Efficient Neural Style Transfer for Volumetric Simulations

JOSHUA AURAND, DisneyResearch|Studios, Switzerland
RAPHAEL ORTIZ, DisneyResearch|Studios, Switzerland
SILVIA NAUER, DisneyResearch|Studios, Switzerland
VINICIUS C. AZEVEDO, DisneyResearch|Studios, Switzerland

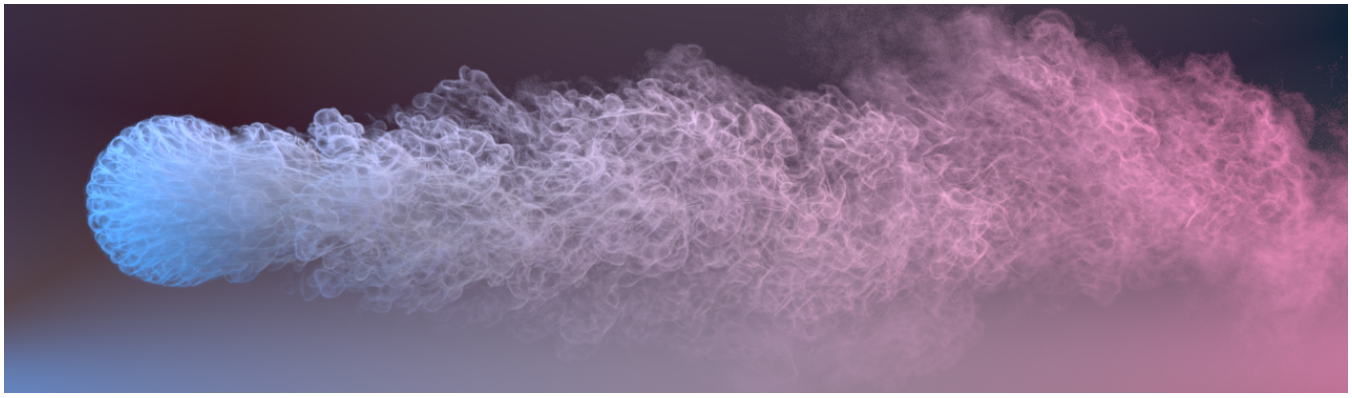


Fig. 1. **High Resolution dark-matter stylization.** Volumetric Style Transfer computed in a high-resolution a simulation of 645 × 609 × 1553. Inference time took only roughly one minute per frame.

Artistically controlling fluids has always been a challenging task. Recently, volumetric Neural Style Transfer (NST) techniques have been used to artistically manipulate smoke simulation data with 2D images. In this work, we revisit previous volumetric NST techniques for smoke, proposing a suite of upgrades that enable stylizations that are significantly faster, simpler, more controllable and less prone to artifacts. Moreover, the energy minimization solved by previous methods is camera dependent. To avoid that, a computationally expensive iterative optimization performed for multiple views sampled around the original simulation is needed, which can take up to several minutes per frame. We propose a simple feed-forward neural network architecture that is able to infer view-independent stylizations that are three orders of the magnitude faster than its optimization-based counterpart.

CCS Concepts: • **Computing methodologies** → **Physical simulation**.

Additional Key Words and Phrases: machine learning, style transfer, smoke simulation

ACM Reference Format:
Joshua Aurand, Raphael Ortiz, Silvia Nauer, and Vinicius C. Azevedo. 2022. Efficient Neural Style Transfer for Volumetric Simulations. *ACM Trans. Graph.* 41, 6, Article 257 (December 2022), 10 pages. https://doi.org/10.1145/3550454.3555517

Authors' addresses: Joshua Aurand, DisneyResearch|Studios, Switzerland, joshua.aurand@inf.ethz.ch; Raphael Ortiz, DisneyResearch|Studios, Switzerland, raphael.ortiz@disneyresearch.com; Silvia Nauer, DisneyResearch|Studios, Switzerland, silvia.nauer@bluewin.ch; Vinicius C. Azevedo, DisneyResearch|Studios, Switzerland, vinicius.azevedo@disneyresearch.com.

0730-0301/2022/12-ART257 $15.00
https://doi.org/10.1145/3550454.3555517

1 INTRODUCTION

Artistically manipulating physically-inspired simulations creates a magic connection between two worlds that can simultaneously express physical realism and an artistic expression conveying deeper meaning. Such connection can be seen when the full power of the dragon's gem is released to vanquish the *Druun* in Raya and the Last Dragon - the blast effect patterns took inspiration from Chladni patterns, cymatics and the symmetry of mandalas [Collier 2022]. For this case, the effects team employed a Neural Style Transfer (NST) [Navarro and Rice 2021] method that enabled artistic control of simulations with given input images. NST, in its original form [Gatys et al. 2016], is a popular technique for artistically stylizing an image while keeping its original content. It computes styles by extracting statistic filter activations of Deep Convolutional Neural Networks (CNNs) pre-trained for image classification tasks, providing a rich range of styles that can model both artistic [Johnson et al. 2016] and photo-realistic [Luan et al. 2017] style transfers.

Recent methods for computing volumetric Neural Style Transfer extend image-based ones by manipulating 3D fluid data through Eulerian (TNST) [Kim et al. 2019b] or Lagrangian (LNST) [Kim et al. 2020] frameworks. Both approaches rely on an iterative optimization which minimizes differences between filter statistics of a given target style and the style of a rendered smoke frame. Given a specified camera viewpoint, a differentiable volumetric renderer automatically enables the transfer of gradients computed in image-space to volumetric data. Temporally coherent smoke stylizations are obtained either by subsequently aligning and smoothing stylization velocity fields (TNST) [Kim et al. 2019a] or by smoothing particle

corrections over multiple frames (LNST) [Kim et al. 2020]. While the temporal coherency enforcement in LNST was more efficient than its Eulerian counterpart, it required a costly conversion step from smoke data to particles. This conversion is impractical for production pipelines, specially when employing such effect at large scales or when applying it to simulations that are continuously changing through artistic direction. In this paper, we revisit the original grid-based approach, proposing a suite of upgrades that enable volumetric stylizations that are significantly faster, simpler, more controllable and less prone to artifacts.

Firstly, we replaced TNST's costly *Mac-Cormack* [Selle et al. 2008] advection by a simpler linear mapping function. This simplification did not impact the quality of the results, since the advection order played a minor role on the quality of the stylizations. Crucial to TNST's slow performance is its inefficient temporal coherency step that required recursive advections to align adjacent stylization velocities. Thus, we replace TNST's Gaussian window smoothing by an Exponential Moving Average method that accumulates contributions from multiple frames, and therefore needs only one advection step to enforce temporal coherency. These upgrades allow volumetric stylizations without the need of a grid-to-particle conversion, with a speed-up of more than two orders of magnitude when compared with the original TNST approach.

Lastly, the energy minimization solved by previous methods is camera dependent. To avoid that, a computationally expensive iterative optimization performed for multiple views sampled around the original simulation is needed, which can take up to several minutes per frame. We propose a simple feed-forward neural network architecture that is able to infer view-independent stylizations that are three orders of the magnitude faster than its optimization-based counterpart. The summarized contributions of this paper include:

- A simplified and more efficient optimization formulation - costly advection algorithms were replaced by simpler mapping functions without loss of quality;
- An improved temporal smoothing algorithm that improves the transport-based algorithm running time by more than two orders of magnitude;
- An extension of the original transport-based approach to work directly with density values through a multiplicative factor;
- An efficient feed-forward architecture that is able to stylize volumetric simulations from arbitrary viewpoints.

2 RELATED WORK

Neural style transfer (NST) methods can be classified into *online* and *offline*. The former class transfers the style by iteratively optimizing an image, while the latter trains a generative Convolutional Neural Network (CNN), computing the stylized result with a single forward pass. In Section 3, we present an online approach that is highly tailored for efficiency, requires a low number of iterations, and therefore can stylize volumetric simulations with fast turnaround times. The choice of parameters (iterations, learning rate, number of octaves, activation layers) impacts the stylization tremendously, and having an efficient optimization scheme allows artists to try different styles without having to train a neural network. In Section 4, we present an offline view-independent approach that efficiently stylizes smoke simulations. We further discuss previous online and offline neural style transfer works and review methods that employ deep learning for fluid simulations.

Online Neural Style Transfer algorithms perform feature matching by iteratively solving an unconstrained minimization problem through back-propagation, modifying values independently to approximate second-order statistics of a given input. The seminal work of Gatys et al. [2016] enabled transferring styles between images, and since then, NST has been a topic of active research. Li et al. [2017a] proposed an efficient way to compute style statistics by measuring discrepancies between two distributions, improving computational efficiency over the traditional Gram matrix [Simonyan and Zisserman 2014]. Further improvements include reduction of instabilities and artifacts by histogram [Risser et al. 2017] and Laplacian [Li et al. 2017b] regularizations, tailoring stylization for portrait [Selim et al. 2016], and enabling long term correspondences in video sequences [Ruder et al. 2018].

Outside the realm of images, mesh stylization was enabled by differentiable renderers with approximate [Kato et al. 2018] and analytic [Liu et al. 2018] derivatives. Further works explored mesh-to-mesh [Yin et al. 2021] and text-to-mesh [Michel et al. 2021] stylizations. Handcrafted energy functions can be used to artistically manipulate meshes without Neural Networks filter activations for cubic [Liu and Jacobson 2019] and Gauss stylizations [Kohlbrenner et al. 2021]. Our work is built upon the transport-based NST (TNST) [Kim et al. 2019b], which proposes an online technique to stylize volumetric smoke data from example images. TNST supports complex styles generated from single images or from network activation maps, creating volumetric stylizations. Kim et al. [2020] further extended this method for better temporal coherency and improved efficiency by reformulating the problem into a Lagrangian framework modelling fluids as sets of particles.

Offline Neural Style Transfer. Computing an unconstrained optimization is computationally expensive, especially when considering 3D density fields. Thus, modern *offline* approaches train deep CNNs to efficiently obtain stylized results, and Johnson et al. [2016] proposed the first feed-forward approach for image stylization. Their method introduced perceptual losses for capturing high-frequency information, which are also widely employed in image super-resolution [Zhu et al. 2017]. Offline approaches were further improved by instance-normalization [Ulyanov et al. 2016], capturing styles across distinct texture scales [Wang et al. 2016], and fine-control over stroke brushes [Jing et al. 2018]. These architectures are limited to the style of a single image, and recent extensions focus on multiple [Dumoulin et al. 2017] or arbitrary [Li et al. 2019] images per-model stylization. Our work belongs to the same category as we use a feed-forward network to achieve high performance stylizations of smoke data. For a thorough review of both online and offline neural style transfer methods we refer to Jing et al. [2019].

Deep Learning for Fluids. Utilizing machine learning architectures to regress fluid representations was first demonstrated by Ladický et al. [2015]. The authors approximated a Lagrangian fluid solver by Regression Forests, achieving impressive efficiency in particle-based fluid computations. CNN-based architectures were employed

in Eulerian-based solvers to substitute the pressure projection step [Tompson et al. 2016; Yang et al. 2016], to synthesize flow simulations from a set of reduced parameters [Kim et al. 2019b] and to approximate steady-state velocity fields for predicting aerodynamic forces [Umetani and Bickel 2018]. Low-resolution fluid simulations are upsampled with patch [Chu and Thuerey 2017; Xie et al. 2018] and dictionary-based [Bai et al. 2019, 2021] approaches to better match their high-resolution counterparts. Differentiable simulation pipelines [Holl et al. 2020; Hu et al. 2018, 2019; Schenck and Fox 2018] that can be automatically coupled with deep learning [Um et al. 2020] architectures are a recent trend due to their natural ability to interface with computer vision. Deep learning was also employed to reconstruct volumetric flows from images with transport constraints and self-supervision [Franz et al. 2021], graph-based Lagrangian fluid simulation [Li and Farimani 2022], super resolution of unsteady data [Han and Wang 2022], interactive modelling of liquid [Yan et al. 2020] and smoke [Kim et al. 2022] with sketches, and for learning meaningful controls [Chu et al. 2021]. For a more complete review of Physics-based Deep Learning we refer to Thuerey et al. [2021].

3 AN EFFICIENT NEURAL STYLE TRANSFER FOR VOLUMETRIC SIMULATIONS

In this Section, we review previous volumetric style transfer algorithms [Kim et al. 2019a, 2020], highlighting differences with our current approach, which is purely Eulerian. The combined proposed changes increase the efficiency of the original TNST, while also making the method simpler and more ready to be integrated into production pipelines.

3.1 Simplified Transport by Linear Mapping

The Transport-Based Neural Style Transfer (TNST) [Kim et al. 2019a] extends the original optimization-based NST algorithm [Gatys et al. 2016] to support volumetric smoke stylization. TNST proposes a multi-level velocity-based approach that naturally follows the input simulation, since the optimization is constrained to deform densities indirectly through transport. A velocity field $\hat{\mathbf{v}}$ is iteratively optimized for stylizing an input density d, minimizing the loss

$$\hat{\mathbf{v}} = \arg\min_{\mathbf{v}} \sum_{\theta \in \Theta} \mathcal{L}(\mathcal{R}_\theta(\mathcal{T}(d, \mathbf{v})), \mathbf{p}), \tag{1}$$

where $\mathcal{T}$ is a transport function, $\mathcal{R}$ is a differentiable renderer, θ is a camera configuration from a set of camera views Θ, and $\mathbf{p}$ denotes user-defined parameters. To obtain volumetric 3D structures, the optimization integrates multiple camera configurations sampled within a specified range of settings, each optimizing the loss for an individual camera viewpoint. The loss function $\mathcal{L}$ is the style loss described by Kim et al. [2020].

TNST velocities $\mathbf{v}$ can be irrotational ($\mathbf{v} = \nabla\phi$), incompressible ($\mathbf{v} = \nabla \times \psi$) or a mixture of both. While incompressibility is desired for fluid simulations, it can be an overly restricting requirement for optimization, particularly when coupled with higher order integrators [Tang et al. 2021]. Since the optimizer is mostly concerned with matching screen-space gradients that get back-propagated to 3D through the shape/transmittance function of the input smoke, advection-order and incompressibility play a secondary role in stylization quality. This observation allows us to make the method more efficient by simplifying the transport function to be a Semi-Lagrangian method with a first-order Euler integrator. In practice, the optimizer finds a linear velocity field to warp densities as

$$\mathcal{T}(d, \mathbf{v}) \approx \mathcal{I}(d, \mathbf{g} + \mathbf{v}), \tag{2}$$

where $\mathcal{I}$ is an interpolation function and $\mathbf{g}$ represents grid density locations, respectively. All the results presented in this paper adopt trilinear interpolation. A comparison between different advection schemes is shown in Figure 2: the linear mapping is 1.5 times faster than Mac-Cormack advection used in [Kim et al. 2019a].

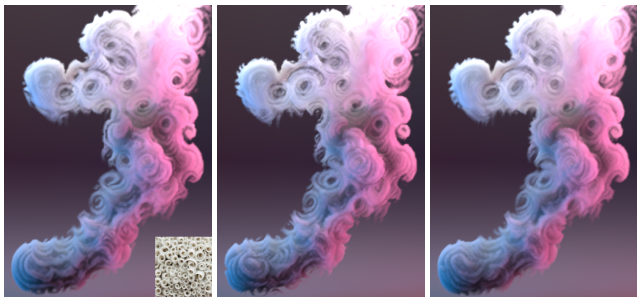


Fig. 2. **Stylization results using distinct advection schemes**: linear mapping (left), Semi-Lagrangian with RK-2 (middle) and MacCormack with RK-2 (right). All the stylizations were generated with an arbitrary velocity field (no incompressible or irrotational velocity fields).

3.2 An Exponential Moving Average Algorithm For Temporal Coherency

TNST optimizes Equation (1) iteratively per-frame due to memory limitations. To enforce temporal coherency, neighboring stylization velocities were first aligned by advection with the baseline simulation. After alignment, these velocities were combined by Gaussian smoothing with a compact kernel that spans w frames. While this approach was able to create temporally coherent volumetric stylizations with 2D input images, it had a crucial limitation: $(w^2 - 1)/4$ advections were required per single-frame iteration, which made the method extremely inefficient. A Lagrangian version of the algorithm (LNST) [Kim et al. 2020] greatly improves this limitation by recasting the optimization of Equation (1) to a particle-based framework as

$$\lambda^\circ = \arg\min_{\lambda^\circ} \sum_{\theta \in \Theta} \mathcal{L}(\mathcal{R}_\theta(\mathcal{I}_{p2g}(\mathbf{x}^\circ, \lambda^\circ), \mathbf{p})), \tag{3}$$

where λ° are per-particle attributes (e.g., density (ρ°) or positions ($\mathbf{x}^\circ$)), and $\mathcal{I}_{p2g}$ is a transfer function that maps particle attributes into a grid. LNST enforces temporal coherency by Gaussian smoothing of particle attribute changes, which is simple and efficient since it requires no alignment between adjacent frames.

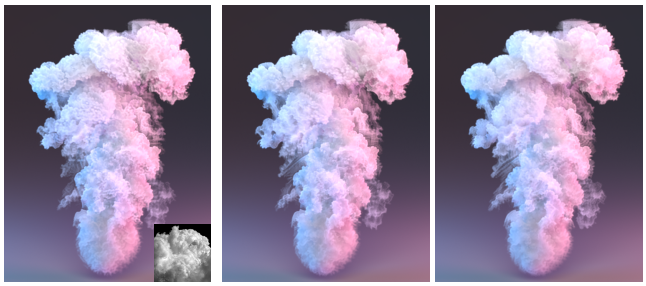
LNST, however, requires a pre-processing step for converting grid-based smoke simulations to particles. This conversion has to enforce a minimal amount of well-distributed particles through the entire simulation, which is crucial for both the efficiency of the stylization and for guaranteeing a good reconstruction quality of the original

grid-based smoke. The grid-to-particle conversion is implemented through a multi-level optimization process that it is time consuming and has several parameters that require careful tuning. It can be specially burdensome for productions [Navarro and Rice 2021], since it does not scale well for large simulations, generating considerable amounts of data in storage-bound production environments.

These shortcomings inspired us to revisit the original Gaussian smoothing temporal coherency algorithm. We experimentally observed that contributions from adjacent frames exponentially decrease as separation between frames increased. This inspired us to employ an Exponential Moving Average (EMA) [Smith 1997] algorithm, which consists of averaging accumulated contributions by

$$\hat{\mathbf{v}}_t^* = \begin{cases} \hat{\mathbf{v}}_0, & t = 0, \\ (1-\alpha)\,\hat{\mathbf{v}}_t + \alpha\,\mathcal{T}(\hat{\mathbf{v}}_{t-1}^*, \mathbf{u}_{t-1})), & t > 0 \end{cases} \tag{4}$$

where $\mathbf{u}_t$ and $\hat{\mathbf{v}}_t$ are the simulation and stylization velocities at the frame t, $\hat{\mathbf{v}}^*$ is the velocity after EMA smoothing, and α is a weight that determines how temporally smooth the stylization will be. Figure 3 shows the effect between using different EMA α weights during the stylization process.



$\alpha = 0$ $\alpha = 0.1$ $\alpha = 0.5$

Fig. 3. **Testing how different values of α affect EMA temporal coherency.** While lower EMA values yield sharper results, resulting patterns flicker through time (better represented on the accompanying video).

The biggest advantage of such an approach is that since it accumulates contributions from multiple frames, it requires only one advection step to enforce temporal smoothness of per-frame iterations. Thus, EMA smoothing allows us to directly use TNST without having to convert the grid into particles, while still being able to maintain temporal coherency and computational efficiency. Figure 4 shows a comparison between TNST and LNST methods for a smoke sequence that uses the same stylization parameters. We used a density-based version of TNST (Section 3.3) to ensure a fair comparison between both approaches.

One particular distinction of EMA is that each iteration of the stylization can cycle through the entire frame range of the input simulation. By swapping the time direction during the cycle, temporal coherency is implemented holistically, producing patterns that will be affected both by the previous and next frames relative to it. The algorithm that outlines the EMA smoothing coupled with a velocity-based stylization is shown in Algorithm (1). Note that we apply the optimization in multiple octaves to achieve multi-scale features in the stylized result, similar to [Kim et al. 2019a].

Algorithm 1: Velocity-based stylization with EMA

Data: Grid-based smoke densities d_t and velocities $\mathbf{u_t}$
Number of total iterations n_{iter}
Number of frames n_{frames}
Set of cameras Θ
Augmented rendering functions $\mathcal{R}_\theta$ for $\theta \in \Theta$
Style-loss parameters p
Maximum velocity norm $||\mathbf{v}_{max}||$
Result: Stylized densities $d_t^\star$ stored on a grid

$f_{begin} \leftarrow 1$; $f_{end} \leftarrow n_{frames}$; $\hat{\mathbf{v}}_t \leftarrow 0.0$
for $i \leftarrow 1$ **to** n_{iter} **do**
  for $t \leftarrow f_{begin}$ **to** f_{end} **do**
    Apply EMA Smoothing for $\hat{\mathbf{v}}_t$:
    $\hat{\mathbf{v}}_t \leftarrow (1-\alpha)\hat{\mathbf{v}}_t + \alpha\mathcal{T}(\hat{\mathbf{v}}_{t-1}, \mathbf{u}_{t-1})$
    Clip velocity norms according to $||\mathbf{v}_{max}||$
    Compute density by $d_t^\star = \mathcal{T}(d_t, \hat{\mathbf{v}}_t)$
    Reset total loss $\mathcal{L}_{tot} \leftarrow 0$
    for $\theta \in \Theta$ **do**
      Render image $I_\theta = \mathcal{R}_\theta(d_t^\star)$ for camera θ
      Accumulate style-loss $\mathcal{L}_{tot} \leftarrow \mathcal{L}_{tot} + \mathcal{L}(I_\theta; p)$
    end
    Compute gradient $\nabla_{\hat{\mathbf{v}}_t}\mathcal{L}_{tot}$
    Update velocity $\hat{\mathbf{v}}_t$ using $\nabla_{\hat{\mathbf{v}}_t}\mathcal{L}_{tot}$
  end
  swap(f_{begin}, f_{end})
end

3.3 Density-based Stylization and Constrained Optimization

The Lagrangian Neural Style Transfer has two modes: optimizing for densities carried by the particles (ρ°), or optimizing for the particles positions ($\mathbf{x}^\circ$). The majority of the results presented by the previous LNST method used per-particle density attributes, which were easier to tune, converged faster and had higher quality for high-frequency styles. Naively implementing the same approach in a grid-based framework will produce undesirable artifacts such as time-incoherent sinks and sources which may hinder the convergence of the optimization, especially with simulations that change significantly over time.

We therefore introduce an adaptation to TNST that only allows changes by modulating the input density with a scaling factor $\mathbf{s}$

$$\hat{\mathbf{s}} = \arg\min_{\mathbf{s}} \sum_{\theta\in\Theta} \mathcal{L}(\mathcal{R}_\theta(\mathbf{d}\cdot\mathbf{s}))\,, \mathbf{p}), \quad \textbf{s.t. } \hat{\mathbf{s}}(x) \in [s_{min}, s_{max}] \tag{5}$$

where $[s_{min}, s_{max}]$ is a bounded interval that constrains the minimum and maximum values of the density modulation in a certain grid voxel. The approach presented in the equation above is specially useful for view-independent stylizations (Section 4), or when using images that have fine-detail structures. This approach also allows us to better compare our method with EMA smoothing and the results shown in LNST (Figure 4). The changes needed in the Algorithm (1) to model the density-based stylization are minimal:

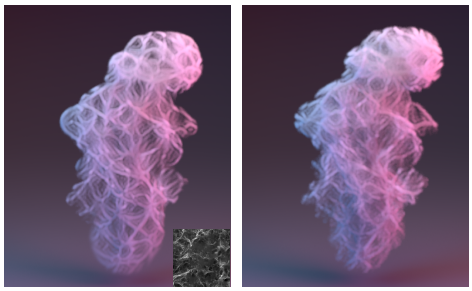


Fig. 4. **A comparison between density-based Eulerian (left) and Lagrangian (right) algorithms.**. Both approaches achieve similar results; the Lagrangian approach takes about 9 minutes for stylizing 90 frames, while the Eulerian one takes 12 minutes (the time to sample particles is not included). Due to memory limitations of the Lagrangian approach a lower resolution version of the *billowy smoke* was used.

the stylization velocity $\hat{\mathbf{v}}_t$ is replaced by a density modulation field $\hat{s}_t$, the transport $\mathcal{T}(d_t, \hat{\mathbf{v}}_t)$ is replaced by $d_t \cdot \hat{s}_t$ and scale factors $\hat{s}_t$ are clamped to the interval $[s_{min}, s_{max}]$.

We notice that implementing hard-limiters for changes during the optimization are also useful for the velocity-based version of TNST (Equation (1)). In this case, however, velocity magnitudes are constrained to be inside a parameter $||\hat{\mathbf{v}}(x)|| < \hat{\mathbf{v}}_{max}$. In [Kim et al. 2019a], the authors used an expanded and blurred density mask that modulates velocities in order to prevent the smoke to "leak-out" from its original shape. While effective, this choice created temporal coherency issues across the border of the smoke, due the smoke changing abruptly in its boundary regions. The velocity magnitude limiter is a more intuitive control, since artists can directly control the amount of stylization and also how much the stylization will expand the original simulation with a single parameter. Figure 5 show different values for the velocity magnitude, and their effects on the stylized result.

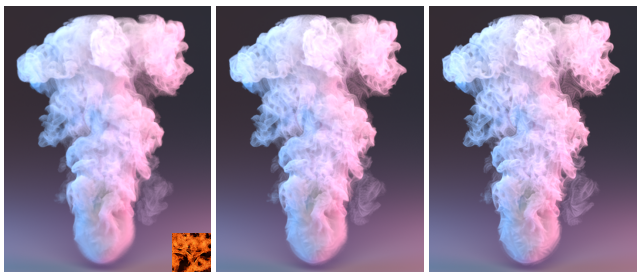


Fig. 5. **Comparison between different maximum velocity magnitude values**. Lower maximum velocity magnitudes limit the stylization, producing softer results while higher ones yield sharper outputs that can go beyond the original smoke shape.

4 A FEED-FORWARD NETWORK FOR VIEW-INDEPENDENT STYLIZATIONS

The volumetric stylization presented in the previous section is heavily dependent on the camera configuration: it matches the style for a set of specified cameras. While this approach allows screen-space control when using a single perspective camera, it fails to stylize for views that were oblivious to the optimizer. One example of how the camera setup impacts the volumetric stylization is shown in Figure 6 (top). To minimize view-dependent artifacts, the stylization can use a larger set of cameras per-frame to be sampled around either a pre-specified path [Kim et al. 2019a] or on a surface of the sphere enclosing the object [Liu et al. 2018]. When the camera is sampled on a sphere around the volume, we call it *omniview* camera distribution. Our omniview setup is created by first uniformly sampling on a sphere, and optimizing its positions to follow a Poisson-disk distribution. The issue, though, is that makes the stylization inefficient, up to several minutes per frame.

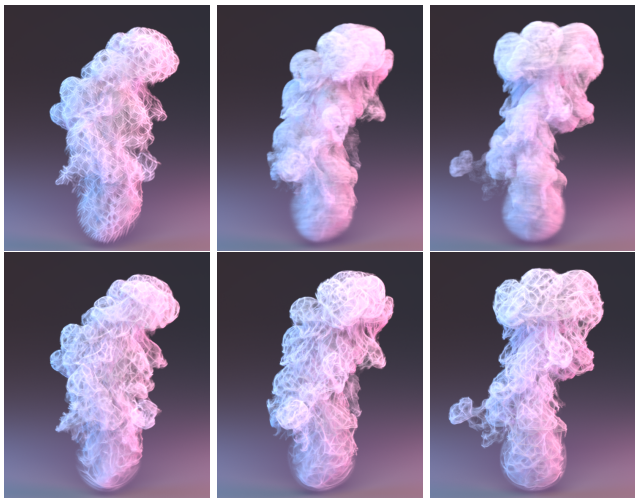


Fig. 6. **View-dependent (top) and view-independent stylizations (bottom).** The top sequence shows a result obtained with a single-view optimization; while the bottom sequence shows the result of applying the proposed feed-forward neural network for view-independent stylization.

To overcome this limitation, we implemented a feed-forward 3D convolutional neural network that takes volumetric density as input and outputs its stylized version. Our neural network trains to minimize either Equation (1) (velocity-based) or Equation (3) (density-based) in an unsupervised fashion, which avoids the need to generate input-output pairs to train the network in a supervised manner. We decided not to make a generalized architecture that can infer stylizations for arbitrary images [Guo et al. 2021], because not all images can produce style features that are consistent across multiple viewpoints. Furthermore, images that fail to represent the style faithfully in 3D can wash-out stylization features into often-times circular or isotropic patterns (Figure 11), which could create potential local minima to a neural network approach that takes arbitrary images as input. What works well in practice is first to test if an image is suitable for omniview stylization by running a specified configuration on a single frame with the efficient Eulerian approach described in the Section 3. This will be a good approximation for

what the artist will obtain once the network is finally trained. Results for the feed-forward omniview approach for a dark-matter style image are shown in Figure 6 (bottom): notice that since this style has an inherently volumetric nature to it, single-view features closely match with the omniview stylization.

By limiting training to a single stylization configuration (e.g., input image, size, etc), the network can remain lightweight. Figure 7 shows results of the network for training times that took between 2 and 18 hours. Notice that stylizing the full sequence for the same example with the optimization approach takes about 12 hours, and it cannot be reused for other examples. The training procedure takes individual patches of the input training dataset, stylizing them independently. Since the network is convolutional in its implementation, it can be evaluated for a different resolution than it was originally trained for — all results for the *billowy smoke* example (Figure 11) used a single patch with a resolution of $250 \times 500 \times 250$ during inference, while the corresponding models used patch sizes of 128^3 during training.

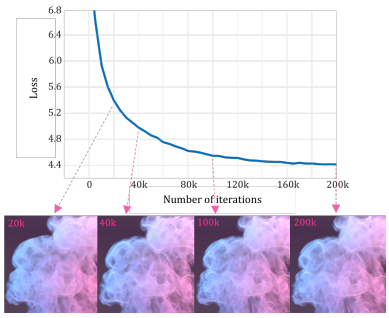


Fig. 7. **Training convergence vs quality of the feed-forward results**. As iterations progress and the loss gets lower, high-frequency details are more visible on stylized results.

Our feed-forward network generalizes surprisingly well, extending to distributions that were not in the training dataset. Figure 8 shows results of the *dark-matter* style applied to the *smokejet* and *bunny* datasets from [Kim et al. 2019a]. These datasets were generated with *Mantaflow* [Thuerey and Pfaff 2018], while the *billowy smoke* sequence that was used to train many results in this paper was generated with *Houdini*. Moreover, we do not explicitly enforce temporal coherence for the feed-forward network (as opposed to Xie et al. [Xie et al. 2018]). Instead we rely on the translational equivariance and continuity of the architecture output to produce temporally coherent stylizations. We hypothesize that since the loss is trained on the style-space of the rendered volume, it is better able to enforce filters that are transformation-invariant, generalizing well for sequences that were not seen during training time.

5 RESULTS

5.1 Implementation

The proposed volumetric style transfer algorithm was implemented in PyTorch [Paszke et al. 2019]. While we implemented both TNST and LNST methods (Figure 4), our work focused on improving the grid-based version of the volumetric style transfer, so the presented results do not include liquid stylizations. We used the *smoke jet* and *bunny* datasets from Kim et al. [2019a] for comparisons with TNST, which were simulated with *mantaflow* [Thuerey and Pfaff 2018]. The *billowy smoke* dataset was simulated with *Houdini's Pyro FX* billowy smoke template applied to a unit sphere source, and all parameters, apart from the grid-spacing which was set to 0.02, used default values. All scenes were rendered with Houdini's Mantra renderer. The ADAM optimizer [Kingma and Ba 2014] was used both for the direct optimization of smoke data and for training the feed-forward network. Our network consists of a simple encoder-decoder pair, and its architecture is shown in (Figure 9).

Differentiable Renderer. Our method builds upon the lightweight differentiable renderer from TNST, which measures how much of a light ray $\boldsymbol{r}$ gets transmitted through the smoke [Fong et al. 2017]. Transmittance τ and image pixel grayscale values I_{ij} are computed according to

$$\begin{aligned} \tau(\boldsymbol{x}, \boldsymbol{r}) &= e^{-\gamma \int_{\boldsymbol{x}}^{\boldsymbol{r}_{max}} d(\boldsymbol{r}) d\boldsymbol{r}} \\ I_{ij} &= \int_0^{\boldsymbol{r}_{max}} d(\boldsymbol{x}) \tau(\boldsymbol{x}, \boldsymbol{r}_{ij}) d(\boldsymbol{x}) \end{aligned} \tag{6}$$

where $\boldsymbol{r}_{ij}$ is a ray traced from the origin of the perspective camera through pixel ij with maximum length $\boldsymbol{r}_{max}$, $d(\boldsymbol{x})$ is the density at position $\boldsymbol{x}$ and γ is the transmittance absorption factor. This integral can be efficiently evaluated by ray-marching in the regular grid, which amounts to simple back-to-front summations that can conveniently be implemented in PyTorch.

Simply rendering the volumetric grid in its original form produces low-resolution images (e.g., a grid of $200 \times 300 \times 200$ voxels would typically generate an image of 200×300 pixels) that once fed through the image classification Neural Network result in low quality stylization gradients. Therefore, we double the resolution of all rendered images before the back-propagation step: this effectively reduces the size of the generated patterns while also guaranteeing high-quality stylization gradients. The size of the stylization patterns is then controlled by resizing the style image input with a user-specified scaling factor.

Regions of high density can be an issue for the optimization as they result in contrastless, uniformly white patches (inset image, right half), which hinders the creation of stylization patterns. We alleviate this problem by employing a differentiable approximation of histogram equalization which improves contrast in flat image regions (inset image, left half). Image pixels are transformed according to

$$\tilde{I}_{ij} = I_{ij} \cdot cdf(I_{ij}) \tag{7}$$

where $cdf(I_{ij})$ is the cumulative distribution function of the pixel grayscale values I_{ij}. We employ a cumulative distribution function computation that ignores black background pixels. This approximation is close to proper histogram equalization since it is only employed for regions with mostly bright pixels (i.e. $I_{ij} \approx 1$).

Perspective Camera. TNST used a set of orthographic cameras with jittered positions to create an illusion of a perspective camera. For single view, we avoid jittering by adding a perspective

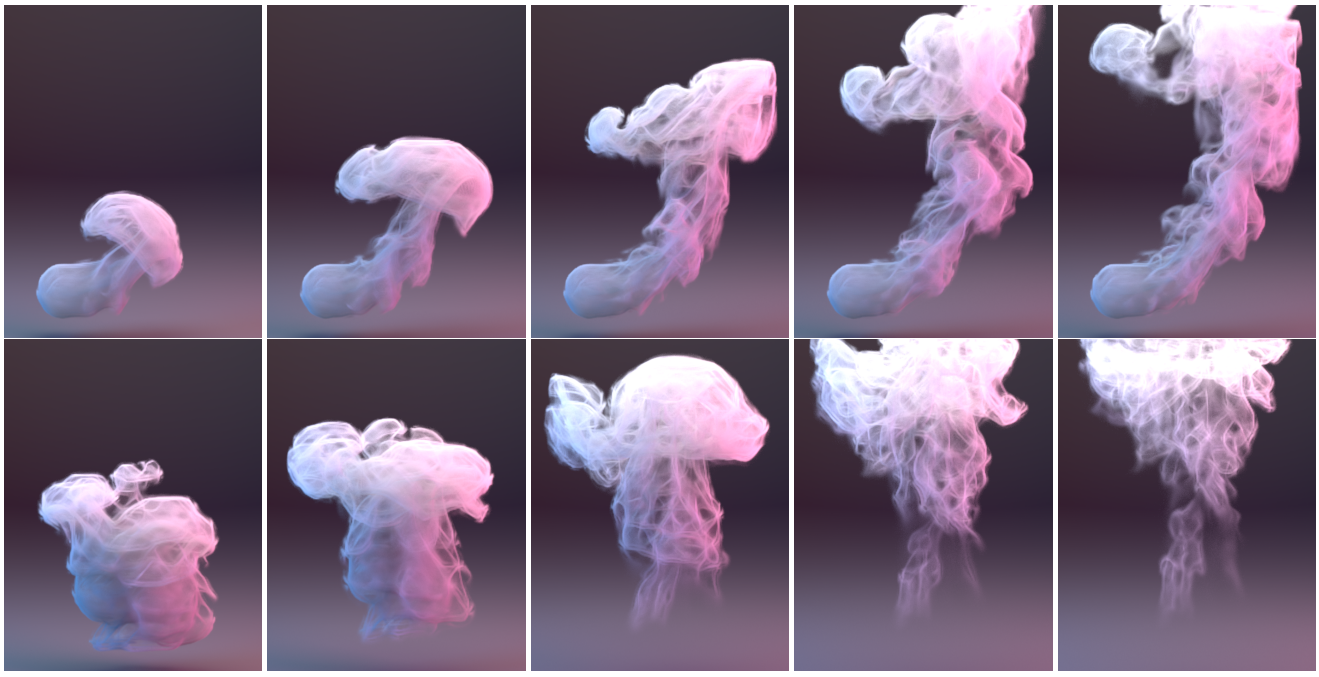


Fig. 8. **Generalization tests for the smoke jet and bunny sequences from [Kim et al. 2019a]**. The feed-Forward network was only trained on patches of the single *billowy smoke* sequence using the *dark matter* style, which demonstrate the generalization capabilities of the proposed approach.

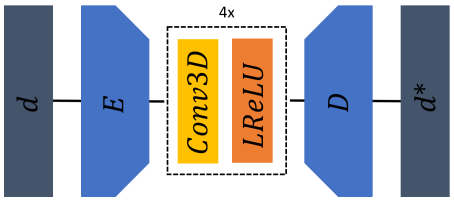


Fig. 9. **Feed-forward network architecture for view-independent stylizations.** We use a simple Encoder-Decoder architecture for our Feed-Forward Network. The Encoder E consists of two strided convolutions, a $5 \times 5 \times 5$ convolution followed by a $3 \times 3 \times 3$ convolution, to decrease the spatial resolution by a factor of 4. Once the original density is downsampled we apply 4 $3 \times 3 \times 3$ convolutions with stride 1. The Decoder D first applies two upsampling steps, trilinear upsampling and a $3 \times 3 \times 3$ convolution, followed by a final $3 \times 3 \times 3$ convolution reducing the number of channels. Every convolution, apart from the final one, is followed by a $LeakyReLU$ activation function with a negative slope of 0.01.

transformation into the rendering pipeline. Camera parameters are automatically extracted from Houdini setups to ensure maximum compatibility between the final rendered result and the internal differentiable renderer employed for stylization. The perspective transformation effectively re-samples a truncated pyramid grid from the original simulation - the warped grid can then be directly used for the transmittance computation by back-to-front summations. However, this introduces a complication; contrary to orthographic projection, the perspective rendering is depth dependent - a problem that is exacerbated if the input is moving because it changes the scale of the created patterns. To produce features scaled independently from the camera, stylization is done relative to the original smoke resolution, e.g., independent from zooming in or out. This is accomplished by back-projecting the object bounding box from camera to object space to measure its resolution. We resolve the inherent depth ambiguity by estimating the height and width at the bounding plane closest to the camera.

Performance comparison. The timings in this paper present a significant speed-up when compared with the original transport-based approach of Kim et al. [2019a]: some of our entire sequences of 120 frames took about 5 minutes, while previous work reported up to 13 minutes for a single frame. The aforementioned improvements in the advection and temporal coherency algorithms (Section 3) do not fully explain such a big performance improvement, and there are other aspects that significantly impacted the performance. Our adaptive rendering coupled with a single perspective camera enabled us to use smaller grid resolutions for rendering while also removing redundancy - TNST needed 9 camera samples jittered around the camera view, while we only need one perspective transformation. Moreover, when reimplementing the pipeline into PyTorch, we optimized previous memory throughput by asynchronous GPU/CPU calls that helped maintaining high GPU utilization. Due to these improvements, a performance speed-up can also be measured for our Lagrangian Neural Style Transfer implementation: it took 0.1 minutes per frame for the *billowy smoke* sequence, which employs a grid resolution of $175 \times 250 \times 175$ and particle count of 2M, against 0.45 minutes of a similar setting (*Colored smoke*) in [Kim et al. 2020].

Feed-Forward Training. Our training procedure extracts patches out of the simulation data. For the *billowy smoke* sequence we used patches with 128^3 voxels. Data augmentation is performed by randomly mirroring and rotating patches, which encourages the network to produce stylizations for arbitrary viewpoints. Gradient clipping is applied to avoid inconsistencies created by sparse volume patches that can not be meaningfully stylized according to the target image. All of the results shown in Figure 11 were trained for approximately 18 hours on a NVIDIA Tesla V100-SXM3. While these GPUs have sufficient memory to utilize a batch size of up to 10, a batch size of 1 lead to faster convergence and more stable training. Visible results are already achieved after 2-4 hours of training, as shown in Figure 7. The main advantage of increasing the training time is the creation of sharper features, which can be especially important for high-frequency style images. For the high-resolution example (Figure 1), we used patch-sizes of 256^3.

For most of the styles we sampled random orientations over the full range of possible rotations, i.e. rotations around the x-axis in the range of $\theta \in [-180°, 180°]$ and rotations around the y-axis in the range of $\phi \in [-90°, 90°]$, except for the *fire* and the *foam* styles. Those styles can not be achieved from all possible views, hence we limited the y-axis rotation to a range of $\phi \in [-20°, 20°]$. Two octaves were used during all view-independent examples to produce features at different scales. Input densities are normalized to a range of $[0, 1]$ before being fed into the Feed-Forward network. We adopt a OneCycle learning rate scheduler [Smith and Topin 2017] with initial and maximal learning rates of 5×10^{-6} and $1 \times 10{-}4$.

5.2 Neural style transfer results

Single-view Examples. Figure 10 displays various styles applied to the *billowy smoke* dataset, with all examples employing the velocity-based stylization (Equation (1)). For image-based stylizations, we represented the input image style by computing the Gram matrices of specified VGG-19 layers. A one-to-one matching with previous implementations is not entirely possible, primarily because the network weights from the classification networks used differ on their TensorFlow and PyTorch versions. The running time and parameters used for these, along with other examples in the paper are shown in Table (1). We highlight that the original TNST [Kim et al. 2019a] (Table 2) took up to 13 minutes for a single frame, while our improved method takes less than 12 seconds per frame for stylizing the *smokejet* dataset (Figure 2).

View-independent results. The inference of the proposed compact feed-forward architecture is very efficient: 120 frames of the *billowy smoke* sequence are stylized in only 2 minutes, including data I/O. The neural network evaluation takes less memory than training, so all feed-forward inferences were performed in a NVIDIA RTX 2080 TI with 12GB memory. Figure 11 shows stylizations computed for the respective inset images. We notice that stylizing in a view-independent setting sometimes yields less sharp results. Particularly, the starry stylization (second row) exhibits larger patterns when compared to the single-view setting (Figure 10), even though this example used a reduced rotation range. Currently, our pipeline still involves a trial-and-error process that requires the artist to test parameters such as the camera sampling patterns, neural network layers and rendering transmittance. Fortunately, the optimization pipeline is efficient enough to iterate over these stylization parameters - the whole process takes under 5 minutes for a single-frame of the *billowy smoke*. This is the process we used to find optimal parameters for the dark-matter example.

High-resolution view-independent stylization. The high-resolution result shown in Figure 1 does not entirely fit on the GPU for inference. Therefore the volume is partitioned into tiles of size 256^3 that are stylized individually. To produce spatially consistent results, the tiles overlap with each other with a margin of 50 voxels and only the smaller non-overlapping portion of the patches are kept. Since our network is convolutional and continuous, this method produces a seamless stitching of the stylized result. Tiling greatly increases inference time, as I/O overhead is added: the full volume is stored on the CPU and each tile has to be transferred to the GPU as the feed-forward network is evaluated. Asynchronous transfers are employed to allow the overlap of computation and communication. This improves inference time but is not able to completely alleviate the I/O bottleneck. The advantage of tiling is the new possibility of stylizing large-resolution volumes on consumer grade GPUs. The full stylization of 180 frames took 3 hours using a NVIDIA RTX 2080 TI. Due to the memory requirements of the direct optimization it would not be feasible to stylize the whole sequence in a temporal coherent manner. Having a GPU with enough memory to simultaneously fit one frame of the full volume and the trained model onto its dedicated memory would greatly improve the inference time.

6 CONCLUSIONS AND FUTURE WORK

In this paper we presented an efficient volumetric Neural Style Transfer for smoke simulations. Our method is faster, simpler and easier to integrate into production pipelines than previous works. Our central contribution is to allow temporally coherent volumetric stylizations without the need of an expensive particle-to-grid conversion step. Moreover, we propose a simple and lightweight architecture that is able to produce view-independent stylizations for high-resolution datasets.

Our method, however, is not without its limitations. Firstly, the generalization capabilities of the proposed view-independent feed-forward network were not thoroughly tested for widely different smoke datasets. An ablation study is needed in order to properly evaluate the trade-off between network size and generalization. We experimentally observed that the proposed feed-forward approach has a bounded output: if the input was never seen, the output of the network at least preserves the original content. Moreover, we did not explicitly included a term for ensuring temporal coherency in the feed-forward architecture. Our method might benefit from such treatment, since only relying on the translational equivariance of the learned convolutional filters might not be enough to enforce temporal coherence.

There are further directions that can be explored for future work. Understanding which type of images can be used for omniview stylizations is an area that needs more exploring. For the results we presented, the dark matter exemplar produced stylizations that were able to be extended to 3D consistently. Other stylization images (e.g., fire or spirals) shown in Figure 11 exhibit discrepancy between

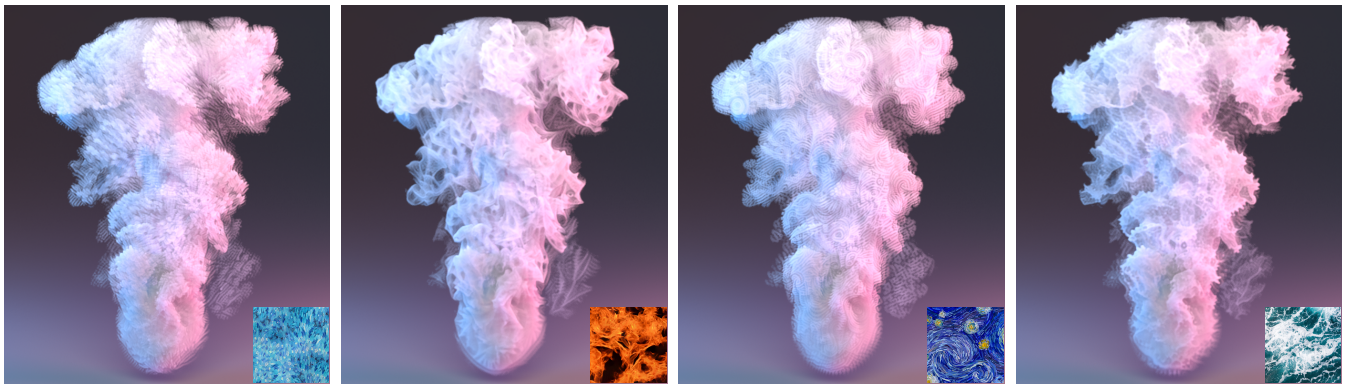


Fig. 10. **Single-view stylization for various styles.** These results demonstrate that our method still produces similar stylizations as previous approaches, but with improved computational efficiency. Running times for these examples are shown in Table 1.

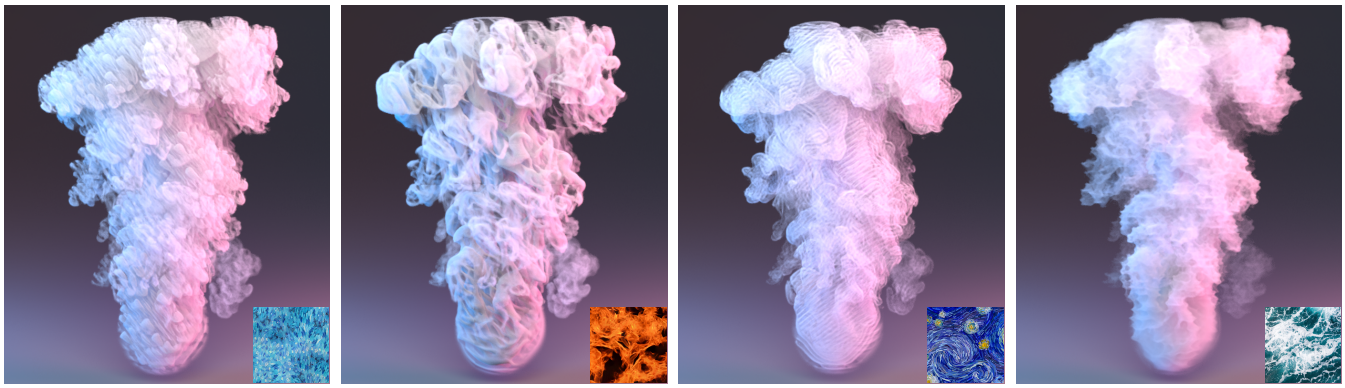


Fig. 11. **Feed-forward neural network results for different styles.** Images display results that are all trained for the *billowy smoke* sequence.

Table 1. Performance table (Timings measured on a NVIDIA RTX 2080 TI). The time needed for generating particles of the *billowy smoke* scene (Fig. 4) was 32 minutes.

Scene	# Frames	Resolution	# Octaves	Method	Iterations	Learning rate	Time (per figure panel)
High Resolution (Fig. 1)	180	$645 \times 609 \times 1553$	2	Feed-Forward	-	-	181m
Smoke Jet (Fig. 2)	120	$200 \times 300 \times 200$	3	velocity	20	2	21m/30m/31m
Billowy Smoke (Fig. 3)	120	$350 \times 500 \times 350$	3	velocity	20	0.2	43m each
Billowy Smoke (Fig. 4)	90	$175 \times 250 \times 175$	3	density	20	0.2	12m (TNST), 9m (LNST)
Billowy Smoke (Fig. 10)	120	$350 \times 500 \times 350$	2-3	velocity	20	2	45m (average)
Billowy Smoke (Fig. 11)	120	$350 \times 500 \times 350$	2-3	Feed-Forward	-	-	2m (average)
Smoke Jet (Fig. 8)	120	$200 \times 300 \times 200$	2	Feed-Forward	-	-	48s
Bunny (Fig. 8)	120	$200 \times 300 \times 200$	2	Feed-Forward	-	-	53s

features that are generated in single and the omniview setups. We believe that our work will further spark the interest in artistically stylizing volumetric simulations with 2D images.

REFERENCES

Kai Bai, Wei Li, Mathieu Desbrun, and Xiaopei Liu. 2019. Dynamic Upsampling of Smoke through Dictionary-based Learning. (oct 2019). arXiv:1910.09166 http://arxiv.org/abs/1910.09166

Kai Bai, Chunhao Wang, Mathieu Desbrun, and Xiaopei Liu. 2021. Predicting high-resolution turbulence details in space and time. *ACM Transactions on Graphics* 40, 6 (dec 2021), 1–16. https://doi.org/10.1145/3478513.3480492

Mengyu Chu and Nils Thuerey. 2017. Data-driven synthesis of smoke flows with CNN-based feature descriptors. *ACM Transactions on Graphics* 36, 4 (jul 2017), 1–14. https://doi.org/10.1145/3072959.3073643

Mengyu Chu, Nils Thuerey, Hans-Peter Seidel, Christian Theobalt, and Rhaleb Zayer. 2021. Learning meaningful controls for fluids. *ACM Transactions on Graphics* 40, 4 (aug 2021), 1–13. https://doi.org/10.1145/3450626.3459845

Graham Collier. 2022. Raya and the Last Dragon. https://www.sidefx.com/community/raya-and-the-last-dragon/

Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. 2017. A Learned Representation For Artistic Style. *ICLR* (2017). https://arxiv.org/abs/1610.07629

Julian Fong, Magnus Wrenninge, Christopher Kulla, and Ralf Habel. 2017. Production volume rendering. In *ACM SIGGRAPH 2017 Courses on - SIGGRAPH '17*. ACM Press, New York, New York, USA, 1–79. https://doi.org/10.1145/3084873.3084907

Erik Franz, Barbara Solenthaler, and Nils Thuerey. 2021. Global Transport for Fluid Reconstruction with Learned Self-Supervision. (apr 2021). arXiv:2104.06031 http://arxiv.org/abs/2104.06031

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. 2016. Image Style Transfer Using Convolutional Neural Networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2414–2423. https://doi.org/10.1109/CVPR.2016.265

Jie Guo, Mengtian Li, Zijing Zong, Yuntao Liu, Jingwu He, Yanwen Guo, and Ling Qi Yan. 2021. Volumetric appearance stylization with stylizing kernel prediction network. *ACM Transactions on Graphics (TOG)* 40, 4 (jul 2021). https://doi.org/10.1145/3450626.3459799

Jun Han and Chaoli Wang. 2022. TSR-VFD: Generating temporal super-resolution for unsteady vector field data. *Computers Graphics* 103 (apr 2022), 168–179. https://doi.org/10.1016/j.cag.2022.02.001

Philipp Holl, Nils Thuerey, and Vladlen Koltun. 2020. Learning to Control PDEs with Differentiable Physics. In *International Conference on Learning Representations*.

Yuanming Hu, Jiancheng Liu, Andrew Spielberg, Joshua B. Tenenbaum, William T. Freeman, Jiajun Wu, Daniela Rus, and Wojciech Matusik. 2018. ChainQueen: A Real-Time Differentiable Physical Simulator for Soft Robotics. (oct 2018). arXiv:1810.01054 http://arxiv.org/abs/1810.01054

Yuanming Hu, Xinxin Zhang, Ming Gao, and Chenfanfu Jiang. 2019. On hybrid lagrangian-eulerian simulation methods: practical notes and high-performance aspects. In *ACM SIGGRAPH 2019 Courses*. ACM, 16.

Yongcheng Jing, Yang Liu, Yezhou Yang, Zunlei Feng, Yizhou Yu, Dacheng Tao, and Mingli Song. 2018. Stroke Controllable Fast Style Transfer with Adaptive Receptive Fields. 244–260. https://doi.org/10.1007/978-3-030-01261-8_15

Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu, and Mingli Song. 2019. Neural Style Transfer: A Review. *IEEE Transactions on Visualization and Computer Graphics* (2019), 1–1. https://doi.org/10.1109/TVCG.2019.2921336

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*.

Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. 2018. Neural 3d mesh renderer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3907–3916.

Byungsoo Kim, Vinicius C. Azevedo, Markus Gross, and Barbara Solenthaler. 2019a. Transport-based neural style transfer for smoke simulations. *ACM Transactions on Graphics* 38, 6 (dec 2019), 1–11. https://doi.org/10.1145/3355089.3356560

Byungsoo Kim, Vinicius C. Azevedo, Markus Gross, and Barbara Solenthaler. 2020. Lagrangian neural style transfer for fluids. *ACM Transactions on Graphics* 39, 4 (aug 2020). https://doi.org/10.1145/3386569.3392473

Byungsoo Kim, Vinicius C. Azevedo, Nils Thuerey, Theodore Kim, Markus Gross, and Barbara Solenthaler. 2019b. Deep Fluids: A Generative Network for Parameterized Fluid Simulations. *Computer Graphics Forum (Proc. Eurographics)* 38, 2 (2019).

Byungsoo Kim, Xingchang Huang, Laura Wuelfroth, Jingwei Tang, Guillaume Cordonnier, Markus Gross, and Barbara Solenthaler. 2022. Deep Reconstruction of 3D Smoke Densities from Artist Sketches. *Computer Graphics Forum (Proc. Eurographics)* 41, 2 (2022).

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. https://doi.org/10.48550/ARXIV.1412.6980

M. Kohlbrenner, U. Finnendahl, T. Djuren, and M. Alexa. 2021. Gauss Stylization: Interactive Artistic Mesh Modeling based on Preferred Surface Normals. *Computer Graphics Forum* 40, 5 (aug 2021), 33–43. https://doi.org/10.1111/cgf.14355

L'ubor Ladický, SoHyeon Jeong, Barbara Solenthaler, Marc Pollefeys, and Markus Gross. 2015. Data-driven fluid simulations using regression forests. *ACM Transactions on Graphics* 34, 6 (oct 2015), 1–9. https://doi.org/10.1145/2816795.2818129

Shaohua Li, Xinxing Xu, Liqiang Nie, and Tat-Seng Chua. 2017b. Laplacian-Steered Neural Style Transfer. In *Proceedings of the 2017 ACM on Multimedia Conference - MM '17*. ACM Press, New York, New York, USA, 1716–1724. https://doi.org/10.1145/3123266.3123425

Xueting Li, Sifei Liu, Jan Kautz, and Ming-Hsuan Yang. 2019. Learning Linear Transformations for Fast Image and Video Style Transfer. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Yanghao Li, Naiyan Wang, Jiaying Liu, and Xiaodi Hou. 2017a. Demystifying Neural Style Transfer. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*. AAAI Press, 2230–2236.

Zijie Li and Amir Barati Farimani. 2022. Graph neural network-accelerated Lagrangian fluid simulation. *Computers Graphics* 103 (apr 2022), 201–211. https://doi.org/10.1016/j.cag.2022.02.004

Hsueh-Ti Derek Liu and Alec Jacobson. 2019. Cubic Stylization. (oct 2019). https://doi.org/10.1145/3355089.3356495 arXiv:1910.02926

Hsueh-Ti Derek Liu, Michael Tao, and Alec Jacobson. 2018. Paparazzi: Surface Editing by way of Multi-View Image Processing. *ACM Transactions on Graphics* (2018).

Fujun Luan, Sylvain Paris, Eli Shechtman, and Kavita Bala. 2017. Deep Photo Style Transfer. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 6997–7005. https://doi.org/10.1109/CVPR.2017.740

Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. 2021. Text2Mesh: Text-Driven Neural Stylization for Meshes. (dec 2021). arXiv:2112.03221 http://arxiv.org/abs/2112.03221

Mike Navarro and Jacob Rice. 2021. Stylizing Volumes with Neural Networks. In *ACM SIGGRAPH 2021 Talks*. ACM, New York, NY, USA, 1–2. https://doi.org/10.1145/3450623.3464652

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. https://doi.org/10.48550/ARXIV.1912.01703

Eric Risser, Pierre Wilmot, and Connelly Barnes. 2017. Stable and Controllable Neural Texture Synthesis and Style Transfer Using Histogram Losses. (jan 2017). arXiv:1701.08893 http://arxiv.org/abs/1701.08893

Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. 2018. Artistic Style Transfer for Videos and Spherical Images. *International Journal of Computer Vision* 126, 11 (nov 2018), 1199–1219. https://doi.org/10.1007/s11263-018-1089-z

Connor Schenck and Dieter Fox. 2018. SPNets: Differentiable Fluid Dynamics for Deep Neural Networks. (jun 2018). arXiv:1806.06094 http://arxiv.org/abs/1806.06094

Ahmed Selim, Mohamed Elgharib, and Linda Doyle. 2016. Painting style transfer for head portraits using convolutional neural networks. *ACM Transactions on Graphics* 35, 4 (jul 2016), 1–18. https://doi.org/10.1145/2897824.2925968

Andrew Selle, Ronald Fedkiw, ByungMoon Kim, Yingjie Liu, and Jarek Rossignac. 2008. An Unconditionally Stable MacCormack Method. *Journal of Scientific Computing* 35, 2-3 (jun 2008), 350–371. https://doi.org/10.1007/s10915-007-9166-4

Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. (sep 2014). arXiv:1409.1556 http://arxiv.org/abs/1409.1556

Leslie N. Smith and Nicholay Topin. 2017. Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates. https://doi.org/10.48550/ARXIV.1708.07120

Steven W Smith. 1997. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, USA.

Jingwei Tang, Vinicius C. Azevedo, Guillaume Cordonnier, and Barbara Solenthaler. 2021. Honey, I Shrunk the Domain: Frequency-aware Force Field Reduction for Efficient Fluids Optimization. *Computer Graphics Forum* 40, 2 (may 2021), 339–353. https://doi.org/10.1111/cgf.142637

Nils Thuerey, Philipp Holl, Maximilian Mueller, Patrick Schnell, Felix Trost, and Kiwon Um. 2021. Physics-based Deep Learning. (sep 2021). arXiv:2109.05237 http://arxiv.org/abs/2109.05237

Nils Thuerey and Tobias Pfaff. 2018. MantaFlow. *http://mantaflow.com*.

Jonathan Tompson, Kristofer Schlachter, Pablo Sprechmann, and Ken Perlin. 2016. Accelerating Eulerian Fluid Simulation With Convolutional Networks. (jul 2016). arXiv:1607.03597 http://arxiv.org/abs/1607.03597

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. 2016. Instance Normalization: The Missing Ingredient for Fast Stylization. (jul 2016). arXiv:1607.08022 http://arxiv.org/abs/1607.08022

Kiwon Um, Robert Brand, Yun, Fei, Philipp Holl, and Nils Thuerey. 2020. Solver-in-the-Loop: Learning from Differentiable Physics to Interact with Iterative PDE-Solvers. (jun 2020). arXiv:2007.00016 http://arxiv.org/abs/2007.00016

Nobuyuki Umetani and Bernd Bickel. 2018. Learning three-dimensional flow for interactive aerodynamic design. *ACM Transactions on Graphics* 37, 4 (jul 2018), 1–10. https://doi.org/10.1145/3197517.3201325

Xin Wang, Geoffrey Oxholm, Da Zhang, and Yuan-Fang Wang. 2016. Multimodal Transfer: A Hierarchical Deep Convolutional Neural Network for Fast Artistic Style Transfer. (nov 2016). arXiv:1612.01895 http://arxiv.org/abs/1612.01895

You Xie, Erik Franz, Mengyu Chu, and Nils Thuerey. 2018. tempoGAN: A Temporally Coherent, Volumetric GAN for Super-resolution Fluid Flow. *arXiv preprint arXiv:1801.09710* (jan 2018). arXiv:1801.09710 http://arxiv.org/abs/1801.09710

Guowei Yan, Zhili Chen, Jimei Yang, and Huamin Wang. 2020. Interactive liquid splash modeling by user sketches. *ACM Transactions on Graphics* 39, 6 (dec 2020), 1–13. https://doi.org/10.1145/3414685.3417832

Cheng Yang, Xubo Yang, and Xiangyun Xiao. 2016. Data-driven projection method in fluid simulation. *Computer Animation and Virtual Worlds* 27, 3-4 (may 2016), 415–424. https://doi.org/10.1002/cav.1695

Kangxue Yin, Jun Gao, Maria Shugrina, Sameh Khamis, and Sanja Fidler. 2021. 3DStyleNet: Creating 3D Shapes with Geometric and Texture Style Variations. (aug 2021). arXiv:2108.12958 http://arxiv.org/abs/2108.12958

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2242–2251. https://doi.org/10.1109/ICCV.2017.244