Controlling Geometric Abstraction and Texture for Artistic Images

Martin Büßemeyer*, Max Reimann*, Benito Buchheim

Hasso Plattner Institute for Digital Engineering University of Potsdam, Germany * authors contributed equally Amir Semmo Digital Masterpieces GmbH Potsdam, Germany

Jürgen Döllner, Matthias Trapp Hasso Plattner Institute for Digital Engineering University of Potsdam, Germany

Abstract-We present a novel method for the interactive control of geometric abstraction and texture in artistic images. Previous example-based stylization methods often entangle shape, texture, and color, while generative methods for image synthesis generally either make assumptions about the input image, such as only allowing faces or do not offer precise editing controls. By contrast, our holistic approach spatially decomposes the input into shapes and a parametric representation of high-frequency details comprising the image's texture, thus enabling independent control of color and texture. Each parameter in this representation controls painterly attributes of a pipeline of differentiable stylization filters. The proposed decoupling of shape and texture enables various options for stylistic editing, including interactive global and local adjustments of shape, stroke, and painterly attributes such as surface relief and contours. Additionally, we demonstrate optimization-based texture style-transfer in the parametric space using reference images and text prompts, as well as the training of single- and arbitrary style parameter prediction networks for real-time texture decomposition.

Index Terms—texture decomposition, neural style transfer, geometric abstraction, texture control, stroke-based rendering

I. INTRODUCTION

While recent methods for image synthesis, such as diffusion models [1], and example-based stylization methods, such as Neural Style Transfer (NST) [2], [3], achieve impressive results, their black-box character and lack of disentangled artistic control variables [4] makes precise adjustment of geometric elements and texture challenging. Thus, given an artistic image obtained from such methods or given an artistic image without having control over its formation process, achieving a desired artistic look may necessitate further adjustments to geometric and textural artistic control variables. These variables include geometric elements such as brush shapes and sizes, and textural elements such as stroke patterns, tonal variations, and other intricate features in the image. In this paper, we propose a novel method for geometric abstraction and texture control that allows example-based as well as parametric control over such artistic variables. Our method¹ accepts an artistic image as input, without any restrictions on the content domain, and outputs an image with applied adjustments in the geometric and textural space which preserve the input color distribution.

To allow for disentangled editing of artistic control variables, we first decompose the input image into primitive shapes representing coarse structure and a parametric representation of high-frequency details that form the image's texture. The coarse structure decomposition is achieved using segmentation techniques, e.g., superpixel segmentation [5], or layered approaches such as neural stroke-based rendering [6], [7], depending on the desired shape primitives. To decompose the high-frequency details into meaningful artistic control variables, a novel lightweight pipeline of differentiable stylization filters is introduced. Filters in this pipeline are based on traditional Image-based Artistic Rendering (IB-AR) filters [8] implemented in an auto-grad enabled framework [9], and are parameterized by explicitly defined stylistic or painterly attributes, such as contours, surface relief (e.g., for oil-paint texture), and local contrast, among others. Detail decomposition is then achieved by optimizing filter parameters such that the output of the filter pipeline, conditioned on the coarse structure image, matches the input image.

The resulting spatial and value decomposition provides a wide range of options for editing textures and geometric abstraction (Fig. 1). The method of geometric abstraction is interchangeable and, depending on the method of choice, provides interactive control over the stroke shapes and the level-of-abstraction. Furthermore, filter parameters can be adjusted interactively on both global and local levels using per-pixel parameter masks, either by manual editing or interpolating values based on extracted image attributes such as depth or saliency. Overall, the decomposed representation allows for independent control of color and texture, enabling color adjustments without affecting the texture and vice versa, which is particularly useful for editing tasks such as correcting artifacts in images created by NST.

Our approach performs particularly well for example-based edits of fine-granular texture patterns by adapting to the underlying coarse shape primitives and achieving a spatially homogeneous appearance. At this, example-based losses can be used to optimize the texture in the parametric space to conform to a desired style image using NST style-losses [2], [10] or text-based losses [11]. For particular stylization tasks such as style transfer, the texture decomposition optimization can be further accelerated by training Parameter Prediction Networks (PPNs) for real-time decomposition.

To summarize, we make the following contributions:

1) We present a holistic approach for geometric abstraction

¹Project and Code: https://maxreimann.github.io/artistic-texture-editing/



Fig. 1: Our method enables texture and geometric abstraction editing while keeping the overall color and structure intact. In A-C, textures are adjusted by increasing the oiliness and stroke thickness (A) and using the text prompts "starry night" (B) and "impressionistic painting" (C). In D - F the geometric abstraction primitives are varied between rectangular strokes (D), blocks (E), and ellipsoids (F). These edits can be combined and interactively adjusted.

and texture editing that decomposes the image into coarse shapes and high-frequency detail texture.

- 2) We present a novel differentiable filter pipeline for texture editing. Compared to prior work it is lightweight and improves parameter editability.
- 3) We introduce PPNs for real-time single and arbitrary style texture decomposition.
- 4) We demonstrate that a texture can be adapted to new styles using example images or text prompts, and can be interactively adjusted on a global or local level.

II. RELATED WORK

Deep network-based methods, particularly NST [2], learn stylistic representations in a black-box fashion, transferring texture style from a reference style image to a content image. In addition to the style-content trade-off control inherent to NST [2], several methods have been developed to enable control over aspects such as color [12] or strokes [13], [14]. CLIPstyler [11] can perform style transfer from text prompts using CLIP-based losses [15]. However, these methods either cannot be interactively controlled (i.e., optimization-based approaches) or are not easily composable with each other (i.e., network-based approaches).

Traditional IB-AR [8], on the other hand, represent styles as a chain of image filters that allow fine-granular control over multiple style aspects, but have to be specifically designed for a particular style, such as for watercolor [16], oil paint [17], or cartoon [18]. Recently, Lötzsch et al. [9] proposed an interactively controllable "whitebox" style representation by optimizing the parameters of such IB-AR filter pipelines to match a stylized reference image. Similar to this work, we also use a filter-based interpretable style representation. However, Lötzsch et al. [9] represent the entire input image in such parameters, i.e., shape, color, and details (e.g., local textures) are entangled in a large set of often redundant parameters and filters. This makes editing tedious and unintuitive, as adjusting parameters controlling painterly attributes may have unwanted side-effects on colors or shapes. Instead of matching the entire input image, we represent color and shape distribution in a geometric abstraction stage and use our subsequent filter pipeline to represent the high frequency components, i.e., the local textures. In contrast to the intricate, style-specific filter pipelines [16], [17] employed by [9], we remove redundancy and improve parameter editability by proposing a light-weight filter pipeline capable of matching arbitrary styles while consisting of only four filters.

While Lötzsch et al. [9] demonstrate the viability of PPNs for a single img2img translation task in combination with an additional postprocessing CNN, we show that PPNs can be adapted for general single- and arbitrary style transfer decomposition tasks, and do not require postprocessing networks.

Our geometric abstraction stage is inspired by approaches that use either segmentation-based representation of shapes [19], [20] or stroke-based rendering [21], [22] to abstract the image into primitive shapes. Specifically, we make use of recent neural painting approaches using optimization-based [7] or prediction-based [6] methods to represent the image as a set of layered primitives, or alternatively use SLIC [5]-based segmentation for non-layered representation.

III. Method

A. Framework Overview

To obtain a decomposed image representation for downstream editing tasks, texture decomposition is initially executed. Fig. 2



Fig. 2: Texture decomposition. The parameter masks P_M controlling individual painterly attributes in a filter pipeline O are optimized through loss \mathcal{L} to match output image I_o and input image I_t . The pipeline expects an abstracted image I_a as input, which can be obtained from different methods for the geometric abstraction of I_t in the segmentation stage S.

shows the two involved stages: (1) a segmentation stage $S(\cdot)$ to control texture granularity and geometric abstraction, and (2) a pipeline of differentiable image filters $O(\cdot)$ to represent image details. The first stage renders an abstracted variant I_a of the input image I_t using shape primitives. The second stage represents the details, i.e., the difference between I_a and I_t , in the parameters of a filter pipeline O. Each of these filter parameters represents a specific artistic control variable, e.g., contours, and local contrast, among others (Sec. III-D), and is controllable on a per-pixel level using parameter masks P_M . Decomposition is achieved by first executing stage S and then optimizing the decomposition loss over pipeline output I_o to adapt P_M , which we detail in the following.

B. Decomposition Loss

Formally, O is parametrized by a set of M parameter masks, i.e., $P_M = \{P_i \in \mathbb{R}^{h \times w} | i \leq M\}$ and expects the output image of the segmentation stage $I_a = S(I_t)$ as input. A stylized output image I_o is thus obtained as:

$$I_o = O\left(P_M, S(I_t)\right) \tag{1}$$

To obtain the decomposed texture representation, the parameters P_M are optimized using a loss function \mathcal{L} that combines a target loss, denoted by \mathcal{L}_{target} , and the total variation loss, denoted by \mathcal{L}_{TV} weighted by λ_{TV} :

$$\mathcal{L} = \mathcal{L}_{target} + \lambda_{TV} \mathcal{L}_{TV} \tag{2}$$

The target loss ensures closeness to a desired target style, while \mathcal{L}_{TV} reduces noise in masks and enforces local consistency.

Different types of loss functions can be used for the target loss as follows. When the objective is to allow for subsequent interactive editing, the ℓ_1 loss is a suitable choice as it optimizes for reconstructing the details of the input image:

$$\mathcal{L}_{target} = \|O(P_M, S(I_t)) - I_t\|_1 \tag{3}$$

When the objective is to control and change the details of a style, text or image-based style transfer losses for \mathcal{L}_{target} can be used, as shown in Sec. IV-B. Similarly, such losses can also be used to train a PPN to reconstruct the details of a style-transferred input image in a single inference step. To demonstrate this, we introduce single-style and arbitrary-style decomposition networks in Sec. III-F.

C. Segmentation Stage

The segmentation stage S divides the input image I_t into distinct shape primitives, e.g., brushstrokes or segments, of uniform color. As S operates on pixels in both the input and output domains, a wide range of methods such as superpixel segmentation or stroke-based rendering can be utilized in this stage. We discuss several choices for the segmentation stage in Sec. IV-A. While changing abstraction settings in the segmentation stage is typically interactive and enables effect previews, the subsequent filter stage must be subsequently re-optimized to adapt to the newly introduced geometric structure.

D. Differentiable Filter Stage

Existing heuristics-based stylization pipelines (e.g., [16]–[18]) are not well-suited for example-based stylization editing tasks, as they were not designed with these specific use cases in mind. They often have non-trainable parameters, such as color quantization [18] which require differentiable proxies [9], and furthermore have repetitive components in the pipeline, such as repeated smoothing, which can hinder the optimization and ease of editing local parameter masks. To this end, we ensure that all parameters in our proposed pipeline are differentiable and that the pipeline is kept simple with intuitive parameters that produce the expected effects, while at the same time having the capacity to match arbitrary textures. To preserve the input color distribution (i.e., the segmentation stage output), filters should not be able to freely alter pixels on the color spectrum during optimization. Consequently, our pipeline's learnable parameters cannot modify color hue.

We propose a lightweight filter pipeline $O(\cdot)$ consisting of following differentiable filters:

- (1) Smoothing: We use a Gaussian smoothing ($\sigma = 1$) followed by a bilateral filter [23], with learnable parameters σ_d (distance kernel size) and σ_r (range kernel size).
- (2) Edge enhancement: We implement eXtended differenceof-Gaussians (XDoG) [24] with learnable parameters contour amount and contour opacity.
- (3) Painterly attributes: We use bump mapping for surface relief control in our evaluation and implement Phong shading [25] with learnable parameters bumpscale, Phong-specularity, and bump-opacity. However, any differentiable painterly filter can be used, e.g., we also implement wet-in-wet filters [26] and wobbling [16] for watercolorization control.
- (4) **Contrast:** This learnable parameter controls the amount of local contrast enhancement applied.

Gradients are computed with respect to both, the parameter masks P_M and the image input. To this end, such image



(a) Inputs I_t, I_a (b) Bilateral σ_d (c) Phong specular (d) Contour α

Fig. 3: Parameter masks P_M optimized without (top row) and with (bottom row) \mathcal{L}_{TV} to fit the input I_t (top row) to the given segmented image I_a .

filters are implemented in an auto-grad-enabled framework following [9]. The ablation study in Sec. V-B shows that all stages are necessary for arbitrary style representation. The proposed pipeline can be easily configured and optimized with further IB-AR techniques [8] for extended control of painterly attributes.

E. Optimization of Parameter Masks

The parameter masks P_M are optimized using gradient descent minimization of \mathcal{L} . Optimizing only \mathcal{L}_{target} as in [9] results in strongly fragmented masks exhibiting large local variations of values (Fig. 3, first row), which makes them hard to edit. When optimizing with \mathcal{L}_{TV} , masks have reduced noise, increased sparsity, and greater smoothness compared to those optimized without constraints.

Detail optimization broadly follows [9]; local parameter masks are optimized with 100 iterations of Adam [27] and a learning rate of 0.01. We decrease the learning rate by a factor of 0.98 every 5 iterations starting from iteration 50. In contrast to [9], smoothing of parameter masks is not required, as the employed filters do not exhibit artifacts. Throughout this paper, we use $\lambda_{TV} = 0.2$ during optimization.

F. Style Transfer Parameter Prediction

While general-purpose, the optimization-based decomposition is compute-intensive, taking approx. 3 min to optimize a 1MPix image on a Nvidia GTX 3090. For known tasks, such as NST, the decomposition step can be sped-up to real-time inference by training a PPN [9] to directly predict P_M , similar to (pixel predicting) NST networks [28], [29]. We propose single and arbitrary style transfer texture decomposition PPNs and train them as follows.

a) Single Style Decomposition PPN: We train a PPN— PPN_{sst}—to decompose the texture of a single style image I_s . We use the NST network of Johnson et al. [28], trained on I_s , to generate a stylized ground-truth image I_t and segment it using S as a preprocessing step to generate training inputs I_a for our filter pipeline O. The training loss for PPN_{sst} is then:

TABLE I: Properties of the superpixel and neural painting methods such as PaintTransformer (PTf) and neural painter (NPtr), for application in the segmentation stage. PTf uses a fixed shape primitive during training, whereas NPtr uses a trainable generator network to synthesize brush shapes.

Method	Туре	Primitive	Control	Runtime
SLIC [5]	Superpixel	segment	#segment	< 0.1 s
PTf [6]	Neural painter	fixed	#layer	< 1 s
NPtr [7]	Neural painter	trainable	#stroke	$< 60 \rm{s}$

$$I_o = O(\text{PPN}_{sst}(I_a), I_a) \tag{4}$$

$$\mathcal{L}_{\text{PPN}_{sst}} = \mathcal{L}_{gram}(I_o, I_s) + \lambda \mathcal{L}_c(I_o, I_t)$$
(5)

where \mathcal{L}_{gram} is the Gram matrix style loss, and \mathcal{L}_c the content loss over VGG [30] features following Gatys et al. [2]. The architecture for PPN_{sst} is adapted from [28] by setting the number of output channels in the last layer to the number of parameter masks ($\#P_M$).

b) Arbitrary Style Decomposition PPN: We train a PPN—PPN_{arb}—on a large set of style images I_s to decompose the textures of arbitrary styles, following the training methodology of previous work on arbitrary style transfer [3], [29]. The architecture for PPN_{arb} is adapted from SANet [29] by setting the number of output channels in the last layer of the decoder to $\#P_M$. For training, we adapt the preprocessing steps from before by using SANet to generate I_t . The training loss for PPN_{arb} is then:

$$I_o = O(\text{PPN}_{arb}(I_a, I_s), I_a) \tag{6}$$

$$\mathcal{L}_{\text{PPN}_{arb}} = \lambda_s \mathcal{L}_{\text{AdaIN}}(I_o, I_s) + \lambda_c \mathcal{L}_c(I_o, I_t)$$
(7)

Following SANet [29], we use the AdaIN [3] style loss. In contrast to SANet, we do not use a structure-retaining identity loss, as our filter pipeline O is itself much more constrained in its ability to change content structures than SANet.

c) Training Details: We trained both PPN_{sst} and PPN_{arb} using the MS-COCO dataset [31] for content images and used the WikiArt dataset as style images for training PPN_{arb}. As PPNs learn to predict suitable parameters over a large set of examples, the predicted masks contain little noise compared to optimized masks, thus \mathcal{L}_{TV} is not required for PPN training. Please see the supplementary for details on training hyperparameters.

IV. CONTROLLING ASPECTS OF STYLE

There are various methods for controlling the decomposed style representation after optimizing or predicting style parameters. We discuss methods for geometric abstraction control, and present techniques such as manual editing, reoptimizing using different style transfer losses, and interpolating predicted parameters.

A. Geometric Abstraction Control

The geometric abstraction of the image can be controlled through the choice of abstraction method (Tab. I), primitive shape, number of strokes, segments, or layers.



(a) SLIC [5] (b) PaintTransf. [6] (c) Neural Painter [7]

Fig. 4: Impact of various segmentation methods on the level of geometric abstraction in segmentation image I_a .

Generally, superpixel methods such as SLIC [5] divide the image into segments that are clustered according to color similarity and proximity, which align well with image regions and make them well-suited for use as an intermediate representation for interactive editing. These methods are primarily employed in our pipeline to represent uniform color regions of the image while abstracting away small-scale details (Fig. 4a), which facilitates downstream editing of both color and fine-structure parameters separately (Sec. IV-C).

Stroke-based rendering methods on the other hand are better suited for stronger geometric abstraction. We integrate two neural painting methods (Tab. I), which are trained to redraw images using a particular shape primitive, such as a brush, rectangle, or circle. While the neural painter [7] is able to create more abstract images, such as 8bit art (Fig. 4c) and optimize flexible shapes, the PaintTransformer [6] uses fixed primitive-shapes, however, is fast enough to allow for immediate visual feedback and adjusting. To enable spatially varying levels of stroke details, we multiply the stroke decision confidence of the PaintTransformer with an optional level of detail parameter mask P_S , please refer to [6] for details on the method. The mask P_S can be acquired either manually or predicted, e.g., using saliency or depth, as done in (Fig. 4b) to threshold the foreground and background levels.

Re-optimization of filter parameters adjusts the fine details to the shape and granularity of geometric elements and results in an integrated look, see Fig. 6, Fig. 10 and Fig. 11, as well as the supplemental material for examples of geometric shape and granularity variations.

B. Texture Style Transfer

Parameters P_M can be optimized using different losses to adapt the style details to new targets for varied artistic expression. As such, given a segmented image, we replace ℓ_1 matching of I_t with directly optimizing a style transfer loss in Eqn. (2). The optimization may either be initialized with a previous parameter decomposition to fine-tune the texture style or start from empty parameter masks.

a) Image-based Parameter Style Transfer: Fig. 5 exemplary shows results of optimizing the self-transport style loss (STROTTS [10]) to adapt the parameter-decomposition of a stylized image to different texture styles. Notice how the overall structure and colors are maintained while the fine details and texture are adapted.



(a) Great Wave (b) ink (c) vertical waves

Fig. 5: Parametric texture style transfer using example images. The decomposed parameters of an image, created by NST [2] using the "Starry Night" style, are retargeted to a new texture style using the shown style example images and are optimized with the STROTSS [10] loss.



(a) S output (b) S = 5000 segments (c) S = PTf-brush

Fig. 6: Re-optimization using CLIPstyler [11] loss with different first stages S. The input painting "rain princess" (teaser) is optimized by the prompt "a cubistic painting". In (b) we use a fine segmentation (5000 segments) and in (c) the PaintTransformer (PTf) [6] with brush-shapes as the first stage.

b) Text-based Parameter Style Transfer: Text-based representation of style allows for more freedom of control as reference images are not required. We adapt the losses from CLIPstyler [11] for text-based style transfer, however, in contrast to their method we do not train a Convolutional Neural Network (CNN), but directly optimize the effect parameters. In particular, we adapt their patch-based and directional loss in most cases, the use of content losses is not necessary as the filter pipeline inherently retains the major content structures and colors. Fig. 6 shows adapted style details of a real-world painting using prompts, with additional prompts results shown



Fig. 7: Using saliency (left) and depth (right) to interpolate parameters of the original NST with the prompts "fire" (left) and "drops" (right).



Fig. 8: Global parameter editing. Values of the bump-map scale P_M are increased (b) and decreased (c) uniformly.

(b) +bumpiness

(c) -bumpiness



Fig. 9: Correcting artifacts in NST images using a styletransfer PPN. Red dots in the face are removed using histogram matching of segments to other colors in the face. The PPN_{sst} re-prediction (candy style) reintegrates the manual edits seamlessly into the overall style.

in the supplemental material. It can be observed that colors remain faithful to the input while details adapt to the prompt and are adjusted to the size of segments.

C. Parameter Editing

(a) w/o detail edits

One major advantage of our filter parameter-based approach is that various stylistic aspects can be edited interactively, using both global and local parameter tuning.

- **Global Parameter Editing:** Parameters masks of the pipeline presented in Sec. III-D can be easily modified after optimization or prediction by uniformly adding or subtracting values on the parameter masks. Fig. 8 shows the result of globally increasing and decreasing the bump mapping parameter by 50%.
- **Local Parameter Editing:** Parameters can be edited locally through adjustment of the parameter masks using brush metaphors. For example, contours can be locally increased in the eye and mouth region of a portrait, to make them stand out (see supplementary video).
- **Parameter Interpolation:** Parameters can be interpolated locally using binary or continuous-valued blending masks. For automated stylization, we test saliency prediction (using LPF [32]) and depth prediction (using DPT [33]) to interpolate between two parameter sets (Fig. 7).

D. Correction of Style Transfer Artifacts

The superpixels can be used for efficient selection and editing of regions in the stylization generated by methods such as NST, particularly for correcting localized artifacts and tailoring local style elements to personal aesthetic preferences. The uniformly colored segments can be easily selected and their colors adjusted while the fine-structural texture is retained. For example, a selected region of superpixels can be color-matched to a different region using histogram matching, which can be used to remove unwanted stylization patterns or colors. In Fig. 9, a single-style PPN re-predicts the style's texture during the editing of segment colors to adapt the textural patterns to the new segment colors and structures in real time, enabling an integrated and interactive editing workflow (see supplementary material).

V. RESULTS AND DISCUSSION

A. Qualitative Comparisons

We compare our texture style transfer against other methods for example-based texture control on their ability to adapt texture style while retaining the overall composition and colors of the input image. Fig. 10 compares our textbased filter-optimization against CLIPstyler [11] and img2img Stable Diffusion [1]. As CLIPstyler also adapts the colors, we add a histogram-matching loss [34] (CLIPstyler-Hist) for a fair comparison. It is visible that CLIPstyler introduces new content structures and the strength of stylization is strongly dependent on the used text prompt, furthermore, colors deviate from the original even with histogram losses. Stable Diffusion, on the other hand, introduces strong variations from the original image content. Fig. 11 compares our image-based filter optimization against STROTTS [10] with histogram-matching loss [34], Gatys NST [2] with color-preserving loss [12], and the Stylized Neural Painting (SNP) style transfer [6]. Similarly to the CLIP-optimized parameters, our method is able to preserve more colors and is spatially more homogeneous than the preceding methods.

B. Quantitative Comparisons

a) Style-matching Capabilities: Tab. II compares stylematching capabilities of our parameter prediction and optimization to their respective baselines. While the singlestyle NST [28] is better in terms of content preservation compared to PPN_{sst}, which is expected as the PPN operates on the segmented stylized image, the style preservation is close. For the arbitrary style PPN, the content preservation is almost on par with its baseline while even improving on the style loss score. For optimization, we measure the ℓ_1 to its target (generated by STROTTS [10]), and achieve very close matching. Compared to the stylization pipelines employed by [9], we achieve similar values, even though their method does not use loss constraints on the parameter masks.

b) Mask Editability: Tab. III compares the effect of the Total variation loss \mathcal{L}_{TV} on mask noisiness. Masks with less noise are typically more editable. Compared to our method without \mathcal{L}_{TV} , the noise in the masks is reduced by more than two orders of magnitude when optimized with \mathcal{L}_{TV} , as measured by noise σ . The masks produced by the oilpaint pipeline of Loetzsch et al. [9] are even more noisy.



Fig. 10: Comparisons to related methods for text-based generation. Text prompts, per row, are 1) "round brushstrokes in the style of monet", 2) "starry night". Please zoom in to compare details.



Fig. 11: Comparisons to related methods for image-based style transfer. Please zoom in to compare details.

TABLE II: Comparison to baselines. To assess decomposition quality, we measure content loss \mathcal{L}_c and style loss \mathcal{L}_s [2] on 10 NST styles and 20 content images (see suppl. material). PPN_{sst} and PPN_{arb} are compared to their pixel-predicting baselines. Optimization is futhermore measured in ℓ_1 distance to its target and compared with the pipelines of Loetzsch et al. [9]. We use \mathcal{L}_{TV} with $\lambda_{tv} = 0.2$. The segmentation stage S uses SLIC [5] with s = 1000 and s = 5000 segments.

			U
NST Method	\mathcal{L}_c	\mathcal{L}_s	ℓ_1^{-1}
PPN _{sst} (s=1K 5K)	0.092 0.082	0.068 0.051	-
Johnson NST [28]	0.063	0.044	-
PPN _{arb} (s=1K 5K)	0.084 0.088	1.434 1.294	-
SANet NST [29]	0.081	1.434	-
Optim. (s=1K 5K)	0.046 0.047	0.442 0.443	0.031—0.026
Watercolor [9]	0.056	0.704	0.036
Oilpaint [9]	0.048	0.426	0.029

 1 The ℓ_1 distance is computed to the output of STROTTS [10]

TABLE III: Comparison of noise levels in parameter masks. Our pipeline optimized with and without \mathcal{L}_{TV} is compared against the oilpaint pipeline of Loetzsch et al. [9]. The experimental setup is the same as used for Tab. II.

Pipeline	σ of estimated noise ¹	\mathcal{L}_{TV}
Ours	0.0039	0.0041
Ours w/o \mathcal{L}_{TV}	0.0858	0.0514
Oilpaint [9]	0.1015	0.0614
1		

 1 We use skimage estimate_sigma to estimate Gaussian noise σ

c) Ablation Study: Tab. IV provides an ablation study using the experimental setup also used for Tab. II, to determine if all filters are necessary in our pipeline by removing filters and computing the ℓ_1 to its target I_t after optimization. We observe that removing any of the filters has a significant impact on the ability to match arbitrary styles. Please refer the supplementary material for the full ablation study and qualitative examples.

TABLE IV: Filter ablation study. The effect of removing filters on the ℓ_1 distance to the target is measured. Refer to suppl. material for study setup.

full	w/o Bilat.	w/o Bump	w/o XDoG	w/o Contrast
0.0255	0.0305	0.0372	0.0358	0.0342

C. Limitations

While filter optimization can accurately match a target image, the quality of decomposition into painterly attributes is contingent on several factors. First, the parameters controlling a filter must produce distinct effects on the output, i.e., be nonoverlapping with other filters as the optimization is guided towards generating plausible outputs and editable masks, but does not take the semantic meaning of parameters into account. Second, the options for explicit painterly control are limited to the pipeline's parameters, adding other differentiable filters requires manual configuration. Further, while our method allows global adaption of textures using geometric abstraction techniques and example-based controls which are visually distributed uniformly, it does not guarantee to maintain statistical textural properties such as spatial homogeneity.

VI. CONCLUSIONS

We presented a lightweight, differentiable pipeline for texture editing using text prompts and image examples, and demonstrated its integration with controllable geometric abstraction techniques. Our approach demonstrates the benefits of a decomposed representation of texture for interactive and exploratory editing. This technology and its future enhancements enable the bridging of established image editors and tools with modern image generation techniques. Joint optimization of parameters and shape primitive placement holds the potential for even better decomposition of texture and style and is an avenue for future research.

ACKNOWLEDGMENTS

This work was funded by the German Federal Ministry of Education and Research (BMBF) (through grants 01IS15041 - "mdViPro" and 01IS19006 - "KI-Labor ITSE"), and the German Federal Ministry for Economic Affairs and Climate Action (BMWK) (through grant 16KN086401 - "PO-NST").

REFERENCES

- [1] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "Highresolution image synthesis with latent diffusion models," in Proc. CVPR, 2022, pp. 10684-10695.
- [2] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image Style Transfer Using Convolutional Neural Networks," in Proc. CVPR, 2016, pp. 2414-2423.
- [3] X. Huang and S. Belongie, "Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization," in Proc. ICCV, 2017, pp. 1501-1510.
- [4] A. Hertzmann, "Toward modeling creative processes for algorithmic painting," arXiv preprint arXiv:2205.01605, 2022.
- R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, [5] "Slic superpixels compared to state-of-the-art superpixel methods," IEEE TPAMI, vol. 34, no. 11, pp. 2274-2282, 2012.
- [6] S. Liu, T. Lin, D. He, F. Li, R. Deng, X. Li, E. Ding, and H. Wang, "Paint Transformer: Feed Forward Neural Painting with Stroke Prediction," in Proc. ICCV, 2021, pp. 6598-6607.
- Z. Zou, T. Shi, S. Qiu, Y. Yuan, and Z. Shi, "Stylized neural painting," [7] in Proc. CVPR, 2021, pp. 15689-15698.

- [8] J. E. Kyprianidis, J. Collomosse, T. Wang, and T. Isenberg, "State of the "Art": A Taxonomy of Artistic Stylization Techniques for Images and Video," IEEE TVCG, vol. 19, no. 5, pp. 866-885, 2012.
- [9] W. Lötzsch, M. Reimann, M. Büssemeyer, A. Semmo, J. Döllner, and M. Trapp, "WISE: Whitebox Image Stylization by Example-Based Learning," in Proc. ECCV, 2022, pp. 135-152.
- [10] N. Kolkin, J. Salavon, and G. Shakhnarovich, "Style Transfer by Relaxed Optimal Transport and Self-Similarity," in Proc. CVPR, 2019.
- [11] G. Kwon and J. C. Ye, "Clipstyler: Image style transfer with a single text condition," in Proc. CVPR, 2022, pp. 18062-18071.
- [12] L. A. Gatys, A. S. Ecker, M. Bethge, A. Hertzmann, and E. Shechtman, "Controlling Perceptual Factors in Neural Style Transfer," in Proc. CVPR, 2017, pp. 3730-3738.
- [13] M. Reimann, B. Buchheim, A. Semmo, J. Döllner, and M. Trapp, "Controlling strokes in fast neural style transfer using content transforms," The Visual Computer, pp. 1-15, 2022.
- [14] 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 Proc. ECCV. 2018.
- [15] A. Radford, J. W. Kim et al., "Learning transferable visual models from natural language supervision," in Proc. ICML, 2021, pp. 8748-8763.
- [16] A. Bousseau, M. Kaplan, J. Thollot, and F. X. Sillion, "Interactive watercolor rendering with temporal coherence and abstraction," in Proc. NPAR, 2006, pp. 141-149.
- [17] A. Semmo, D. Limberger, J. E. Kyprianidis, and J. Döllner, "Image Stylization by Interactive Oil Paint Filtering," Computers & Graphics, vol. 55, pp. 157-171, 2016.
- [18] H. Winnemöller, S. C. Olsen, and B. Gooch, "Real-Time Video Abstraction," ACM TOG, vol. 25, no. 3, pp. 1221-1226, 2006.
- [19] Y.-Z. Song, P. L. Rosin, P. M. Hall, and J. P. Collomosse, "Arty shapes." in CAe, 2008, pp. 65-72.
- [20] L. Ihde, A. Semmo, J. Döllner, and M. Trapp, "Design space of geometry-based image abstraction techniques with vectorization applications," 2022.
- [21] A. Hertzmann, "Painterly Rendering with Curved Brush Strokes of Multiple Sizes," in *Proc. SIGGRAPH*, 1998, pp. 453–460.
- [22] Z. Huang, W. Heng, and S. Zhou, "Learning to paint with model-based deep reinforcement learning," in Proc. ICCV, 2019, pp. 8709-8718.
- [23] C. Tomasi and R. Manduchi, "Bilateral Filtering for Gray and Color Images," in Proc. ICCV, 1998, pp. 839-846.
- [24] H. Winnemöller, J. E. Kyprianidis, and S. C. Olsen, "Xdog: an extended difference-of-gaussians compendium including advanced image stylization," Computers & Graphics, vol. 36, no. 6, pp. 740-753, 2012.
- [25] B. T. Phong, "Illumination for Computer Generated Pictures," Communications of the ACM, vol. 18, no. 6, pp. 311-317, 1975.
- [26] M. Wang, B. Wang, Y. Fei, K. Qian, W. Wang, J. Chen, and J.-H. Yong, 'Towards Photo Watercolorization with Artistic Verisimilitude," IEEE *TVCG*, vol. 20, no. 10, pp. 1451–1460, 2014. [27] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization,"
- in Proc. ICLR, 2015.
- [28] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution," in Proc. ECCV, 2016.
- [29] D. Y. Park and K. H. Lee, "Arbitrary Style Transfer with Style-Attentional Networks," in *Proc. CVPR*, 2019, pp. 5880–5888.
- [30] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in Proc. ICLR, 2015.
- [31] T. Lin et al., "Microsoft COCO: common objects in context," CoRR, vol. abs/1405.0312, 2014.
- [32] J. Wei, S. Wang, Z. Wu, C. Su, Q. Huang, and Q. Tian, "Label decoupling framework for salient object detection," in Proc. CVPR, 2020, pp. 13 025-13 034.
- [33] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction," in Proc. ICCV, 2021, pp. 12179-12188.
- [34] R. Jonschkowski and O. Brock, "End-to-end learnable histogram filters," 2016.
- [35] K. Nichol, "Kaggle painter by numbers (wikiart)," 2016. [Online]. Available: https://www.kaggle.com/c/painter-by-numbers
- [36] P. L. Rosin, Y.-K. Lai, D. Mould, R. Yi, I. Berger, L. Doyle, S. Lee, C. Li, Y.-J. Liu, A. Semmo, A. Shamir, M. Son, and H. Winnemöller, "NPRportrait 1.0: A three-level benchmark for non-photorealistic rendering of portraits," CVM, vol. 8, no. 3, pp. 445-465, 2022.
- [37] R. Menkman, The glitch moment (um). Institute of Network Cultures Amsterdam, 2011.

Controlling Geometric Abstraction and Texture for Artistic Images -Supplementary Material

I. PPN TRAINING

a) Training and implementation: We trained both PPN_{sst} and PPNarb for 24 epochs using the MS-COCO dataset [31] for content images and the WikiArt dataset [35] for style images for PPN_{arb}. Both datasets contain approximately 80,000 training images. During training, we resize the shorter edge of the input images to 256 pixels and then randomly crop a region of size 256×256 pixels. The style images are also resized to a resolution of 256 pixels. The parameters of the effect pipeline have specified ranges that vary for each parameter, such as the contour opacity range of [0,1]. In order to improve training stability, we normalize the predicted parameter ranges to be between [-0.5, 0.5] utilizing the sigmoid function following the final convolutional layer of the PPN. The parameter ranges are subsequently scaled back to their original values prior to being input in the effect pipeline. To ensure performance on diverse settings of the segmentation stage S, we uniformly vary the global smoothing sigma and kernel sizes within the ranges of [1.0, 3.0] and [1, 7], respectively, as well as the number of segments within the range of [470, 5475]. We used the Adam optimizer [27] with a learning rate of 0.0005 for PPN_{sst} and 0.00001 for PPN_{arb}. A style weight between $[1 \times 10^{10}, 1 \times 10^{11}]$ is utilized for PPN_{sst}, depending on the style image with a content weight of $1x10^5$. We employ the same loss weights for PPNarb as Park et al. [29] use for SANet. Using the identity retaining loss of SANet [29] is not necessary for training our PPN_{arb}, as our lightweight filter pipeline is not capable of altering the semantic content structures given by I_a in a major way.

II. RUNTIME PERFORMANCE

In Tab. V, we present a comparison of the runtime performance of our proposed methods for various image sizes. Both PPNs are about equally fast as they only require a forward pass of the network. Furthermore the SLIC segmentation takes around 75% of the execution time, and does not have to be continously re-executed in interactive editing scenarios. On the other hand, the optimization method requires several seconds even for small images and is roughly compareable in runtime to other optimization-based NSTs.

III. ABLATION STUDY - FILTER PIPELINE

In our proposed lightweight pipeline, a set of filters were selected based on their ability to reconstruct texture after geometric abstraction and their capacity to offer meaningful artistic control parameters. The effectiveness of the pipeline is verified by its ability to match a wide range of styles, as shown in Table 2 of the main paper. To determine if all the chosen TABLE V: Runtime comparison of our parameter prediction and optimization methods, using a NVIDIA RTX 3090. The optimization is performed for 100 iterations. Note that computation time may not increase linearly with image size due to GPU under-utilization for small images. For the PPNs, we show runtime of the complete method, including target generation using feedforward NST [28], and segmentation using SLIC [5], and in parentheses the runtime for the PPN and effect only. Results are presented in seconds.

Image Size	Optimization	PPN _{sst}	PPN _{arb}
256x256	11.45	0.20 (0.08)	0.20 (0.08)
512x512	17.63	0.54 (0.12)	0.55 (0.13)

TABLE VI: Full ablation study results. In the first two rows, we show results for the full pipeline, all subsequent rows show the results for pipeline configurations with one filter removed. All rows except the first use \mathcal{L}_{TV} during optimization. All configurations were executed using 1000 and 5000 segments of SLIC segmentation, denoted as (1K|5K). The experimental setup is described in Sec. III.

Pipeline Config	ℓ_1 to ST [10]	\mathcal{L}_c	\mathcal{L}_s
w/o \mathcal{L}_{TV}	0.028 0.023	0.047 0.048	0.409 0.421
w. \mathcal{L}_{TV}	0.031 0.026	0.046 0.047	0.442 0.443
No Bilateral	0.031 0.031	0.046 0.047	0.442 0.448
No Bump Mapping	0.037 0.037	0.046 0.047	0.465 0.466
No XDoG	0.036 0.036	0.046 0.047	0.503 0.505
No Contrast	0.030 0.034	0.044 0.044	0.510 0.499

filters in the pipeline are indeed necessary for representing these styles, an ablation study was conducted, as reported in Table 4 of the main paper. The following sections provide the full details and results of the ablation study.

a) Study setup.: We selected 20 random content images from MS COCO [31] and 10 popular style images with fine and coarse grained texture - we show these style images and a selection of the content images in Figs. 12 and 16. We used the STROTTS [10] style transfer algorithm to generate a NST result and then optimized parameters to match this result using the ℓ_1 loss for \mathcal{L}_{target} . We tested different pipeline configurations, where each configuration except our full pipeline has one filter deactivated. For evaluation, we used the following metrics: VGG [30]-based content and style loss [2], and the ℓ_1 difference to the ground truth generated by STROTTS. We used the following hyperparameters throughout the study: content image size (long edge) of 512, style image size of 256, and content and style weights of 0.01 and 5000 respectively, 0.2 for λ_{TV} , and Adam with a learning rate of 0.01 for 500 iterations. We executed the study with 1000 and 5000 segments using SLIC [5] in the segmentation stage.



Fig. 12: Example results of the ablation study. Each column contains one stylization target with the results of different pipeline configurations.



(d) Color Amount (e) Edge Darkening (f) Wetness Scale

Fig. 13: Selected parameter masks of the watercolor [16] differentiable filter pipeline [9]. We show the masks after optimization to match a NST image (c). The parameters color Amount, Edge Darkening, and Wetness Scale are not used or do not display variation in values.

b) Results.: In Tab. VI, we show full results of our ablation study. Using \mathcal{L}_{TV} on the full pipeline only slightly increases the ℓ_1 loss to the STROTTS [10] ground truth, whereas removing one filter increases the loss value considerably. Example images from the ablation study are shown in Figure 12. Our proposed full pipeline, outlined in Section 3.3 of the main paper, and its optimization target, as generated by STROTTS [10], demonstrate a close match. The subsequent rows highlight that the removal of a single filter impairs the pipeline's capability to reconstruct details. All ablated configurations exhibit difficulties in reconstructing the clock face in the first column. The second column illustrates that the absence of bump mapping leads to a failure in reconstructing areas of high luminance, such as on the forehead and hat. The removal of any of the other filters results in difficulties in painting over segment boundaries, resulting in artifacts as seen in the forehead area. The third column demonstrates that small-scale contours are not properly retained. The last column illustrates the pipeline's inability to locally enhance contrast when the contrast filter is omitted.

c) Other pipelines.: We briefly test other pipelines for arbitrary style representation. The oilpaint and watercolor pipelines of Semmo [17] and Bousseau [16], as implemented in the differentiable framework of Loetzsch et al. [9] are not well-suited for editing. Next to the noisiness of their parameter masks (Table 3., main paper), they often contain redundant parameters, which do not contain any information after optimization. For example, Fig. 13 depicts that parameters such as edge-darkening, wetness-scale or color-amount do not contain locally adjusted values - even though one might for example expect colour amount to be increased in the saturated areas of the stylized image.

IV. STYLE-MATCHING CAPABILITIES

In Table 2 of the main paper, we quantitatively evaluated the style-matching capabilities of our proposed parameter prediction and optimization methods against their respective pixel-predicting and optimizing NST baselines. The results indicate that our methods are capable of achieving good stylematching performance. We provide additional visual evidence in Fig. 16 by showing the output of our parameter optimization and single and arbitrary style parameter prediction networks. The results are visually very similar to their baselines, e.g., refer to Fig. 17 for a comparison between the STROTTS [10] baseline and the parameter optimization. However, it is noted that the arbitrary style parameter prediction network (PPN_{arb}) sometimes overuses the bump-mapping parameter, resulting in an excessive prediction of specular oilpaint texture where the input NST would not have placed such elements. Moreover, we compare our filter pipeline with the style-specific differentiable pipelines implemented by Lötzsch et al. [9] in Fig. 17. We compare example results obtained from Lötzsch et al. [9] for the oil-paint [17] and watercolor [16] filter pipelines with our method. Both style-specific pipelines tend to have problems in matching the color in small regions, even though a dedicated hue adaption step is employed by these pipelines.

V. EDITING APPLICATION

We implemented an prototypical application that makes use of our decomposed shape and texture representation to edit results of a NST. We provide editing metaphors for the structural abstracted output I_a of the segmentation stage S(using SLIC [5]). This enables coarse granular edits of color and structure. The application further uses a PPN_{sst} to predict parameter masks P_M for the filter pipeline. After performing structural edits on I_a , the P_M can be re-predicted by PPN_{sst} to adapt fine details to these edits.

A. Editing Tools

The user interface allows selection of one or multiple individual segments to perform various forms of editing, which are detailed in the following:

- **Content Image Interpolation** Interpolates I_a with the content image I_c , enabling structure reconstruction from I_c by adjusting the colors of the affected segments.
- **Color Interpolation** Interpolates a region with a user-defined color, allowing for adjustment of a region's color.
- **Color Palette Matching** Matches the color palette of a destination region to the color palette of a source region, facilitating the matching of color patterns between regions while maintaining their structures.
- **Copy Region** Copies a region to another image region, making it possible to move a structure or style element.
- **Change Level of Detail** Alters the number of segments used to display a region, which can increase the level of detail for important regions like facial features or decrease the level of detail for backgrounds.



Output after Mask interpolation

Fig. 14: Illustration of the example editing process described in Sec. V-B. The top row displays the initial state, while the middle two rows present the intermediate results after the first and second editing step, respectively. The bottom row exhibits the final output after the parameter mask interpolation, alongside the content and style image used in the editing process. Our method allows changing structures of the image and the parameter masks adapt automatically after reprediction.



(a) Input (b) Segmented (NPtr [7]) (c) P_M Optimized (ℓ_1) (d) P_M re-predict (AST PPN) (e) Hand-painted by A. Lister

Fig. 15: Creating 8-bit art similar to the style of Adam Lister (e). The input image (a) is segmented by the neural painter [7] (NPtr) method using 300 strokes (b). After optimizing the parameter masks P_M with \mathcal{L} to re-add the details (c), they are further edited, e.g., by re-reprediction using the arbitrary style PPN (d).

B. Editing Example

The application's capabilities are demonstrated through an example usage, highlighting both structural and detail edits. To showcase an editing workflow, we depict the editing steps in Fig. 14. We use the candy style image to train both a Johnson NST and a PPN_{sst} and then use our application to stylize a content image (sourced from NPRP [36]). For each editing step (row two and three of Fig. 14), we provide the segmentation, the pipeline output, and the pipeline output after repredicting the visual parameters after the described edits have been made.

The first step of the editing process, depicted in the second row of Fig. 14, entails eliminating all circular style elements. This is achieved by selecting the entire face, excluding the mouth, nose, and eyes, and utilizing the *Content Image Interpolation* tool to remove circular structures. Since the content image lacks these circular structures, they are replaced by the original content. Next, the *Change Level of Detail* tool is applied to reduce the facial region's level of detail, followed by employing the *Color Palette Matching* tool selecting a yellow face region to reinstate the face color, ensuring consistency with the style image. Throughout this procedure, parameter masks encoding fine details remain unaltered. After repredicting the parameter masks using the PPN_{sst}, they adapt to the structural modifications, leading to the elimination of the initial circular structures in the parameter masks.

In the second editing step displayed row three of Fig. 14, the *Change Level of Detail* tool was used to increase the level of detail of the mouth, eyes, and nose. Upon re-prediction of parameter masks, only minor differences emerge, as the structure remains unchanged.

In the final step, the saliency mask is utilized to adaptively enhance the contrast parameter mask. This leads to an increased contrast within the saliency region, which, in this instance, corresponds to the woman's face.

VI. FURTHER RESULTS

a) Geometric Abstractions: The combination of the geometric abstraction stage with the subsequent filter-based stage allows for reintroduction of details, which can mimic the style of "8-bit" quantized artwork such as the watercolor paintings of Adam Lister, see Fig. 15. Direct ℓ_1 optimization may, however, cause more details to be reintroduced than desired, even though such "glitch-art" effects (Fig. 15c) have artistic purpose of their own [37]. Instead, the parameters can, for example, be repredicted using PPN_{arb} with $I_s = I_t$ to only retain the inputs' textures (Fig. 15d).

Further, in Fig. 21 we show the full teaser images for geometric abstraction editing. In Fig. 22 and Fig. 23, we vary the geometric abstraction while keeping the texture fixed.

b) Comparisons with related works: In Figs. 19 and 20 we provide further qualitative comparisons to text-based and image-based stylization methods.

c) Parameter Masks: In Fig. 24 we showcase a full set of parameter masks after optimization.

d) CLIPStyler Experiments: In Figs. 18 and 25 we experiment with different CLIPStyler-loss text-prompts.

e) Editing and Blending: In Fig. 26 we show the intermediate results of our interactive editing workflow (see supplemental video). In Fig. 27 we show results for blending with a saliency and depth mask.



(a) Content & Style

(b) Optimization

(c) PPN_{sst}

(d) PPN_{arb}

Fig. 16: Comparison of parameter optimization and prediction for the task of style transfer. For each method, we use the style target shown on the left. We compare optimization, the single style PPN (PPN_{sst}) and the arbitrary style PPN (PPN_{arb}). During parameter optimization (b), we optimize \mathcal{L}_{target} using ℓ_1 loss to the output of STROTSS [10]. Each method is capable of predicting or optimizing parameters that produce artifact-free textures and match the visual fidelity of their CNN-based counterparts. We show results predicted on 5000 segments of SLIC [5].



Content & Style

STROTSS [10] Target

Ours

Watercolor [9]

Fig. 17: Comparison of our proposed Arbitrary Style Pipeline with the differentiable, style-specific Oilpaint [17] and Watercolor [26] effect pipelines by Lötzsch et al. [9]. The pipelines were optimized to match the stylization result generated using the STROTSS [10] NST via ℓ_1 loss.





(a) Original

(b) "the scream"



(c) "foam"

(d) "icy frost"



(e) "impressionism by Monet" (f) "regular grid" Fig. 18: Texture style transfer of "rain princess" using CLIPstyler [11]-losses. PtF-Circle is used for segmentation.



(a) Original

(b) CLIPstyler [11]

(c) CLIPstyler-Hist



(d) Stable Diffusion [1]

(e) Ours (fine)

(f) Ours (coarse)



(g) Original



(h) Clipstyler [11]



(i) Clipstyler-Hist



(j) Stable Diffusion [1]

(k) Ours (fine)

(l) Ours (coarse)

Fig. 19: Further comparisons to related methods for text-based generation. Text prompts, per row, are 1) "icy frost" and 2) "triangles".



(b) Style



(c) Color preserv. NST [12]



(d) SNP [7]



(e) STROTSS [10]-Hist



(f) Ours



(g) Input





(i) Color preserv. NST [12]



(j) SNP [7]

(k) STROTSS [10]-Hist

(l) Ours

Fig. 20: Further comparison of image-based style transfer methods with color preservation and our proposed method. We use STROTSS [10] with Histogram Matching Loss [34].





(b) S=Ptf-rect. Texture is re-optimized with STROTSS [10].



(c) S=SNP - 8bit art blocks. Texture is reoptimized with STROTSS [10].



(d) S = PTf-Circles. Texture is reoptimized with CLIPStyler [11].

Fig. 21: Full images of the teaser figure. We compare different geometric abstraction primitives from segmentation using PaintTransformer [6] (Ptf) and stylized neural painter [6] (SNP) with subsequent reoptimization. See Fig. 3 of the main paper for the outputs of the segmentation method (I_a) of the respective images. We reoptimize with a canvas texture as target. In (c), local contour enhancements are made in a final step. In (d) we reoptimize with "round brushstrokes in the style of monet" as target.



(a) Input

(b) 5000 segments

(c) 1250 segments



(d) 700 segments (e) 300 segments (f) 100 segments (b). When reducing the number of segments (without re-optimization), the underlying coarse-structure is spatially quantized while the high-frequency details (i.e., texture) remain intact.





(b) SLIC + CLIPstyler Prompt: Triangles

Fig. 23: Comparing the influence of the segmentation stage after re-optimization by a text prompt.



Fig. 24: A complete set of parameter masks (P_M) after ℓ_1 optimization. We match the target (a) and use a SLIC [5] segmentation with 5000 segments. The Total Variation loss was utilized during the optimization process. To enhance visibility, some of the parameter masks have been adjusted to increase luminosity. We highlight these with an asterisk (*).



(a) CLIPstyler Prompt: Shiny Chrome



(b) CLIPstyler Prompt: Camera Bokeh



(c) CLIPstyler Prompt: Triangels



(d) CLIPstyler Prompt: Icy Frost

Fig. 25: Re-Optimization experiments using different text prompts. The input painting was segmented with PaintTransformer [6] and circle primitives.



(a) Global Re-Optimization using text prompt: Fire



(b) Global Re-Optimization using text prompt: Star





(c) Blending parameters for "Fire" in the face region with "Star" (d) Fine-tune global bump and specularity parameters to inin the background using depth mask. crease oiliness.

Fig. 26: An exemplary four-step iterative editing process demonstrating the usage of text prompts, parameter interpolation and global editing. Our method offers the advantage of allowing for adjustment of global and local parameter values at any point in the editing workflow. Our supplementary video illustrates the interactive creation of such images.



Fig. 27: High-res figure 7 of the main paper. First row shows saliency parameter blending, second row shows depth blending. In the first row, we use "Starry Night" by Van Gogh as our style image and interpolate with the prompt "fire". In the second row, we use "Woman with a Hat" by Matisse as our style image and interpolate with the prompt "drops" and furthermore use the depth mask for adaptive circle sizes in the PaintTransformer.