Artistically controlling the shape, motion and appearance of fluid simulations pose major challenges in visual effects production. In this paper, we present a neural style transfer approach from images to 3D fluids formulated in a Lagrangian viewpoint. Using particles for style transfer has unique benefits compared to grid-based techniques. Attributes are stored on the particles and hence are trivially transported by the particle motion. This intrinsically ensures temporal consistency of the optimized stylized structure and notably improves the resulting quality. Simultaneously, the expensive, recursive alignment of stylization velocity fields of grid approaches is unnecessary, reducing the computation time to less than an hour and rendering neural flow stylization practical in production settings. Moreover, the Lagrangian representation improves artistic control as it allows for multi-fluid stylization and consistent color transfer from images, and the generality of the method enables stylization of smoke and liquids likewise.


Additional Key Words and Phrases: physically-based animation, fluid simulation, deep learning, neural style transfer

ACM Reference Format:

1 INTRODUCTION

In visual effects production, physics-based simulations are not only used to realistically re-create natural phenomena, but also as a tool to convey stories and trigger emotions. Hence, artistically controlling the shape, motion and the appearance of simulations is essential for providing directability for physics. Specifically to fluids, the major challenge is the non-linearity of the underlying fluid motion equations, which makes optimizations towards a desired target difficult. Keyframe matching either through expensive fully-optimized simulations [McNamara et al. 2004; Pan and Manocha 2017; Treuille et al. 2003] or simpler distance-based forces [Nielsen and Bridson 2011; Raveendran et al. 2012] provide control over the shape of fluids. The fluid motion can be enhanced with turbulence synthesis approaches [Kim et al. 2008; Sato et al. 2018] or guided by coarse grid simulations [Nielsen and Bridson 2011], while patch-based texture composition [Gagnon et al. 2019; Jamriška et al. 2015] enables manipulation over appearance by automatic transfer of input 2D image patterns.

The recently introduced Transport-based Neural Style Transfer (TNST) [Kim et al. 2019a] takes flow appearance and motion control to a new level: arbitrary styles and semantic structures given by 2D input images are automatically transferred to 3D smoke simulations. The achieved effects range from natural turbulent structures
to complex artistic patterns and intricate motifs. The method extends traditional image-based Neural Style Transfer [Gatys et al. 2016] by reformulating it as a transport-based optimization. Thus, TNST is physically inspired, as it computes the density transport from a source input smoke to a desired target configuration, allowing control over the amount of dissipated smoke during the stylization process. However, TNST faces challenges when dealing with time coherency due to its grid-based discretization. The velocity field computed for the stylization is performed independently for each time step, and the individually computed velocities are recursively aligned for a given window size. Large window sizes are required, rendering the recursive computation expensive while still accumulating inaccuracies in the alignment that can manifest as discontinuities. Moreover, transport-based style transfer is only able to advect density values that are present in the original simulation, and therefore it does not inherently support color information or stylizations that undergo heavy structural changes.

Thus, in this work, we reformulate Neural Style Transfer in a Lagrangian setting (see Figure 2), demonstrating its superior properties compared to its Eulerian counterpart. In our Lagrangian formulation, we optimize per-particle attributes such as positions, densities and color. This intrinsically ensures better temporal consistency as shown for example in Figure 3, eliminating the need for the expensive recursive alignment of stylization velocity fields. The Lagrangian approach reduces the computational cost to enforce time coherency, increasing the speed of results from one day to a single hour. The Lagrangian Style transfer framework is completely oblivious to the underlying fluid solver type. Since the loss function is based on filter activations from pre-trained classification networks, we transfer the information back and forth from particles to the grids, where loss functions and attributes can be jointly updated. We propose regularization strategies that help to conserve the mass of the underlying simulations, avoiding oversampling of stylization particles. Our results demonstrate novel artistic manipulations, such as stylization of liquids, color stylization, stylization of multiple fluids, and time-varying stylization.

2 RELATED WORK

Lagrangian Fluids have become popular for simulating incompressible fluids and interactions with various materials. Since the introduction of SPH to computer graphics [Desbrun and Gascuel 1996; Müller et al. 2003], various extensions have been presented that made it possible to efficiently simulate millions of particles on a single desktop computer. Accordingly, particle methods reached an unprecedented level of visual quality, where fine-scale surface effects and flow details are reliably captured. To enforce incompressibility, the original state equation based method [Becker and Teschner 2007; Monaghan 2005] has been replaced by pressure Poisson equation (PPE) solvers using either a single source term for density invariance [Ilmsen et al. 2014; Solenthaler and Pajarola 2009] or two PPEs to additionally account for divergence-free velocities [Bender and Koschier 2015]. Solvers closely related to PPE have been presented, such as Local Poisson SPH [He et al. 2012], Constraint Fluids [Servin...
et al. 2012] and Position-based Fluids [Macklin and Mueller 2013]. Boundary handling is computed with particle-based approaches that sample boundary geometry (e.g. [Gissler et al. 2019]) or implicit methods that typically use a signed distance field (e.g. [Koschier and Bender 2017]). Extensions include highly viscous fluids (e.g. [Peer et al. 2015]), and multiple phases and fluid mixing (e.g. [Ren et al. 2014]). An overview of recent developments in SPH can be found in the course notes of Koschier et al. [2019].

**Hybrid Lagrangian-Eulerian Fluids** combine the versatility of the particles representation to track transported quantities with the capacity of grids to enforce incompressibility. Among popular approaches, the Fluid Implicit Particle Method (FLIP) [Brachet et al. 1988] was first employed in graphics to animate sand and water [Zhu and Bridson 2005]. Due to its ability to accurately capture sub-grid details it has been widely adopted for liquid simulations, being extended to animation of turbulent water [Kim et al. 2006], coupled with SPH for modelling small scale splashes [Losasso et al. 2008], improved for efficiency [Ando et al. 2013; Ferstl et al. 2016], used in fluid control [Pan et al. 2013], and enhanced with better particle distribution [Ando and Tsuruno 2011; Um et al. 2014]. The Material Point Method (MPM) [Stomakhin et al. 2013] was used to simulate a wide class of solid materials [Jiang et al. 2016]. Recent work on hybrid approaches extended the information tracked by the particles by affine [Jiang et al. 2015] and polynomial [Fu et al. 2017] transformations. For a thorough discussion of hybrid continuum models, we refer to Hu et al. [2019b].

**Patch-based Appearance Transfer** methods compute similarities between source and target datasets in local neighborhoods, modifying the appearance of the source by transferring best-matched features from the target dataset. Kwatra et al. [2005] employ local similarity measures in an energy-based optimization, enabling texture patches animated by flow fields. This approach was further extended to liquid surfaces [Bargteil et al. 2006; Kwatra et al. 2006], and improved by modifying the texture based on visually salient features of the liquid mesh [Narain et al. 2007]. Jamriška et al. [2015] improved previous work with better temporal coherency and matching precision for obtaining high-quality 2D textured fluids. Texturing liquid simulations was also implemented in a Lagrangian framework by using individually tracked surface patches [Gagnon et al. 2016, 2019; Yu et al. 2011]. Image and video-based approaches also take inspiration from fluid transport. Bousseau et al. [2007] proposed a bidirectional advection scheme to reduce patch distortions. Regenerative morphing and image melding techniques were combined with patch-based tracking to produce in-betweenes for artist-stylized keyframes [Browning et al. 2014]. Recent advances in patch-based appearance transfer often rely on evaluating the underlying 3D geometric information; examples include improving template matching by a novel similarity measure [Talmi et al. 2017], patch matching for illumination effects [Fiser et al. 2016], extensions to texture mapping [Bi et al. 2017] and intricate texture motifs [Diamanti et al. 2015]. While these approaches were successful in 2D settings and for texturing liquids, they cannot inherently support 3D volumetric data.

**Velocity Synthesis** methods augment flow simulations with velocity fields, which manipulate or enhance volumetric data. Due to the inability of pressure-velocity style transfer to properly conserve different energy scales of flow phenomena, sub-grid turbulence [Kim et al. 2008; Narain et al. 2008; Schechter and Bridson 2008] was modelled for better energy conservation. These approaches were extended to model turbulence in the wake of solid boundaries [Haff et al. 2009], liquid surfaces [Kim et al. 2013] and example-based turbulence synthesis [Sato et al. 2018]. In order to merge fluids of different simulation instances [Thuerey 2016] or separated by void regions [Sato et al. 2018], velocity fields where synthesized by solving an unconstrained energy minimization problem. Lastly, the Transport-based Neural Style Transfer (TNST) [Kim et al. 2019a] can also be seen as a velocity synthesis method: at each time-step, the method optimizes a velocity field that transports the smoke towards a desired stylization.

**Machine Learning & Fluids** was first introduced to graphics by Ladický et al. [2015]. They used Regression Forests to predict positions of fluid particles over time, resulting in a substantial performance gain compared to traditional Lagrangian solvers. CNN-based architectures were employed in Eulerian-based solvers to substitute the pressure projection step [Tompson et al. 2017; Yang et al. 2016] and to synthesize flow simulations from a set of reduced parameters [Kim et al. 2019b]. An LSTM architecture [Wiewel et al. 2019] predicted changes on pressure fields for multiple subsequent time-steps, speeding up the pressure projection step. Differentiable fluid solvers [Holl et al. 2020; Hu et al. 2020, 2019a; Schenck and Fox 2018] have been introduced that can be automatically coupled with deep learning architectures and provide a natural interface for image-based applications. Patch-based [Chu and Thuerey 2017] and GAN-based [Xie et al. 2018] fluid super-resolution enhance coarse simulations with rich turbulence details, while also being computationally inexpensive. While these approaches produce detailed, high-quality results, they do not support transfer of arbitrary smoke styles.

**Differentiable Rendering and Stylization** is used in Neural Style Transfer algorithms to transfer the style of a source image to a target image by matching features of a pre-trained classified network [Gatys et al. 2016]. However, stylizing 3D data requires a differentiable renderer to map the representation to image space. Loper and Black [2014] proposed the first fully differentiable renderer with automatically computed derivatives, while a novel differentiable volume sampling was implemented by Yan et al. [2016]. Raster-based differentiable rendering for meshes for stylization with approximate [Kato et al. 2018] and analytic [Liu et al. 2018] derivatives was proposed to approximate visibility changes and mesh filters, respectively. A cubic stylization algorithm [Liu and Jacobson 2019] was implemented by minimizing a constrained energy formulation and employed to mesh stylization. Closer to our work, Kim et al. [2019a] defines an Eulerian framework for a transport-based neural style transfer of smoke. Their approach computes individually stylized velocity fields per-frame, and temporal coherence is enforced by aligning subsequent stylization velocity fields and performing smoothing. We compare the Eulerian approach with our method in the subsequent sections. For an overview on differentiable rendering and neural style transfer we refer to Yifan et al. [2019] and Jing et al. [2019], respectively.
3 EULERIAN TRANSPORT-BASED NST

We briefly review previous Eulerian-based TNST [Kim et al. 2019a] for completeness and to better compare against our novel Lagrangian approach. Transport-Based Neural Style Transfer (TNST) extends the original NST algorithm to transfer the style of a given image to a flow-based 3D smoke density. As opposed to NST where individual pixel sizes of the target image are optimized, TNST optimizes a velocity field that modifies density values through indirect smoke transport. The velocity field $\hat{v}$ that stylizes the input density $d$ is defined by a loss function $\mathcal{L}$ computed from a pre-trained image classification CNN by

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

where $T$ is a transport function that advects $d$ with $v$, generating the stylized density $\hat{d} = T(d, \hat{v}); \mathcal{R}$ is a differentiable renderer converting the density field to image-space for a specific view $\theta$ by $l = \mathcal{R}_\theta(\hat{d})$, and $p$ denotes the set of user-defined parameters used in the stylization process. The velocity field contributions are individually computed per view, resulting in a 3D volumetric smoke stylization. While the authors separate the velocity field into its 1-D feature map $F$, the Gram matrix is calculated for a given layer $l$:

$$\hat{F}^l(I) = \sum_{i=0}^{L} \frac{1}{4C^2_l(H_l \times W_l)^2} \sum_{m,n} \left( G^l_{mn}(I) - G^l_{mn}(I_s) \right)^2,$$

where the Gram matrix $G$ computes correlations between different filter responses. The Gram matrix is calculated for a given layer $l$ and two channels $m$ and $n$, by iterating over all pixels of the flattened 1-D feature map $\hat{F}^l(I)$ as

$$G^l_{mn}(I) = \sum_{i=0}^{L} \hat{F}^l_{mi}(I) \hat{F}^l_{ni}(I).$$

Extending the single frame stylization in a time-coherent fashion is expensive and inaccurate when computed in an Eulerian framework. TNST aligns stylization velocities by recursively advecting them with the simulation velocities for a given window size as shown in Figure 4. The recursive nature renders this computation inefficient time- and memory-wise, especially when large window sizes are employed to enable smooth transitions between consecutive frames. Due to the large memory requirement, this operation often has to be computed on the CPU, which generates additional overhead by the use of expensive data transfer operations.

4 LAGRANGIAN NST

In contrast to its Eulerian counterpart, the Lagrangian representation uses particles that carry quantities such as the position, density and color value. Neural style transfer methods compute loss functions based on filter activations from pre-trained classification networks, which are trained on image datasets. Thus, we have to transfer the information back and forth from particles to the grids, where loss functions and attributes can be jointly updated. We take inspiration from hybrid Lagrangian-Eulerian fluid simulation pipelines that use grid-to-particle $I_{g\rightarrow p}$ and particle-to-grid $I_{p\rightarrow g}$ transfers as

$$\lambda^o = I_{g\rightarrow p}(x^o, \lambda^o) \text{ and } \lambda^g = I_{p\rightarrow g}(x^p, \lambda^p, h, x^g),$$

where $\lambda^o$ and $\lambda^g$ are attributes defined on the particle and grid, respectively, $x^o$ refers to all particle positions, $x^g$ are grid nodes to which values are transferred, and $h$ is the support size of the particle-to-grid transfer.

Our grid-to-particle transfer employs a regular grid cubic interpolant, while the particle-to-grid transfer uses standard radial basis functions. Regular Cartesian grids facilitate finding grid vertices around an arbitrary particle position. For this, we extended a differentiable point cloud projector [Insafutdinov and Dosovitskiy 2018] to arbitrary grid resolution, neighborhood size and custom kernel functions. Given all the neighboring particles $j \in \partial \Omega_h$ around a grid node $x$, a grid attribute $\lambda^g$ is computed by summing up weighted particle contributions as

$$\lambda^g(x) = \sum_{j \in \partial \Omega_h} \lambda^o_j W(||x - x^o_j||, h) \sum_{j \in \partial \Omega_h} W(||x - x^g_j||, h),$$

where we chose $W$ to be the cubic B-spline kernel, which is also often used in SPH simulations [Monaghan 2005]:

$$W(r, h)_{\text{cubic}} = \begin{cases} \frac{1}{6} - r^2 + \frac{1}{2}r^3, & 0 \leq r \leq 1, \\ \frac{1}{6}(2-r)^3, & 1 \leq r \leq 2, \\ 0, & r > 2. \end{cases}$$

We now have all the necessary elements to convert the previous Eulerian style transfer (Equation (1)) into a Lagrangian framework. Given a set of Lagrangian attributes $\lambda^o$, the optimization objective for a single frame is

$$\hat{\lambda}^o = \arg\min_{\lambda^o} \sum_{\theta \in \Theta} \sum_{\lambda^p \in \Lambda^p} w^{\lambda_p}_{\lambda^o} \mathcal{L}(\mathcal{R}_\theta(I_{g\rightarrow p}(x^o, \lambda^p)), p),$$

where $w^{\lambda_p}_{\lambda^o}$ are weights for the losses that include Lagrangian attributes. In case of particle position $x^o$ given as the target quantity $\lambda^o$, we use the SPH density $I_{g\rightarrow p}(x^o) = \sum_{j \in \partial \Omega_h} m_j W(||x - x^o_j||, h)$, where $m_j$ represents the mass of the $j$-th particle [Bender 2016]. Note that our losses are evaluated similarly as in the original Eulerian
method, since the gradients computed in image-space also modify grid values ($\lambda^0$). However, these gradients are automatically propagated back to the particles by auto-differentiating the particle-to-grid $I_{P2g}$ function. Thus, our method only reformulates the domain of the optimization, sharing the same stylization possibilities (semantic and content transfers) as in the original TNST.

Since the Lagrangian optimization is completely oblivious to the underlying solver type, the chosen attributes for creating stylizations can be arbitrarily combined, enabling a wide range of artistic manipulations in different scene setups. We outline two strategies and demonstrate their impact on the stylization. The first one is particularly suitable for participating volumetric data, which are often simulated with grid-based solvers. It involves optimizing a scalar value carried by the Lagrangian stylization particles by Equation (7). For most of our smoke scenes, this scalar value is the density, though it can also be the color or emission. The regularization term

$$L(\lambda^p) = \left( \sum \Delta \rho^c \right)^2 - \sum \log ||\Delta \lambda^0||_1$$

(8)

reinforces the conservation of the original amount of smoke. It minimizes the total net smoke change, preventing the stylization to undesirably fade out particles and keeping changes non-zero by minimizing cross-entropy loss at the same time. Figure 5 demonstrates the impact of different regularizer weights.

Fig. 5. Different weights for the density regularization show the trade-off between pronounced structures and conservation of mass. The images on the left show results with zero, low, and high weights, respectively, and the right image is the ground truth.

The second strategy is suitable if the underlying fluid solver is particle-based or hybrid, which is often the case for liquids. For these simulations, we can define particle positions as the optimized Lagrangian attributes. However, generating stylizations by modifying particle displacements may cause cluttering or regions with insufficient particles. The regularization penalizes irregular distribution of particle positions and is defined as

$$L(x^\lambda)_{Ax} = ||I_{P2g}(x^\lambda) - \rho_0^Ax||^2_2$$

(9)

where $\rho_0^Ax$ corresponds to the rest density for cells that contain particles, and is zero otherwise. Note that Equation (9) does not account for the particle deficiency near fluid surfaces. This could be addressed by adding virtual particles [Schechter and Bridson 2008] or applying (variants of) the Shepard correction to the kernel function [Reinhardt et al. 2019]. We show the impact of this regularizer on the particle sampling in Figure 6, highlighting the trade-off between uniform distribution and stylization strength.

We notice that both regularizations in Equation (8) and Equation (9) are different incarnations of the mass conservation property commonly used in fluid simulations. In TNST, mass conservation is enforced by decomposing the stylization velocities into their irrotational and incompressible parts, which can be optimized independently. Both techniques enable a high degree of artistic control over the content manipulation.

4.1 An Efficient Particle-Based Smoke Re-Simulation

If the input is a grid-based simulation, we have to sample and re-simulate particles. We can use a sparse representation with only one particle per voxel, in contrast to hybrid liquid simulations that usually sample 8 particles per voxel to properly capture momentum conservation [Zhu and Bridson 2005]. Combining a low number of particles with a position integration algorithm that accumulates errors over time will yield irregularly distributed particles [Ando and Tsuruno 2011]. This manifests in a rendered image as smoke with overly dense or void regions. We therefore solve the following optimization problem

$$\hat{x}^\rho, \hat{\rho}^o = \arg \min_{x^\lambda, \rho^0} \sum \frac{||I_{P2g}(x^\lambda, \rho^0) - \rho_0^Ax||^2_2}{2}.$$  

(10)

The optimization problem presented above is not only severely under-constrained but also has a time-varying objective term, and optimizing for Equation (10) is challenging if tackled jointly for both particle positions $x^\lambda$ and densities $\rho^0$. Thus, we use a heuristic approach for solving this optimization, subdividing it into two steps, position optimization and multi-scale density update (Section 4.1.1). Firstly, we minimize the irregular distribution of particle positions by employing a position-based update, optimizing particle distributions using Equation (9) as objective. The distribution of the particles is optimized per frame and serves as an input for optimizing subsequent frames, enabling temporally coherent position updates. Equation (9) can be automatically computed by our fully differentiable pipeline.

4.1.1 Multi-scale Density Representation. In addition to the position update, we also compute smoke densities individually carried by the particles to further eliminate small gaps that may appear due to the sparse discretization, further enhancing the solution of Equation (10). Owing to the low number of sampled particles and the mismatches
between grid and particle transfers, carrying a constant density will either produce grainy (Figure 7, (b)) or diffuse (Figure 7, (c)) volumetric representations, depending on if particle re-distribution (Equation (9)) is applied or not. A simple approach is to interpolate density values directly from the grid over time. Larger kernel sizes could be used to remedy sparse sampling, but would excessively smooth structures and degrade quality.

We take inspiration from Laplacian pyramids, where distinct grid resolution levels are treated separately. In our case, we compute residuals of different support kernel sizes of the particle-to-grid transfer. This efficiently captures both low- and high-frequency information, covering potentially empty smoke regions while also providing sharp reconstruction results. The residual computation of kernels of varying support sizes is synergistically coupled with matching grid resolutions, which creates an efficient multi-scale representation of the smoke.

The multi-scale reconstruction works as follows: we first sample grid densities to the particles. This represents the smoke low-frequency information, which we interpolate to the particle variables \( \rho_0 \). The variables above the first level (e.g., \( \rho_1, \rho_2 \)) will carry residual information computed between subsequent levels. The Lagrangian representations vary between each level because they perform grid-to-particle transfers with progressively reduced kernel support sizes. To compare residuals between Lagrangian representations, we make use of particle-to-grid transfers, which act as a low-pass filter, similarly to blurring operations of Laplacian pyramids. This process is performed until the original grid resolution is matched. Our multi-scale reconstruction is summarized in Algorithm (1). Figure 7 illustrates the impact of using a single scale without (d) and with (e) particle re-distribution (Equation (9)). The multi-scale result with re-distribution (f), which corresponds to our final method, has a higher PSNR (31.89) than its single-scale counterpart (31.39) and is very close to the ground truth (a).

**Algorithm 1:** Multi-scale Density Reconstruction

**Data:** Particle positions \( x^p \) optimized by Equation (9)

- Original grid-based smoke simulation \( \rho^g \)
- Grid node positions \( x^g \)
- Coarsest support kernel radius \( r \)
- Number of pyramid subdivisions \( n_s \)

**Result:** Multi-scale residual density \( \rho^r \) stored on particles

1. \( \rho_0^r \leftarrow I_{g2p}(x^g, \rho^g) \)
2. \( \rho_0^r \leftarrow I_{p2g}(x^p, \rho_0^r, r, x^g) \)
3. for \( i \leftarrow 1 \) to \( n_s \) do
4. \( \rho_i^r \leftarrow I_{g2p}(x^g, \rho_i^r) \)
5. \( r \leftarrow \frac{r}{2} \)
6. \( \rho_i^r \leftarrow I_{p2g}(x^p, \rho_i^r, r, x^g) \)
7. end

4.2 Temporal coherency

The major advantage of our Lagrangian discretization is the inexpensive enforcing of temporal coherency. Since quantities are carried individually per particle, it is intrinsically simple to track how attributes change over time. Neural style gradients are computed on the grid and need to be updated once the neighborhood of a particle changes. To ensure smooth transitions, we apply a Gaussian filter over the density changes of a particle, as shown in Figure 8. Besides being sensitive to density neighborhood changes, stylization gradients are also influenced by the density carried by the particle itself (Section 4.1.1).

![Fig. 8. Particle density (circles) variation for a single particle over time. Temporal coherency is enforced by smoothing density gradients used for stylization from adjacent frames.](image)

To further improve efficiency, and in contrast to TNST, we can keyframe stylizations, i.e., apply stylization to keyframes and interpolate particle attributes in-between. In practice, we reduced the stylization frames by a factor of 2 at max, but more drastic approximations could be used. Sparse keyframes still show temporally smooth transitions, but quality is degraded. Nevertheless, sparse keyframing would still be useful for generating quick previews of the simulation. The impact of sparse keyframing (every 10 frames) is shown in Figure 9.
5 RESULTS

We implemented the method with the tensorflow framework and computed results on a TITAN Xp GPU (12GB). We used mantaflow [Thuerey and Pfaff 2018] for smoke scene generation, a 3D smoke dataset from Kim et al. [2019a] for comparisons with TNST, a 2D smoke dataset from Jamriska et al. [2015] for color stylization, SPlasHSPlasH [Bender 2016; Koschier et al. 2019] for liquid simulations and Houdini for rendering.

Performance. Using particles for stylization eliminates the need for recursively aligning stylization velocities from subsequent frames, which notably improves the computational performance. In combination with our sparse particle representation for smoke (1 particle per cell), simulations of size $200 \times 300 \times 200$ can now be stylized within an hour instead of a day (TNST). The computation time per frame is 0.66 minutes for the Smoke Jet scene shown in Figure 11, which is a speed-up of a factor of 20.41 compared to TNST. This improvement allows artists to more easily test different reference structures (input images) and hence renders neural flow stylization better applicable in production environments. Table 1 gives an overview of the timings and parameters for the individual test scenes. Keyframing (every other frame) was applied to the Smoke Jet (Figure 11) and Double Jets (Figure 12) examples.

Table 1. Performance table.

<table>
<thead>
<tr>
<th>Scene</th>
<th>Resolution</th>
<th># Particles</th>
<th>Time (m/f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moving Sphere (Fig. 10)</td>
<td>$192 \times 192 \times 192$</td>
<td>237K</td>
<td>0.8</td>
</tr>
<tr>
<td>Smoke Jet (Fig. 11)</td>
<td>$200 \times 300 \times 200$</td>
<td>1.2M</td>
<td>0.66</td>
</tr>
<tr>
<td>Double Jets (Fig. 12)</td>
<td>$200 \times 200 \times 200$</td>
<td>$2M/2M$</td>
<td>0.45</td>
</tr>
<tr>
<td>Chocolate (Fig. 13)</td>
<td>$200 \times 200 \times 200$</td>
<td>80K</td>
<td>0.05</td>
</tr>
<tr>
<td>Colored Smoke (Fig. 3)</td>
<td>$800 \times 800$</td>
<td>136K</td>
<td>1.21</td>
</tr>
<tr>
<td>Dam Break (Fig. 14)</td>
<td>$512 \times 1024$</td>
<td>23K</td>
<td>0.58</td>
</tr>
<tr>
<td>Double Dam (Fig. 15)</td>
<td>$512 \times 1024$</td>
<td>$31K/8K$</td>
<td>0.65</td>
</tr>
</tbody>
</table>

Time-coherency. To illustrate the benefit of the Lagrangian formulation, we use a simple test scene where we initialize a smoke sphere with a uniform density. We then move the smoke artificially to the right, and apply the neural stylization to every frame of the sequence. We compare the results of LNST and TNST for different time instances in Figure 10. The top row shows the results of TNST. It can be seen that TNST is not able to preserve constant stylized textures in regions where the density function does not change. This is due to the recursive alignment of stylization gradients, which accumulate errors especially for bigger window sizes. The second row shows the corresponding results with LNST, demonstrating consistent stylization over time since gradients are constant. Also when applied a shearing deformation to the sphere, as shown in the third row, structures remain coherent. If an artist prefers to have changing structures in such situations, noise can be added to the densities carried by the particles, which in turn will induce stylization gradients as shown in the last row.

Smoke Stylization. Figure 11 shows a direct comparison of LNST and TNST applied to the smoke jet dataset of Kim et al. [2019a]. While the resulting structures inherently depend on the underlying representation, they naturally differ and cannot be directly compared with each other. It can be observed, however, that the Lagrangian stylization may lead to more pronounced structures, well visible in the semantic transfer net and the style transfer blue strokes, and that boundaries are smoother, noticeable in the Seated Nude example.

Multi-fluid Stylization. Stylization of multiple fluids is naturally enabled by stylizing different sets of particles with different input images. Figure 12 shows a simulation of two smoke jets colliding, where the left one is stylized with the semantic feature net and the right one with the style transfer of the input image spirals. Transferred structures are retained per fluid type even if the flow undergoes complex mixing effects.

Stylization of Liquids. We use a simple differentiable renderer for stylization of liquids. Unlike smoke renderer, which integrates media radiance scattered in the medium, we compute the amount of diffused light, i.e., absorbed light except transmitted by its liquid volume [Ihmsen et al. 2012], which is given by

$$
\tau(x, r) = e^{-\int_0^r d(t) \, dr}
$$

$$
I_{ij} = 1 - \tau(t_{max}, r).
$$

(11)
Fig. 11. Semantic transfer applied to the smokejet simulation of [Kim et al. 2019a] (leftmost column). Stylized results are shown for our LNST (top) and TNST (bottom) for semantic feature transfer net (second column) and input images blue strokes, Seated Nude, and fire (last three columns).

Fig. 12. Two colliding smoke jets, which are stylized individually with the semantic feature net and input image spirals. The Lagrangian representation enables coherent stylization of multiple fluids even if the flow undergoes complex mixing.

Figure 13 shows the results of a stylized SPH simulation computed with SPlisHSPlasH [Bender 2016]. We applied the patterns spiral and diagonal to a thin sheet simulation.

Color Transfer. We transfer color information from input images to flow fields by storing a color value per particle and optimizing it by Equation (7). This can be applied to any grid-based or particle-based smoke or liquid simulation. In Figure 14 we applied the color stylization to a 2D dam break simulation using different example images, and in Figure 15 to two liquids with distinct types (and hence color). The accompanying videos show that local color structures change very smoothly over time, which is attributed to the improved time-coherency of the Lagrangian stylization. This is especially well visible in Figure 3, where two subsequent frames are shown for TNST and LNST. In this example, we have transferred the style blue stroke to a smoke scene. The close-up views reveal discontinuities for TNST, while LNST shows smooth transitions for color structures.

We have presented a Lagrangian approach for neural flow stylization with respect to artist control of the stylization. A key property of our approach is that it is not restricted to any particular fluid solver type (i.e., grids, particles, hybrid solvers). To enable this, we have introduced a strategy for grid-to-particle transfer (and vice versa) to efficiently update attributes and gradients, and a re-simulation that can be effectively applied to grid and particle fluid representations. This generality of our method facilitates seamless integration of neural style transfer into existing content production workflows.

A current limitation of the method is that we use a simple differentiable renderer for liquids. While this works well for some scenarios, a dedicated differentiable renderer for liquids would improve the resulting quality and especially also support a wider range of liquid simulation setups. Similar to the smoke renderer, such a liquid renderer must be differentiable as gradients are back-propagated in the optimization. It must also be efficient as the renderer is used in each step of the optimization. Although the complexity of the renderer has a direct influence on the quality of the results, we suspect that, analogously to our smoke renderer, a lightweight renderer that can recover the core flow structures is sufficient for stylizing liquids.

We have shown that LNST enables novel effects and a high degree of art-directability, which renders flow stylization more practical in production workflows. However, we have not tested the method on large-scale simulations that are typically used in such settings. While our method can handle up to 2 million particles, larger scenes are restricted by the available memory. Moreover, in practical settings the scene complexity is higher, which potentially poses challenges with respect to artist control of the stylization.

By reducing the computation time for stylizing an entire simulation from one day with TNST to a single hour with LNST renders the method much more practical for digital artists. However, for testing different input structures, a real-time method would be desirable. Recent concepts presented on neural image stylization might be mapped to 3D simulations to further improve efficiency.

ACKNOWLEDGMENTS

The authors would like to thank Fraser Rothnie for his artistic contributions. The work was supported by the Swiss National Science Foundation under Grant No.: 200021_168997.

REFERENCES


ACM Trans. Graph., Vol. 39, No. 4, Article 1. Publication date: July 2020.