and progressively refines the image.

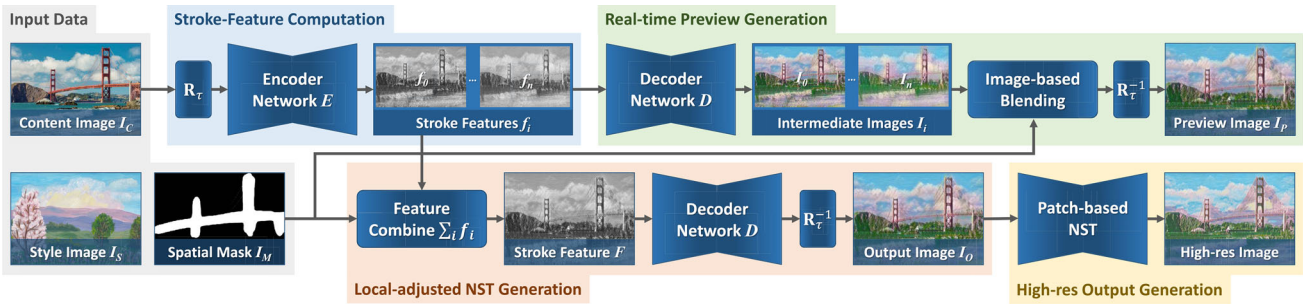


Fig. 9 Overview of the processing stages, components, and data flow for the interactive editing approach

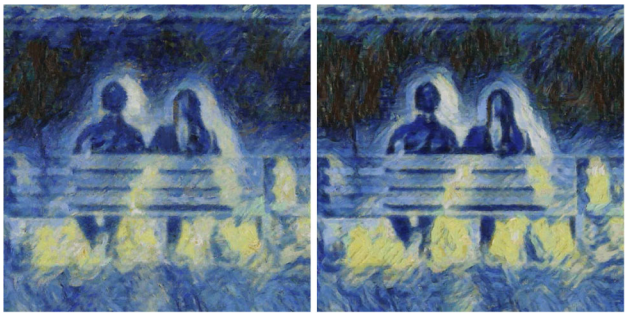
5.2 Local editing of image regions

While encoder and decoder execution is fast at approx. 100 ms for 1 Mpix images, the response time would not allow for a smooth and interactive experience. Therefore, intermediate results I_i and features f_i are pre-generated for several stroke sizes or rotations. The spatial mask I_M is created interactively by the user and continuously blends pre-computed results according to I_M in image space to generate a preview I_P.

While the presented adjustable network architecture works well for global adjustments and large-scale local adjustments, patterns on different stroke levels are generally not placed consistently, i.e., a swirly brushstroke from Van Gogh's "Starry Night" might be generated at differing locations when generating images at different stroke scales. This limitation originated from the feedforward networks' activation and stylization response depending on the scale of the input (Sect. 3.1). To overcome this limitation in *StyleTune*, we add the architecture of Jing et al. [17] as an option for detail control in scenarios where strokes are expected to flow seamlessly between different local adjustments. Style transfer models are converted to CoreML [29] and weights are quantized to 16 bit for optimized execution on mobile devices supporting iOS. The app implementation is based on Apple's Swift, CoreML, and Metal Application Programming Interfaces (APIs) for Graphics Processing Unit (GPU)-based processing. Neural network operations that are not part of the CoreML standard are implemented using Metal shaders, e.g., for local feature merging.

5.3 Patch-based upsampling

In a final step, high-resolution images can be created using the patch-based upsampling algorithm introduced by Texler et al. [41]. The output of the previous processing and editing steps is hereby used as a guidance image to reconstruct a high-resolution output using patches from the style image. Specifically, we implement a variant of this method to run on-device using mobile GPU-optimized operations (contribution C.5), which is described in the following.



(a) 1 iteration (b) 3 iterations

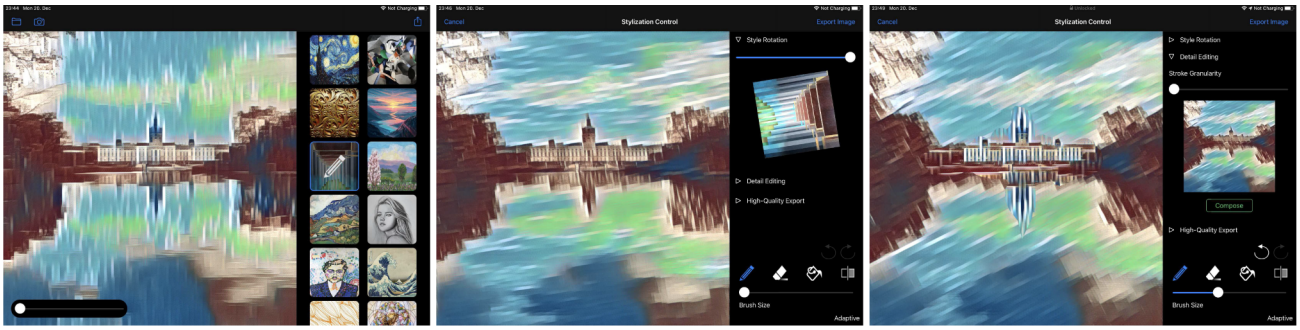
Fig. 10 PatchMatch [3] iterations in patch-based upsampling (zoomed in). Higher iterations enhance content details and transfer more fine granular strokes from the style image

Error metric The patch-based reconstruction is optimized according to a patch-similarity error metric E. It uses a pair of guidance channels, the source guide image G_S, initialized with a blurred version of the style image I_S, and the target guide image G_T, initialized with a subsampled version of the style transfer result I_O. Following Texler et al. [41], the error metric for matching two patches $p \in G_S$ and $q \in G_T$ is defined as

$$E(I_\mathrm{S}, I_\mathrm{R}, G_S, G_T, p, q) = \\ \|I_\mathrm{S}(p) - I_\mathrm{R}(q)\|^2 + \lambda_g \|G_S(p) - G_T(q)\|^2,$$

where I_R is the resulting image that is updated by the patch-based algorithm iteratively. Here, the first term promotes texture coherence by directly matching colors in the style image to the output image, while the second term matches patches in the guidance space, and λ_g is a parameter used to weight both terms.

Optimization Texler et al. use an Expectation Maximization (EM) approach based on StyLit [9] and efficient patch-match queries [3,4] to optimize the error metric. The optimization is run in a course-to-fine manner on different



(a) Adjusting stroke sizes globally (b) Adjusting orientation of strokes (c) Local retouches of stroke granularity

Fig. 11 Screenshots of *StyleTune*: After selecting a style, stroke sizes can be adjusted globally. The user can then adjust the stroke orientation, and retouch parts of the image with different stroke sizes using *brush metaphors*

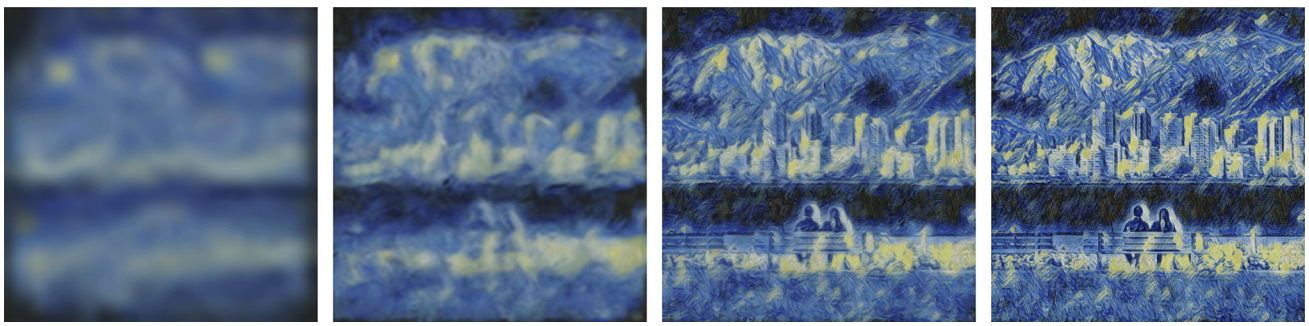


Fig. 12 Processing pyramid levels of patch-based upsampling, displayed as progress to the user

pyramid levels, with a subsampling factor of 2 between each. The result of the optimization is a Nearest Neighbor Field (NNF) that assigns a source patch to each target patch. In a final step, the NNF is upscaled to the original resolution of the style image (in many cases 4K and more) and, using a majority voting step [42], the output image is synthesized from high-resolution style image patches.

On-device implementation We implement patch-based upsampling in Swift and use Apple's Accelerate library for efficient CPU processing and Metal shaders for GPU-based processing. Following StyLit [9], nearest-neighbor retrieval is accelerated using Barnes et al. [3] PatchMatch algorithm. It is processed in three phases: initialization, patch propagation pass, and random search pass; each is executed as a separate Metal kernel. To efficiently generate pseudorandom numbers on the GPU, a Metal-based implementation [47] of Monte-Carlo random number generation [30] is used. We use a patch size of 5 pixels, set $\lambda_g = 2$, and, in contrast to StyLit [9], use only 3 patch-match iterations to reduce computation time. Using even fewer iterations can achieve acceptable results in some cases, as Fig. 10 shows. However, it also often leads to washed-out textures and low contrast. Intermediate images are continuously updated to visualize the progress for the user.

6 User interface of mobile application

Figure 11 illustrates the three-step process used to create the final stylization result using *StyleTune*. In the following, the interactive image editing and enhancement workflow is described. The user experience is designed to accommodate editing on multiple levels of control [16].

Selection of content and style images After loading or capturing an input image I_C, the user chooses a style image I_S among a variety of trained styles (Fig. 11a). The application then applies the selected NST model to the content image. The resulting style change is presented as a real-time preview image I_P at 1Mpix resolution, allowing users to navigate styles and select a style for further editing quickly.

Adjustment of stroke size and orientation Once a style transfer model has been selected, users can interactively adjust the stroke size λ_S on a global level by using a slider. Furthermore, interactive adjustments of global stroke rotation can be made in an additional editing view (Fig. 11b). After entering the local editing view, preview results are pre-generated for several stroke sizes or rotations, which incurs a brief loading time. Users can then locally apply different stroke sizes or orientations using a painting brush metaphor, which blends precomputed results in image space. Stroke edits can be merged in feature space on-demand to create seamless stroke transitions (Fig. 11c).

Fig. 13 Comparing global stroke size adjustment from lowest to highest level, predicted on an image with edge length of 2048 pixels. The AAMS network [46] predicts stroke sizes using self-attention, depicted are zoom-ins on smallest and largest generated strokes. Our approach can represent a higher range of stroke sizes than the adaptive network of Jing et al. [17] and AAMS

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

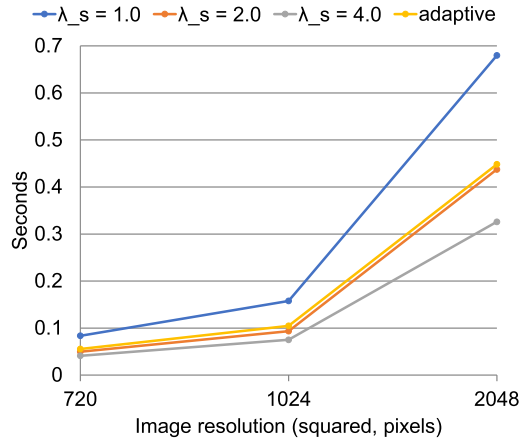
Fig. 14 Comparison between global stylization and locally adjusted versions produced using *StyleTune*

Creation of high-resolution images In a final step, the composition can be rendered at a very high resolution using patch-based upsampling [41]. Upsampling is executed on-device and continuously displays the progressive refinement of the output image, as shown in Fig. 12. Once the final image has been rendered, it can be explored using zoom and pan gestures e.g., to reveal fine-grained details such as canvas structure and paint bristles that have been transferred from the style image. Users can then export and share their creations.

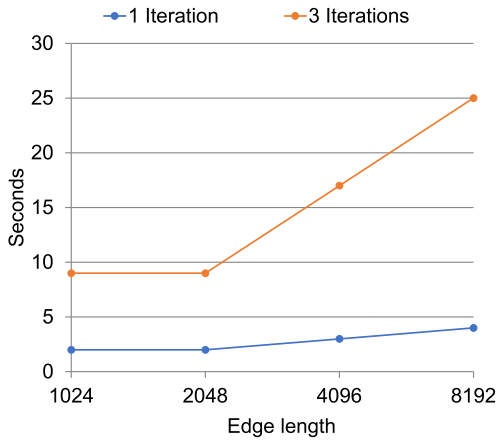
7 Results

7.1 Qualitative comparison

Figure 13 compares results of our method for stroke size adjustment to that of Jing et al. [17] and the Attentionaware Multistroke Style Transfer (AAMS) of Yao et al. [46], which, to our knowledge, are the only previous works for explicit stroke size control. For moderate stroke-sizes, our results are similar to other single style transfer networks (e.g., those of Johnson et al. [19] and Jing et al. [17]), which are still considered state-of-the-art quality-wise compared to arbitrary style transfer approaches [18]. As Fig. 13 shows, our architecture for multivariate control is able to represent a broader range of stroke sizes. AAMS swaps elements from the style image to the target image, which can lead to unwanted artifacts such as an eye from the style image appearing on the mountain (c.f. first column). The common, single branch NST architectures [17,19] also produce much more diminutive and subtle stroke size elements when applied to high-resolution images as they are typically trained on low-resolution images (usually 256×256 pixels) and retain this stroke scale. Diminishing stroke sizes at large resolutions are often not desirable, as



(a) Network inference time



(b) Patchbased upsampling runtime

Fig. 15 **a** Performance comparison of our adjustable architecture and the stroke-adaptive network of Jing et al. [17]. Performance of our architecture varies with the stroke scale λ_s. Tests were performed using a Nvidia GTX 1080Ti GPU and averaged over 100 runs; **b** Performance of our metal-based implementation of patch-based upsampling [41] for different input sizes on a MacBook Pro 2017. Runtime can be influenced by the iteration count of the PatchMatch [3] algorithm

users typically want to create large texture marks for visual effects on high-resolution images as well. Aside from the much higher computational cost to train on large resolution images, the stylization capacity of the network—mainly driven by its receptive field size—is often not sufficient to create large texture marks at common image resolutions of edge length 1024 and above. Our architecture mitigates this by downscaling inputs to the stylization branch according to the desired stroke size and adding back high-frequency details after stylization. In theory, arbitrarily large stroke sizes λ_s can be set; however, in practice, large stroke sizes ($\lambda_s \geq 8$) tend to lose sharpness as the style-branch resolution decreases. A further advantage of our approach is that the architecture can represent size-consistent strokes across different output image resolutions. Therefore, users can efficiently perform editing on lower-resolution previews and export them to a higher-resolution final image.

7.2 Exemplary editing results

In Fig. 14, *StyleTune* was used to locally retouch the face and hair of the person. By using smaller stroke granularities and reducing the style intensity, the face is less obstructed by stroke texture marks, and a visually more pleasing result is achieved. Additionally, the higher luminance directs the gaze and creates a focal point in the image. Orienting the strokes along flowlines of the hair in Fig. 14c, furthermore increases the image's depth due to the improved visual separation of foreground subject and background. The local guidance tools offered by *StyleTune* thus enable a form of art direction, where semiotic aspects of the style are individually placed and adjusted and represent a step toward *semiotics-based loss functions* [37].

7.3 Performance considerations

Our proposed network architecture has similar runtime performance characteristics to other feedforward neural style transfers, such as the architecture of Johnson et al. [19] or Jing et al. [17]. Furthermore, the runtime depends on the stroke-size settings, as larger λ_S inversely create smaller stylization branch input images. As Fig. 15 shows, inference with stroke sizes $\lambda_S \geq 2$ performs equally as fast or faster than the adaptive network of Jing et al. [17]. The runtime performance of patch-based upsampling scales linearly with edge-length (Fig. 15b), using fewer iterations of PatchMatch [3] offers a trade-off between runtime and visual quality.

Tests on mobile were executed using an iPad Pro 3rd generation equipped with an Apple A12X Bionic and 4GB Random Access Memory (RAM). On input image resolution of 1024×1024 pixels, global style transfer required approx. 0:5 s and pre-computation of ten stroke sizes using the style-encoder network required approx. 5 s. While the image-based blending of previews is computed in real-time, blending and decoding strokes in feature space required approx. 3 s. Empirically, ten levels of strokes sizes provide fine-granular and visually smooth level transitions when exploring styles using the global stroke-scale slider. For local editing, on the other hand, as low as three levels are sufficient for most editing purposes, as stroke scales are often used to create visual contrasts between stylized areas. For patch-based upsampling, mobile execution is approx. 30% slower compared to desktop processing on MacOS (Fig. 15b).

7.4 Limitations

While our introduced network architecture and editing modes enable more degrees of artistic freedom for feedforward neural style transfer, there are still some limitations to overcome. For the adjustable network architecture, the previously described inconsistency in style element placement presents a limitation for use-cases that involve local, fine detail editing. Further, the possibility of combining rotations and stroke scale edits in *StyleTune* makes it necessary to recompute all intermediate stroke-scale results if the stroke rotation is changed, which can limit exploratory workflows as they require instant feedback for a sophisticated user experience. While the presented non-parametric warp-based reversible editing can achieve a large variety of effects without requiring any changes to the network, using it as an editing tool in practice remains challenging due to the counter-intuitive effects of reversible warping on style elements. Finally, patch-based upsampling of style transfer results noticeably alters their global appearance in ways that the user may not intend.

8 Conclusions and future work

This work introduces a method for multivariate stroke control over feedforward neural style transfer. We propose a new style transfer architecture for simultaneous stroke size and style intensity control. Furthermore, we introduce the concept of reversible input transformations that can adjust elements of the style, such as the stroke orientation, regardless of style transfer architecture. To demonstrate the real-world applicability of our approach as a tool for expressive style transfer editing, we implement a mobile app that enables interactively adjusting and retouching the aforementioned stylistic aspects. The use cases and results show that our method provides more expressive control than comparable state-of-the-art methods and new modes of expression that were previously not possible. Furthermore, we demonstrate the potential of patch-based upsampling as part of a style transfer editing pipeline to generate high-resolution output renderings and demonstrate that a mobile on-device implementation can provide interactive feedback to the user. To this end, we believe this work provides a further step toward making style transfer a more expressive tool for art-directed image stylization in casual and professional applications.

As future work, we plan to further explore making style elements locally persistent across multiple scales and deepen the integration of stroke editing patch-based upsampling by adding a neural representation of stroke size and orientation. Furthermore, the proposed approach may be generalizable to 3D data [6,27].

Funding Open Access funding enabled and organized by Projekt DEAL. This work was partially funded by the German Federal Ministry of Education and Research (BMBF) through grants 01IS18092 (“mdViPro”) and 01IS19006 (“KI-LAB-ITSE”).

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Amato, G., Behrmann, M., Bimbot, F., Caramiaux, B., Falchi, F., Garcia, A., Geurts, J., Gibert, J., Gravier, G., Holken, H., et al.: AI in the media and creative industries. arXiv preprint arXiv:1905.04175 (2019)
2. Babaeizadeh, M., Ghiasi, G.: Adjustable real-time style transfer. In: 8th International Conference on Learning Representations, ICLR 2020 (2020)
3. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B.: Patchmatch: a randomized correspondence algorithm for structural image editing. ACM Trans. Graph. **28**(3), 24 (2009)
4. Barnes, C., Zhang, F.L., Lou, L., Wu, X., Hu, S.M.: Patchtable: efficient patch queries for large datasets and applications. ACM Trans. Graph. **34**(4), 1–10 (2015)
5. Bookstein, F.L.: Principal warps: thin-plate splines and the decomposition of deformations. IEEE Trans. Pattern Anal. Mach. Intell. **11**(6), 567–585 (1989)
6. Chen, D., Yuan, L., Liao, J., Yu, N., Hua, G.: Stereoscopic neural style transfer. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6654–6663 (2018)
7. Dapkus, D.: How to transfer styles to images with Adobe Photoshop. https://creativecloud.adobe.com/de/discover/article/how-to-transfer-styles-to-images-with-adobe-photoshop
8. Dumoulin, V., Shlens, J., Kudlur, M.: A Learned representation for artistic style. In: ICLR (2017)
9. Fišer, J., Jamriška, O., Lukáč, M., Shechtman, E., Asente, P., Lu, J., Sỳkora, D.: Stylit: illumination-guided example-based stylization of 3d renderings. ACM Trans. Graph. **35**(4), 1–11 (2016)
10. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2414–2423. IEEE Computer Society (2016)
11. Gatys, L.A., Ecker, A.S., Bethge, M., Hertzmann, A., Shechtman, E.: 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, pp. 3730–3738. IEEE Computer Society (2017)
12. Gobbi, D.G., Peters, T.M.: Generalized 3d nonlinear transformations for medical imaging: an object-oriented implementation in VTK. Comput. Med. Imaging Graph. **27**(4), 255–265 (2003)
13. Gu, S., Chen, C., Liao, J., Yuan, L.: Arbitrary style transfer with deep feature reshuffle. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8222–8231 (2018)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2016)
15. Huang, X., Belongie, S.: Arbitrary style transfer in real-time with adaptive instance normalization. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 1510–1519. IEEE Computer Society (2017)
16. Isenberg, T.: Interactive NPAR: what type of tools should we create? In: Proceedings of the NPAR, Expressive '16, pp. 89–96. Eurographics Association, Goslar, DEU (2016)
17. Jing, Y., Liu, Y., Yang, Y., Feng, Z., Yu, Y., Tao, D., Song, M.: Stroke controllable fast style transfer with adaptive receptive fields. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 244–260 (2018)
18. Jing, Y., Yang, Y., Feng, Z., Ye, J., Yu, Y., Song, M.: Neural style transfer: a review. IEEE Trans. Vis. Comput. Graph. **26**(11), 3365–3385 (2020)
19. Johnson, J., Alahi, A., Fei-Fei, L.: 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, Lecture Notes in Computer Science, vol. 9906, pp. 694–711. Springer (2016)
20. Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., Aila, T.: Analyzing and improving the image quality of StyleGAN. In: Proceedings of the CVPR (2020)
21. Kingma, D.P., Ba, J.: 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)
22. Klingbeil, M., Pasewaldt, S., Semmo, A., Döllner, J.: Challenges in user experience design of image filtering apps. In: Proceedings SIGGRAPH ASIA Mobile Graphics and Interactive Applications. ACM, New York (2017)
23. Kopf, J., Cohen, M.F., Lischinski, D., Uyttendaele, M.: Joint bilateral upsampling. ACM Trans. Graph. **26**(3), 96–102 (2007)
24. Kyprianidis, J.E., Collomosse, J., Wang, T., Isenberg, T.: State of the "art": a taxonomy of artistic stylization techniques for images and video. IEEE Trans. Vis. Comput. Graph. **19**(5), 866–885 (2012)
25. Li, Y., Fang, C., Yang, J., Wang, Z., Lu, X., Yang, M.H.: Universal style transfer via feature transforms. In: Advances in Neural Information Processing Systems (2017)
26. Li, Y., Huang, J.B., Ahuja, N., Yang, M.H.: Deep joint image filtering. In: European Conference on Computer Vision, pp. 154–169. Springer (2016)
27. Liang, Y., He, F., Zeng, X.: 3d mesh simplification with feature preservation based on whale optimization algorithm and differential evolution. Integr. Comput.-Aided Eng. **27**(4), 417–435 (2020)
28. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. In: Proceedings of the ECCV, pp. 740–755. Springer, Cham (2014)
29. Marques, O.: Machine Learning with Core ML, pp. 29–40. Springer, Cham (2020)
30. Mohanty, S., Mohanty, A.K., Carminati, F.: Efficient pseudo-random number generation for Monte-Carlo simulations using graphic processors. J. Phys.: Conf. Ser. **368**, 012024 (2012)
31. Moiseenkov, A., Poyaganov, O., Frolov, I., Usoltsev, A.: Prisma. Version: 4.3.4. https://prisma-ai.com/ (2021)
32. Pasewaldt, S., Semmo, A., Döllner, J., Schlegel, F.: BeCasso: artistic image processing and editing on mobile devices. In: SIGGRAPH ASIA 2016, Macao, December 5–8, 2016—Mobile Graphics and Interactive Applications, p. 14:1. ACM (2016)
33. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019)
34. Reimann, M., Buchheim, B., Semmo, A., Döllner, J., Trapp, M.: Interactive multi-level stroke control for neural style transfer. In: 2021 International Conference on Cyberworlds (CW), pp. 1–8 (2021)
35. Reimann, M., Klingbeil, M., Pasewaldt, S., Semmo, A., Trapp, M., Döllner, J.: MaeSTrO: a mobile app for style transfer orchestration using neural networks. In: 2018 International Conference on Cyberworlds, CW 2018, Singapore, October 3–5, 2018, pp. 9–16. IEEE Computer Society (2018)
36. Reimann, M., Klingbeil, M., Pasewaldt, S., Semmo, A., Trapp, M., Döllner, J.: Locally controllable neural style transfer on mobile devices. Vis. Comput. **35**(11), 1531–1547 (2019). https://doi.org/10.1007/s00371-019-01654-1
37. Semmo, A., Isenberg, T., Döllner, J.: Neural style transfer: a paradigm shift for image-based artistic rendering? In: Proceedings International Symposium on Non-Photorealistic Animation and Rendering (NPAR), pp. 5:1–5:13. ACM, New York (2017)
38. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: 3rd International Conference on Learning Representations, ICLR 2015. San Diego, CA, USA (2015)
39. Su, H., Jampani, V., Sun, D., Gallo, O., Learned-Miller, E., Kautz, J.: Pixel-adaptive convolutional neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11166–11175 (2019)
40. Tewari, A., Fried, O., Thies, J., Sitzmann, V., Lombardi, S., Sunkavalli, K., Martin-Brualla, R., Simon, T., Saragih, J., Nießner, M., et al.: State of the art on neural rendering. In: Computer Graphics Forum, vol. 39, pp. 701–727. Wiley Online Library (2020)
41. Texler, O., Fišer, J., Lukáč, M., Lu, J., Shechtman, E., Sýkora, D.: Enhancing neural style transfer using patch-based synthesis. In: Proceedings of the NPAR, Expressive '19, pp. 43–50. Eurographics Association, Goslar, DEU (2019)
42. Wexler, Y., Shechtman, E., Irani, M.: Space-time completion of video. IEEE Trans. Pattern Anal. Mach. Intell. **29**(3), 463–476 (2007)
43. Wu, H., Sun, Z., Zhang, Y., Li, Q.: Direction-aware neural style transfer with texture enhancement. Neurocomputing **370**, 39–55 (2019)
44. Wu, H., Zheng, S., Zhang, J., Huang, K.: Fast end-to-end trainable guided filter. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1838–1847 (2018)
45. Yang, L., Yang, L., Zhao, M., Zheng, Y.: Controlling stroke size in fast style transfer with recurrent convolutional neural network. In: Computer Graphics Forum, vol. 37, pp. 97–107. Wiley Online Library (2018)
46. Yao, Y., Ren, J., Xie, X., Liu, W., Liu, Y., Wang, J.: Attention-aware multi-stroke style transfer. In: IEEE Conference on Computer Vision and Pattern Recognition. CVPR, pp. 1467–1475. Computer Vision Foundation/IEEE, Long Beach, CA, USA (2019)
47. Youssef, V.: Loki: a random number generator for Metal (2017). https://github.com/YoussefV/Loki
48. Zhang, H., Dana, K.: Multi-style generative network for real-time transfer. In: Computer Vision—ECCV 2018 Workshops, pp. 349–365. Springer (2019)

49. Zhu, S.C., Guo, C.E., Wang, Y., Xu, Z.: What are textons? Int. J. Comput. Vis. **62**(1), 121–143 (2005)

Max Reimann is a Ph.D. candidate at the visual computing group of the Hasso Plattner Institute at the University of Potsdam, Germany, where he received his master's degree in 2018. His research interests include image and video processing, and computer vision for scene understanding. He is particularly interested in deep-learning-based image and video stylization techniques, and how to evolve these into interactive tools for casual creativity.

Benito Buchheim is currently pursuing his master's degree in IT-Systemsstanding interest in computer graphics and visual computing, and in his current role as a student research assistant at the institute's visual computing group, his research focuses on deep-learning-based stylization techniques such as neural style transfer.

Amir Semmo received his doctoral degree in 2016 at the Hasso Plattner Institute in Potsdam, Germany, on the topic of non-photorealistic rendering for 3D geospatial data. Since 2019 he is the Head of R&D at Digital Masterpieces GmbH in Potsdam. His principle research topics are related to image and video processing, computer vision and GPU computing. He is particularly interested in expressive and artistic rendering under the umbrella of interactive casual creativity, and stylization of multi-dimensional image and video data.

Jürgen Döllner obtained his doctorate in computer science at the University of Münster (1996) on modeling and rendering in computer graphics and habilitated after stays abroad. Since 2001 he is professor for analysis, planning and construction of complex systems at the Hasso Plattner Institute of the University of Potsdam. His work focuses on visual computing, especially in the areas of geospatial analytics, software analytics and video analytics. His visual analytics group has so far given rise to a number of software technology start-ups.

Matthias Trapp studied computer science at the University of Potsdam and the Hasso Plattner Institute, Germany (2000–2007) where he received his Ph.D. in Computer Science (2013). During his post-doctoral studies, he was heading the junior research group on "4D-nD Geovisualization" (2012–2017). Since 2017, he is a senior researcher at the Hasso Plattner Institute. His major research areas are computer graphics, image and video processing, geovisualization, software visualization, and information visualization with a focus on GPU-based techniques.