Neural Smoke Stylization with Color Transfer

Fabienne Christen, Byungsoo Kim, Vinicius C. Azevedo and Barbara Solenthaler

ETH Zurich

Abstract

Artistically controlling fluid simulations requires a large amount of manual work by an artist. The recently presented transport-based neural style transfer approach simplifies workflows as it transfers the style of arbitrary input images onto 3D smoke simulations. However, the method only modifies the shape of the fluid but omits color information. In this work, we therefore extend the previous approach to obtain a complete pipeline for transferring shape and color information onto 2D and 3D smoke simulations with neural networks. Our results demonstrate that our method successfully transfers colored style features consistently in space and time to smoke data for different input textures.

CCS Concepts

• ***Computing methodologies*** $\rightarrow$ ***Physical simulation; Image processing; Neural networks;***

1. Introduction

Physically-based fluid simulations have become an integral part of special effects in movie production and graphics for computer games. However, artistic control of such simulations is not well supported and hence remains tedious, resource intensive and costly. Recent work on fluid control include target-driven optimization to find artificial forces to match given keyframes [TMPS03, PM17] and velocity synthesis methods that allow augmentation with turbulent structures [TKG08, SSN18]. With a neural flow stylization approach [KAGS19], more complex styles and semantic structures have been transferred in a post-processing step. Features from natural images are transferred onto smoke simulations, enabling general content-aware manipulations ranging from simple patterns to intricate motifs. The method is physically inspired, as it computes the density transport from a source input smoke to a desired target configuration. Stylizations from different camera viewpoints are merged to compute a 3D reconstruction of the smoke. While structural information is successfully transferred onto smoke data, color information was omitted. However, transferring texture information represents a valuable control tool for artists to change the appearance of a fluid. Our work therefore extends the transport-based neural flow stylization of Kim et al. [KAGS19] with a subsequent color optimization step that allows artists to control both style and color based on example images. The application is related to [JFA*15] that uses a flow-guided synthesis approach to transfer textures onto fluids.

Flow stylization approaches extend existing image style transfer methods with spatio-temporal constraints. In the image processing literature, [GEB16] automated the style transfer with neural networks and introduced several ways for the user to control the stylization effects [GEB*17]. Multiple follow-up works added new terms to the loss function of the original method to enhance the performance of the method. Histogram loss [RWB17] prevents instabilities in the form of varying brightness and contrast throughout the stylized image and avoids washed out results, Laplacian loss [LXNC17] preserves low-level details of the content image and a regularization term for photorealistic style transfer [LPSB17] overcomes distortion problems that appear with the original loss function. [RDB18] explored style transfer for video sequences ensuring the resulting frames to be temporally coherent and stable.

The optimization of three-dimensional smoke data is possible through the use of a differentiable renderer. A differentiable renderer enables the computation of derivatives [LB14], and recent approaches presented a multipurpose differentiable ray tracer that integrates various parameters such as camera pose, scene geometry, materials, and lighting parameters [LADL18, MNDJ19]. A lightweight and efficient differentiable renderer can be used in our case, as for flow stylization only the main flow structures need to be represented [KAGS19].

2. Preliminaries

Our approach for colorized smoke stylization is based on the original neural style transfer for images [GEB16] and the transport-based neural style transfer for fluid simulations [KAGS19], which are briefly introduced in the following.

2.1. Neural Style Transfer

Neural Style Transfer (NST) is the process of synthesizing an image I from a style image I_S and a content image I_C through optimization using a convolutional neural network (CNN). The CNN

© 2020 The Author(s)
Eurographics Proceedings © 2020 The Eurographics Association.

DOI: 10.2312/egs.20201015

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

is trained for natural image classification and its layers provide the feature space for the stylization. Using this CNN, the stylization can be formulated as an optimization problem as

$$I^* = \arg\min_I \alpha \mathcal{L}_c(I, I_C) + \beta \mathcal{L}_s(I, I_S), \tag{1}$$

where $\mathcal{L}_c$ is the content loss, $\mathcal{L}_s$ is the style loss and α and β are weighing factors. The content loss is spatially aware and aims at preserving the overall structure of I_C in the synthesized image. The style loss, on the other hand, optimizes for style structures independently of their image position. Let F_I^l be the feature representation of image I on layer l. The content loss $\mathcal{L}_C$ and the style loss $\mathcal{L}_S$ can then be formulated as

$$\mathcal{L}_c(I, I_C) = \sum_l (F_I^l - F_{I_C}^l)^2 \tag{2}$$

$$\mathcal{L}_s(I, I_S) = \sum_l (G_I^l - G_{I_S}^l)^2 \tag{3}$$

where $G_X^l = (F_X^l)^T (F_X^l)$ is the Gram matrix of the feature representation on layer l of an image X.

2.2. Transport-Based Neural Style Transfer

Transport-Based Neural Style Transfer (TNST) extends the original NST algorithm to transfer the appearance of a given image to flow-based smoke density. As opposed to NST where the stylized image is optimized, the optimization formulation for TNST outputs a velocity field. Consequently, no image pixels are modified directly. Instead, the input density d is transported by the optimized velocity field v^* to obtain the final stylized density d^*. Both v^* and d^* are obtained through optimization analogously to Equation 1 with

$$v^* = \arg\min_v \alpha \mathcal{L}_c(\mathcal{R}_\theta(\mathcal{T}(d,v)), I_C) + \beta \mathcal{L}_s(\mathcal{R}_\theta(\mathcal{T}(d,v)), I_S) \tag{4}$$

$$d^* = \mathcal{T}(d, v^*). \tag{5}$$

The transport function $\mathcal{T}(d,v)$ advects the density by the given velocity. The renderer $\mathcal{R}_\theta(d)$ renders a 2D greyscale image of the density d at viewpoint angle θ. Several viewpoints can be selected for the optimization to avoid distortions in the final stylized 3D density d^* (see [KAGS19]). The loss functions $\mathcal{L}_C$ and $\mathcal{L}_S$ maintain their definitions from Equation 2 and 3. The content loss can be neglected in our case, as we only have a style image and there is no content that needs to be preserved.

To extend the single frame stylization to multiple frames in a time coherent way, TNST aligns the stylization velocities with the input velocities. This is done recursively for a pre-defined window size. Increasing the window size enhances smoothness between consecutive frames, but simultaneously leads to larger memory requirements due to the recursive nature of the velocity alignment.

3. Method

Our method uses both NST and TNST as illustrated in Figure 1. In a first step, TNST is applied to the input frames of the smoke simulation to transfer structural information. This step corresponds to the approach of Kim et al. [KAGS19], and optimizes density values at each point. In a second step, we apply a color style optimization that modifies the color at each point while keeping the density values constant.

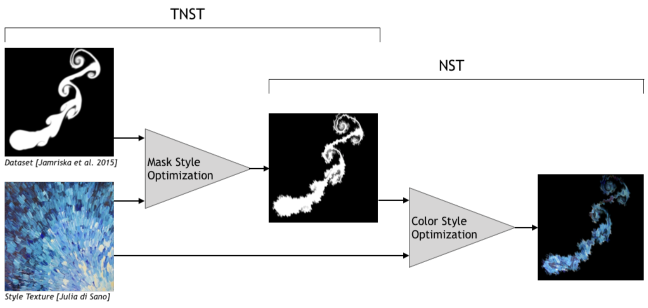


Figure 1: *Two-step pipeline for colorized image style transfer.*

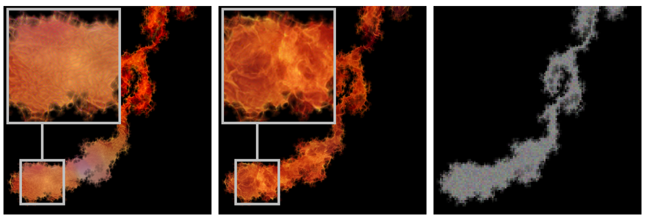
3.1. Color Style Optimization

In the second step of the pipeline, color is added to the stylized mask d^* from the previous step. This part creates and optimizes color channels for d^*, but does not further modify the density mask. The colorization process is performed using the original NST algorithm with a few alterations. Again, the desired style is given by the style image I_S and there is no content to preserve or transfer. Hence, we formulate the color style optimization as a simplified version of Equation 1 without content loss:

$$d^*_{RGB} = \arg\min_d \mathcal{L}_s((\mathcal{R}_{RGB,\theta}(d), I_S). \tag{6}$$

Since color information is now part of the optimization, the renderer $\mathcal{R}_{RGB,\theta}(d)$ computes a 2D color image from a viewpoint θ.

The proper initialization is crucial for the success of the color style optimization. As opposed to the original NST, there is no content loss, so any bias that is introduced in the initial condition can persist in the output. When starting the color style optimization from the stylized density d^*, the initial pixel values of the area that will be stylized are close to white. This leads to washout effects as shown in Figure 2(a). In Figure 2(b), the stylized mask d^* is initially multiplied pointwise with a white noise mask (Figure 2(c)). This initial condition converges to a satisfying result.



(a) Output from density (b) Output from noise (c) Noise mask

Figure 2: *Initialization with a) stylized density only, b) density combined with the noise mask of c).*

The color style optimization needs to be constrained to only optimize on the pixels that actually contain density. We obtain a guidance mask T^l by downsampling d^* to the size of each layer l that was selected for the style feature extraction and apply it to the style feature representation on layer l with [GEB*17]

$$\hat{F}^l_{G(I)} = T^l \circ F^l_{G(I)}, \tag{7}$$

where $\circ$ denotes element-wise multiplication. This way of guiding the stylization will lead to some overflow at the boundaries, because the receptive fields of neurons near the boundaries can overlap the masked out regions. This overflow can be removed from the final stylized density by applying the guidance mask once in the end.

3.2. Rendering

The greyscale renderer $\mathcal{R}_\theta$ and color renderer $\mathcal{R}_{RGB,\theta}$ are part of the optimization pipeline and therefore need to be differentiable and lightweight. $\mathcal{R}_\theta$ renders the smoke by calculating the pixel intensity along a ray in normal direction to the camera as proposed by [KAGS19]. More specifically, the transmittance $\tau(x)$ and the intensity I at each image pixel ij are defined as [FWKH17]

$$\tau(x) = e^{-\gamma \int_0^{r_{max}} d(r_{ij}) dr} \tag{8}$$

$$I_{ij} = \int_0^{r_{max}} d(r_{ij}) \tau(r_{ij}) dr. \tag{9}$$

The transmittance factor γ defines how much light is lost due to absorption and scattering, $d(x)$ evaluates the amount of density at point x, r_{ij} is the ray through pixel ij normal to the camera and r_{max} is the length of the ray. For the color style optimization, we extend this formulation to support color fields. The $C = \{R, G, B\}$ emission values at each pixel ij are computed by

$$C_{ij} = \int_0^{r_{max}} C(r_{ij}) d(r_{ij}) \tau(r_{ij}) dr. \tag{10}$$

The density d_{ij} is multiplied into the emitted colors and can be seen as the emission factor at each point. Note that the *RGB* emission values are normalized to $[0..255]$. The impact of the transmittance value on the greyscale and colorized results is shown in Figure 3.

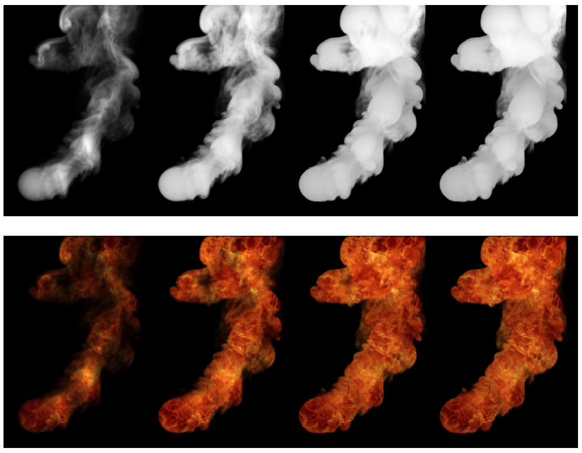
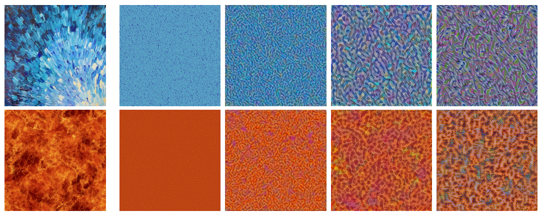


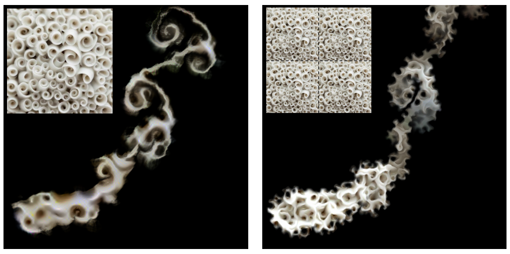
Figure 3: *Output of the greyscale renderer $\mathcal{R}_\theta$ (top) and color renderer $\mathcal{R}_{RGB,\theta}$ (bottom) for different transmittance factors* $\gamma = [0.001, 0.1, 0.5, 1]$.

3.3. Controlling the Stylization

We used the VGG-19 network [SZ15] for the feature extraction, which consists of 19 layers and has been trained for natural image classification. The stylization can be controlled by selecting layers in the CNN. The deeper a layer is positioned in the CNN, the higher is the complexity of the extracted features, as illustrated in Figure 4(a) for two different input images. The shallow layers optimize for low-level features, while deeper layers generate high-level features. The size of the stylized features depends on the size of the input image. Tiling can be used to progressively increase the input size to generate smaller scale structures as shown in Figure 4(b).



(a) *Feature complexity increases with deeper layers of the CNN ('relu1_1'.. 'relu4_1').*



(b) *Structure size can be controlled by tiling the input [spi].*

Figure 4: *Stylized structures can be controlled by selecting corresponding layers in the CNN (a) and tiling the input image (b).*

4. Results

We implemented the stylization with TensorFlow and used the Adam optimizer with a learning rate of 0.5 and 1 for the 2D and 3D examples, respectively, for 300 iterations. For our results, we selected the layers '*relu2_1*' and '*relu3_1*' of the VGG-19 network for the feature extraction. The results were evaluated on a Xeon E5-2630v4 CPU (4 cores) and a NVIDIA GTX 1080 Ti 11GB GPU. The mask and color style optimizations take 20 and 60 seconds per frame, respectively, for a resolution of 800x800.

We applied the style and color transfer to the 2D smoke data set of [JFA*15] using different input images as shown in Figure 5. Color information is transferred coherently in space and time (see accompanying video sequences), and hence complements the mask stylization of [KAGS19]. The 3D results were computed with a data set of [KAGS19]. Figures 3 and 6 show the colorized outcome with the 3D pipeline that optimizes for multiple viewpoints as described in the original paper of [KAGS19]. The quality of the stylized and colorized structures highly depend on the resolution of the simulation. For 3D, we used a resolution of 200x300x300, resulting in less sharp structures than in 2D (800x800, Figure 5). In Figure 6 we compare the results of the 2D and 3D pipelines. In 3D some artifacts appear near the boundary; this happens because the optimization of the 3D data is more difficult: In 3D, many different emission values along a ray influence the color intensity at each pixel of the rendered image, while in 2D the intensity comes from a single value only.

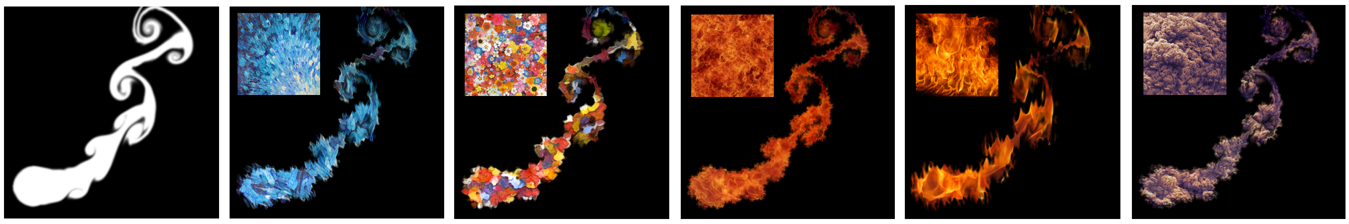


Figure 5: *2D single frame color stylization applied to a data set of [JFA*15] using different input images (blue strokes, flower, flame, fire [git] and volcano ©Richard Roscoe).*

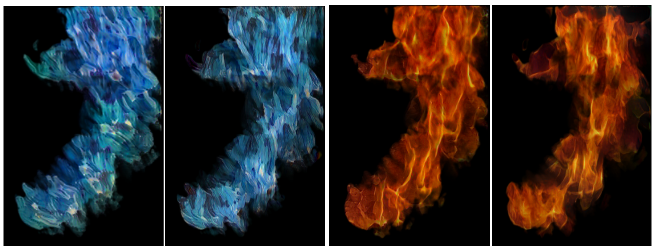


Figure 6: *3D (left) and 2D (right) color stylization applied to a data set of [KAGS19] for two input images (blue strokes and fire).*

5. Discussion and Conclusion

In this work we extended an existing flow stylization approach by adding color transfer. The color stylization is coherent in space and time, and can be applied to 2D and 3D smoke densities. Our optimization is performed in two sequential steps, but a single-step optimization could be explored as well. A current limitation of the 3D stylization pipeline is that it is computationally expensive, which makes stylization of entire simulation sequences and higher-resolution inputs challenging in practical settings. Our method directly optimizes for the stylized images during the training stage in an online fashion. Other research in the field of neural style transfer explores model-optimization based offline techniques. This type of style transfer moves the time intensive optimization into the training phase, thereby gaining the advantage of stylizing images in a single forward pass. Using this optimization method would greatly reduce the time that the stylization takes and render 3D stylizations more practical. Further, for the best outcome, the differentiable renderer that is used in the optimization should match the final high-quality rendering of the smoke. Our differentiable renderer could be adapted accordingly. Lastly, our neural approach needs to be evaluated with respect to existing texture synthesis methods.

6. Acknowledgments

We thank [JFA*15] for sharing the dataset. This work was supported by the Swiss National Science Foundation (Grant No. 200021_168997).

References

[FWKH17] Fong J., Wrenninge M., Kulla C., Habel R.: Production volume rendering: Siggraph 2017 course. In *ACM SIGGRAPH Courses* (2017), pp. 2:1–2:79. 3

[GEB16] Gatys L. A., Ecker A. S., Bethge M.: Image style transfer using convolutional neural networks. In *CVPR* (2016), pp. 2414–2423. 1

[GEB*17] Gatys L. A., Ecker A. S., Bethge M., Hertzmann A., Shechtman E.: Controlling perceptual factors in neural style transfer. In *CVPR* (2017), pp. 3730–3738. 1, 2

[git] Neural style transfer github image sources. `https://github.com/titu1994/Neural-Style-Transfer`. Accessed: Feb 2020. 4

[JFA*15] Jamriška O., Fišer J., Asente P., Lu J., Shechtman E., Sýkora D.: LazyFluids: Appearance transfer for fluid animations. *ACM TOG 34*, 4 (2015). 1, 3, 4

[KAGS19] Kim B., Azevedo V. C., Gross M. H., Solenthaler B.: Transport-based neural style transfer for smoke simulations. *ACM TOG (SIGGRAPH Asia) 38*, 6 (2019), 188:1–188:11. 1, 2, 3, 4

[LADL18] Li T.-M., Aittala M., Durand F., Lehtinen J.: Differentiable Monte Carlo ray tracing through edge sampling. *ACM TOG 37*, 6 (Dec. 2018), 222:1–222:11. 1

[LB14] Loper M. M., Black M. J.: OpenDR: An approximate differentiable renderer. In *ECCV* (2014), vol. 8695, pp. 154–169. 1

[LPSB17] Luan F., Paris S., Shechtman E., Bala K.: Deep photo style transfer. In *Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 6997–7005. 1

[LXNC17] Li S., Xu X., Nie L., Chua T.-S.: Laplacian-steered neural style transfer. In *ACM International Conference on Multimedia* (2017), pp. 1716–1724. 1

[MNDJ19] Merlin Nimier-David Delio Vicini T. Z., Jakob W.: Mitsuba 2: A retargetable forward and inverse renderer. *ACM TOG* (2019). 1

[PM17] Pan Z., Manocha D.: Efficient solver for spacetime control of smoke. *ACM TOG 36*, 4 (2017). 1

[RDB18] Ruder M., Dosovitskiy A., Brox T.: Artistic style transfer for videos and spherical images. *International Journal of Computer Vision 126*, 11 (2018), 1199–1219. 1

[RWB17] Risser E., Wilmot P., Barnes C.: Stable and controllable neural texture synthesis and style transfer using histogram losses, 2017. `arXiv:1701.08893`. 1

[spi] Spiral image source. `http://ardezart.com/?attachment_id=252`. Accessed: Feb 2020. 3

[SSN18] Syuhei Sato Yoshinori Dobashi T. K., Nishita T.: Example-based turbulence style transfer. *ACM TOG 37*, 4 (2018). 1

[SZ15] Simonyan K., Zisserman A.: Very deep convolutional networks for large-scale image recognition. In *ICLR* (2015). 3

[TKG08] Theodore Kim Nils Thuerey D. J., Gross M.: Wavelet turbulence for fluid simulation. *ACM TOG 27* (2008). 1

[TMPS03] Treuille A., McNamara A., Popović Z., Stam J.: Keyframe control of smoke simulations. *ACM TOG 22*, 3 (2003), 716. 1