TSDN: Transport-based Stylization for Dynamic NeRF

Y. Gong[1], M. Song[1], X. Ren[2], Y. Liao[1] and Y. Zhang[†1]

[1]College of Computer Science, Sichuan University, China
[2]Wechat, Tencent Inc., China

Figure 1: *We introduce TSDN, a new method for dynamic NeRF stylization. TSDN is able to repaint dynamic scenes with detailed geometrically stylized features while maintaining the original color style if desired.*

Abstract

While previous Neural Radiance Fields (NeRF) stylization methods achieve visually appealing results on transferring color style for static NeRF scenes, they lack the ability to stylize dynamic NeRF scenes with geometrically stylized features (like brushstrokes or feature elements from artists' works), which is also important for style transfer. However, directly stylizing each frame of dynamic NeRF independently with geometrically stylized features would lead to flickering results due to bad feature alignment. To overcome these problems, in this paper, we propose Transport-based Stylization for Dynamic NeRF (TSDN), a new dynamic NeRF stylization method that is able to stylize geometric features and align them with the motion in the scene. TSDN utilizes stylization guiding velocity fields to advect dynamic NeRF to get stylized results and then transfers these velocity fields between frames to maintain feature alignment. Also, to deal with the noisy stylized results due to the ambiguity of the deformation field, we propose a feature advection scheme and a novel regularization function specified for dynamic NeRF. The experiment results show that our method has the ability to stylize dynamic scenes with detailed geometrically stylized features from videos or multi-view image inputs, while preserving the original color style if desired. This capability is not present in previous video stylization methods.

CCS Concepts

• ***Computing methodologies*** → ***Image processing; Volumetric models;*** *Shape representations;*

† Corresponding author.

DOI: 10.2312/pg.20241300

1. Introduction

Extensive researches have been done on how to style 3D scenes by combining neural style transfer with NeRF [CTT*22, ZKB*22, KPL23, LZC*23, NPLX22]. This combination produces visually appealing stylized 3D scenes from multi-view image inputs.

Though most of these stylization works focus on transferring color features, transferring geometrical elements like brushstrokes or elements from artists' work would yield superior results that more closely resemble the style image. Among them, [ZKB*22] focuses on transferring a static scene with such geometrically stylized features, and is shown to be better stylized according to user preference. [KKSS20] also pointed out that geometric features are vital to the stylization. In the other line of research, while the initial NeRF [MST*20] was designed for static scenes, subsequent research expanded it to dynamic NeRF [PSH*21, PSB*21, GCD*22, FYW*22, DZY*21, GSKH21], allowing the depiction of dynamic 3D scenes from images and novel view synthesis.

However, transferring the geometrical features of the style image to a dynamic NeRF is still a challenge. This is because current static NeRF stylization approaches are not sufficient for stylizing detailed geometry features. Also, they lack the ability to maintain temporal consistency by aligning stylized features with the motion of surfaces in a dynamic scene.

To address the limitation mentioned above, inspired by the fluid stylization method Transport-based Neural Style Transfer (TNST) [KAGS19], we introduce a novel approach called Transport-based Stylization for Dynamic NeRF (TSDN) to stylize general scenes of dynamic NeRF in this paper. We adapt the stylization guiding velocity field to advect NeRF field to get more faithful geometrically stylized results. To maintain temporal consistency, we leverage advection to transport stylization guiding velocity fields from frame to frame.

As a result, our method is capable of stylizing a dynamic scene with geometrically stylized features. In the meantime, we can align such features with surface motion to maintain temporal consistency. Applying TSDN to videos by turning them into dynamic NeRF introduces a new type of video stylization. This type of video stylization enables users to synthesize stylized novel views and repaint the video with detailed geometrically stylized features from the brushstrokes of artists' work or object elements from the style image, while optionally maintaining the color style of the video unchanged. In other words, while previous video stylization methods aimed to preserve object structures to maintain temporal consistency, our method is able to alter the structure of the object to match the target style and also maintain temporal consistency by feature alignment between frames. To our knowledge, this sort of stylization is not possible with previous video stylization methods.

Technically, we introduce stylization guiding velocity fields to stylized dynamic NeRF. We also propose an advection scheme for dynamic NeRF to get rid of noise when transferring features between frames. We present a novel regularization term to regularize the deformation field's ambiguity, thus addressing feature degeneration. Simultaneously, we design a novel two-stage back propagation scheme to address the out-of-memory issue when applying this regularization term. At last, we provide a more suitable TNST pipeline for dynamic NeRF to prevent features from being washed out during iterations and becoming noisy when applying stylization guiding velocity fields.

2. Related work

2.1. Dynamic NeRF

[MST*20] first introduced NeRF, significantly accelerating the development of 3D scene reconstruction from photos. There are two mainstream approaches to the research on dynamic NeRF. The first class is based on a deformation field that changes over time while maintaining a static canonical frame. [PSH*21, PSB*21, GCD*22, LGM*23] propose to use the deformation field, which is able to learn large deformation or movement. [GCD*22] makes further improvements by replacing the MLP with voxel grids, accelerating the training time and inference time. Another class, [DZY*21, GSKH21, LSZ*22, XHKK21], adds a time-embedding before feeding the feature vector into the MLP while sampling the positions in the scenes. The MLP outputs radiance or density according to time, which makes the scenes dynamic as time evolves. [FYW*22] introduces a hybrid strategy of the above two methods.

2.2. Neural style transfer

The seminal work [GEB16] first introduced Neural Style Transfer. For videos or sequences, [TFK*20] proposes to use a random sampled path-based training for keyframe-based video stylization. [WYXL20, WXZ*20] leverages optical flow to add a temporal constraint to the stylized frames of videos. To speed things up, [WHSX21, DTD*20] proposes online versions of Neural Style Transfer, which use feed-forward networks to avoid time-consuming optimization. [KKP*22] makes further improvements to the original gram matrix loss by proposing neural neighboring style transfer, which is flexible enough to produce high-quality stylized results.

2.3. Stylized NeRF

Researchers have extensively studied style transfer for NeRF. [CTT*22] introduces a hypernetwork to alter the radiance field in order to learn an arbitrary style transfer. [ZKB*22] proposes to use a novel style loss called Nearest Neighbor Feature Matching (NNFM), which is similar to the [KKP*22]. NNFM is able to produce higher-quality stylized results, like faithful brushstrokes. [NPLX22] proposes to adapt the style transfer in image space and then optimize the radiance field using the stylized ones. [XLSL23] proposes to include additional changes in the density field by Deform Net, making it possible to stylize NeRF by transferring style information to the geometry. Our method resembles the work in [XLSL23]; both try to use another field to deform NeRF fields to get stylized results. However, we focus on how to stylize a dynamic scene while maintaining feature alignment, while they focus on a static scene. [KPL23, WJC*23, SCD*23] introduces approaches offering a semantic NeRF style transfer framework. [LZC*23] proposes a zero-shot manner to stylize NeRF without multiple optimization iterations. There are also a few works that focus on

example-based NeRF stylization rather than using style reference images. [ZHX*23] use a single stylized 2D view as a reference to stylize the 3D NeRF scene. [FLNP*24] proposes using DINO [CTM*21] features extracted from multi-view rendered images from NeRF fields to help the transfer from an example NeRF to a target NeRF.

There are also some concurrent works that share some similarities with ours. [JNS*24] proposes to use the depth map as a style guide to transfer the geometry of the static NeRF scenes, while our method focuses on the dynamic NeRF scene and uses the color map as the style guide. [LCW*24] also focus on dynamic NeRF scenes, but they target stylizing the appearance using a 2D stylized view as a reference and freezing the geometry. In contrast, we stylize both appearance and geometry with a style image.

2.4. Transport-Based Neural Style Transfer

Prior researchers extensively investigated Transport-Based Neural Style Transfer, which is used to stylize fluid motion and add dynamic fluid motion patterns to the simulation sequence. [KAGS19] first proposes to stylize an Euler fluid density field using a stylized density field. It focuses on optimizing a stylization guiding velocity field to advect the density field of fluid simulation. Later [KAGS20] focus on methods for stylized Lagrangian view fluid simulation. Subsequently, [AONA22] proposes a simplified version of the previous Euler method, [KAGS19]. They [AONA22] also show linear advection and no divergence projection are needed for satisfying results. This technique has already been applied in film production [HHK*23, KAOT23].

3. Preliminaries

3.1. Transport-Based Neural Style Transfer(TNST) for fluids

Transport Neural Style Transfer (TNST) [KAGS19] was designed for fluid stylization. We first introduce how TNST stylizes a single frame and then talk about how to maintain temporal consistency by aligning features between frames.

TNST is primarily based on the advection equation to stylize the density field and align features between frames to maintain temporal consistency. Advection equation is a well-established partial differential equation in physics. It describes how a physical quantity is transported via a velocity field. Given a velocity field $\mathbf{u}$, the advection equation for a general field q is:

$$\frac{\partial q}{\partial t} + \mathbf{u} \cdot \nabla q = 0$$

Solving this equation with a time step gives the evolution of field q. An advection operator $\mathcal{A}$ is employed to represent such an operation:

$$q_{t+1} = \mathcal{A}(q_t, \mathbf{u}_t)$$

TNST runs in an iterative manner. For frame t at iteration i, TNST aims to optimize a stylization guiding velocity field $\mathbf{v}_{t,i}$ that can advect the density field d_t in a way that minimizes the differences of latent features between the rendered image of the advected density field and the target style image, as in Eq. 1. We follow the same notion from [AONA22] that we use $\mathbf{u}_t$ to represent the simulation velocity field and use $\mathbf{v}_{t,i}$ to represent the stylization guiding velocity field. There is no subscript i in $\mathbf{u}_t$ because the simulation velocity fields don't change in iterations. For frame t at iteration i, TNST optimized the following loss function:

$$\mathbf{v}_{t,i} = \arg\min_{\mathbf{v}_{t,i}} L_{\text{TNST}}(\mathbf{v}_{t,i}, d_t) \tag{1}$$

$$L_{\text{TNST}}(\mathbf{v}_{t,i}, d_t) = \lambda_c * L_{\text{VGG}}(I_{c,t}, I_{s,t}) + \lambda_s * L_{\text{VGG}}(I_{\text{Style}}, I_{s,t})$$

$$I_{s,t} = \mathcal{R}(\mathcal{A}(d_t, \mathbf{v}_{t,i})), I_{c,t} = \mathcal{R}(d_t)$$

$$L_{\text{VGG}}(I_1, I_2) = \|(\Phi(I_1)) - (\Phi(I_2))\|_2$$

where $\mathcal{R}$ represents a differential render. $I_{s,t}$ is the transmistance results rendered from the density field advected by the stylization guiding velocity field, and $I_{c,t}$ is the transmistance rendered results rendered from the original density field without advection. Φ represents neural latent feature statics. λ_c and λ_s refers to the weight of content and style during Neural Style Transfer. I_{Style} is the target style image.

To maintain temporal consistency, the core problem is to find a suitable initial value $\hat{\mathbf{v}}_{t,i}$ used for the optimization of Eq. 1 that can align features between frames. [AONA22] proposes to obatin $\hat{\mathbf{v}}_{t,i}$ by advecting the stylization guiding velocity $\mathbf{v}_{t-1,i}$ with the simulation velocity $\mathbf{u}_{t-1}$. The reason that this routine could lead to temporal consistency results is that the advected $\hat{\mathbf{v}}_{t,i}$ aligns the stylized features between frame $t-1$ and t. Such alignment leads to strong feature consistency, resulting in coherent outcomes, as highlighted by [RDB16].

$$\hat{\mathbf{v}}_{t,i} = \mathcal{A}(\mathbf{v}_{t-1,i}, \mathbf{u}_{t-1}) \tag{2}$$

[AONA22] adapts a pipeline that takes multiple iterations. To blend results from multiple iterations, it further suggests to combine the initial values with results from previous iterations using an Exponential Moving Average method.

$$\hat{\mathbf{v}}_{t,i} = \begin{cases} \mathbf{v}_{0,i-1}, & t = 0 \\ (1-\alpha)\mathbf{v}_{t,i-1} + \alpha\mathcal{A}(\mathbf{v}_{t-1,i}, \mathbf{u}_{t-1})), & t > 0 \end{cases} \tag{3}$$

3.2. Dynamic NeRF

NeRF employs fields to represent a continuous scene as a 4D function: $\bar{\mathbf{c}} := [\mathbf{c}, \sigma](\mathbf{x}, \mathbf{d}) : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^4$, where $\mathbf{c}$ represents the RGB color, σ denotes the volume density, $\mathbf{x} = (x, y, z)$ indicates the sample position, and $\mathbf{d}$ corresponds to the view direction. The type of dynamic NeRF we adapt is based on a deformation field that changes over time while maintaining a static canonical frame. A deformation field is another field that takes sample position and time as input and then outputs the offset of its sample position: $\mathbf{T}(\mathbf{x}, t) : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}^3$. The sampled value of the deformation field is added to the ray sample positions, resulting in new sample positions. These new sample positions are then used to sample the canonical frame, leading to time-dependent results.

$$\bar{\mathbf{c}}_t(\mathbf{x}, \mathbf{d}) = \bar{\mathbf{c}}(\mathbf{x} + \mathbf{T}(\mathbf{x}, t), \mathbf{d}) \tag{4}$$

4. Method

We begin by explaining how to geometrically stylize a single frame of dynamic NeRF with a stylization guiding velocity field. And then we show how to extend this single-frame stylization to multi-frame stylization by aligning features to maintain temporal consistency in Sec. 4.2. We also cover important improvements that are needed for dynamic NeRF to align features in this part. Next, we demonstrate the new optimization pipeline we designed for dynamic NeRF stylization that can avoid features being washed out during the phase of aligning features. Finally, we demonstrate the style loss we use to achieve stylized brushstroke results.

For preparation, we train an unstylized dynamic NeRF $\bar{\mathbf{c}}$ with the video or the image inputs. The stylization guiding velocity fields $\mathbf{v}_t$ are first initiated with a zero value for each frame of the sequence. (To notice, we drop the subscript i in $\mathbf{v}_t$ because TSDN doesn't run in an iterative routine like the original TNST. More could be found in Sec. 4.3.)

4.1. Single-frame geometrical stylization for dynamic NeRF

The original TNST adapts the stylization guiding velocity field to advect the density field d_t to stylize the fluid at frame t. We leverage the similar idea, using the stylization guiding velocity field $\mathbf{v}_t$ to advect the NeRF field $\bar{\mathbf{c}}_t$ to stylize a single frame of dynamic NeRF.

$\mathbf{v}_t$ is obtained by optimizing the following style loss function with an initial value $\hat{\mathbf{v}}_t$:

$$\mathbf{v}_t = \arg\min_{\mathbf{v}_t} L_{\text{TSDN}}(\mathbf{v}_t, \bar{\mathbf{c}}_t) \tag{5}$$

$$L_{\text{TSDN}}(\mathbf{v}_t, \bar{\mathbf{c}}_t) = \lambda_c * L_{\text{NNFM}}\left(I_{c,t}, I_{s,t}\right) + \lambda_s * L_{\text{VGG}}\left(I_{\text{Style}}, I_{s,t}\right)$$

$$I_{s,t} = \mathcal{R}(\bar{\mathbf{c}}_t^{sty}), I_{c,t} = \mathcal{R}(\bar{\mathbf{c}}_t), \bar{\mathbf{c}}_t^{sty} = \mathcal{A}(\bar{\mathbf{c}}_t, \mathbf{v}_t)$$

where $\mathcal{R}$ represents a differential render, that takes the NeRF fields as input and output an image. I_{Style} is the target stylized image. $I_{c,t}$ and $I_{s,t}$ are the original and stylized NeRF rendered results separately. λ_s controls the stylization strength, and λ_c controls the resembleness to $I_{c,t}$. Stylized NeRF fields $\bar{\mathbf{c}}_t^{sty}$ is obtained by advecting the unstylized NeRF fields $\bar{\mathbf{c}}_t$ by the stylization guiding velocity field $\mathbf{v}_t$, and $\mathcal{A}$ is the advection operator.

To notice, the stylization guiding velocity field $\mathbf{v}_t$ transports the NeRF field to form geometry features without altering the original color style. This is because $\bar{\mathbf{c}}_t$ is only advected by the stylization guiding velocity field $\mathbf{v}_t$, leading to the fact that the radiance field $\mathbf{c}_t$ in $\bar{\mathbf{c}}_t$ is only being redistributed.

4.2. Multi-frame geometrical stylization with temporal consistency for dynamic NeRF

In the previous section, we talk about how to geometrically stylize a single frame. In this section, we extend it to stylize multiple frames while maintain temporal consistency.

To maintain temporal consistency, the core is to align features between framses. This is because aligning features from the previous frame to the current one makes frames maintaining strong feature consistency, resulting in coherent outcomes.

To achieve feature alignment, the original TNST tries to find a suitable initial value $\hat{\mathbf{v}}_t$ used for the optimization of Eq. 1. It is accomplished by transferring the stylization guiding velocity field $\mathbf{v}_{t-1}$ from the previous frame to the current frame by the simulation velocity field $\mathbf{u}_{t-1}$, as in Eq. 2.

Inspired by this idea, in the stylization of dynamic NeRF, we would expect to find a suitable initial value $\hat{\mathbf{v}}_t$ used for the optimization of Eq. 5 that can align features between frames. This may be achieved by transferring the previous stylization guiding velocity field $\mathbf{v}_{t-1}$ to $\hat{\mathbf{v}}_t$ using the dynamic mechanism of the dynamic NeRF.

However, naively transferring stylization guiding velocity field between frames will lead to noisy results. A straightforward solution would be to leverage the definition of deformation field to transfer the $\mathbf{v}_{t-1}$ between frames as σ and $\mathbf{c}$ as in Eq. 4. However, such transfers would result in noisy results. To overcome this challenge, we propose a new advection scheme (Sec. 4.2.1) for dyanmic NeRF to address the noisy results in Sec. 4.2.1.

Ambigious deformation fields will lead to feature degeneration. The undetermined problem of dynamic NeRF optimization would result in an ambiguous deformation field. Such ambiguity would lead to feature degeneration after the transfer of the stylization guiding velocity field. To overcome the above problem, we propose a novel regularization term in Sec. 4.2.2 to prevent feature degeneration. Our improvements result in an additional function $\mathcal{G}_{\text{TSDN}}$ that is used to find the initial value $\hat{\mathbf{v}}_t$.

$$\hat{\mathbf{v}}_t = \mathcal{G}_{\text{TSDN}}(\mathbf{v}_{t-1}, \mathbf{u}_{t-1}, \delta\mathbf{u}_{t-1})$$

where $\delta\mathbf{u}_{t-1}$ is obtained by optimizing our novel regularization function.

4.2.1. Noise-free advection schemes

A straightforward approach to transferring the stylization guiding velocity field between frames is to utilize the deformation field directly, as in Eq. 6. The stylization guiding velocity field in such an approach is defined in the reference field and accessed by the deformation field. Such transfer is the same way as σ and $\mathbf{c}$ are transported between frames in dynamic NeRF, as in Eq. 4.

$$\mathbf{v}_t(\mathbf{x}) = \mathbf{v}(\mathbf{x} + \mathbf{T}(\mathbf{x}, t)) \tag{6}$$

However, such a scheme is vulnerable due to the disturbances in the deformation field. The disturbances would lead to noisy advected stylized results.

To overcome the challenge above, we propose a feature advection scheme by explicitly acquiring the object velocity field. At time t, we utilize Finite Difference (Eq. 7) and the below extrapolation strategy to calculate the object's velocity field. Then we use it to advect the stylization guiding velocity field $\mathbf{v}_{t-1}$ between frames, as in Eq. 2.

$$\Delta\mathbf{x}_t = \mathbf{T}(\mathbf{x}_G, t), \Delta\mathbf{x}_{t-1} = \mathbf{T}(\mathbf{x}_G, t-1)$$

$$\mathbf{u}_{t-1} = \frac{-(\Delta\mathbf{x}_t - \Delta\mathbf{x}_{t-1})}{\Delta t} \tag{7}$$

$\mathbf{x}_G$ refers to the gird center of the field that we want to advect. The reason why there is a negative sign in the derivation of $\mathbf{u}_t$ is that the deformation field acts as a pull-back operation. So the motion of the

object is opposite of that look-up operation from the deformation field.

After that, we leverage the density value to mask whether a grid point in $\mathbf{u}_t$ is valid, and then apply an extrapolation from the valid area to the invalid area. The extrapolation provides a valid value for positions that are not regularized because they do not contribute to σ or $\mathbf{c}$ during the optimization of dynamic NeRF. Then we apply a Gaussian kernel to the velocity field, which helps filter the high-frequency noise from the deformation field. After the derivation of the object velocity field, we adapt a second-order Runge-Kutta method to advect the stylization guiding velocity field by it. The reason why we explicitly acquire u is that this leads to discretizing sampling from the implicit continuous deformation fields. The advection procedure becomes more controllable and smoother. Also, the discretizing and above operations help prevent advected results from being affected by high-frequency noise in the deformation fields.

4.2.2. Feature-maintainable regularization function

In this section, we first analyze the factors that cause stylized feature degeneration due to the ambiguous nature of the deformation field, and then introduce our novel regularization function to regularize the deformation field.

The underdetermination of the dynamic NeRF optimization problem causes the degeneration. When reconstructing the dynamic NeRF from a sequence of single-view image inputs, there are infinite solutions for deformation fields. This is due to the fact that merely focusing on the mean square error or other pixel-wise loss between the target images and the rendered ones is not sufficient to achieve realistic motion reconstruction results. Such ambiguity in the deformation field will derive an ambiguous object velocity field. For example, in an outdoor environment, the color of a point on the ground with diffuse material may be quite smooth in its neighborhood due to the diffuse illumination. In the previous frame, all points in the neighborhood of the target point that share the same color can be regarded as potential candidates for the prior location of the target point. The recent work of [YYZ*23] also highlights additional ambiguity, although it addresses different problems. To tackle this issue, we present a compromise solution: a regularization function L_{Reg} that tries to "dig out" stylized features to help advect stylized features smoothly. It aims to find a delta object velocity $\delta\mathbf{u}_{t-1}$ to regularize the object velocity field between frames $t-1$ and t.

$$\delta\mathbf{u}_{t-1} = \underset{\delta\mathbf{u}_{t-1}}{\arg\min}\ L_{\text{Reg}}(\delta\mathbf{u}_{t-1}, \bar{\mathbf{c}}, \mathbf{u}_{t-1}, \mathbf{v}_{t-1}) \tag{8}$$

$$\begin{aligned} L_{\text{Reg}}(\delta\mathbf{u}_{t-1}, \bar{\mathbf{c}}, \mathbf{u}_{t-1}, \mathbf{v}_{t-1}) &= \lambda_{\text{TSDN}} * L_{\text{TSDN}}(\mathbf{v}_t^*, \bar{\mathbf{c}}_t) \\ &+ \lambda_{\text{mseE}} * L_{\text{mse}}\left(I_{\text{c},t}, I_{\text{c},t}^*\right) + \lambda_{\text{guided}} * \left(1.0 - L_{\text{SSIM}}\left(I_{\text{s},t}, I_{\text{guided},t}\right)\right) \end{aligned} \tag{9}$$

$$\begin{aligned} \mathbf{u}_{t-1}^* = \mathbf{u}_{t-1} + \delta\mathbf{u}_{t-1}, \mathbf{v}_t^* = \mathcal{A}(\mathbf{v}_{t-1}, \mathbf{u}_{t-1}^*)) \\ I_{\text{c},t} = \mathcal{R}(\bar{\mathbf{c}}_t) \\ I_{\text{c},t}^* = \mathcal{R}(\bar{\mathbf{c}}_t^*), \bar{\mathbf{c}}_t^* = \mathcal{A}(\bar{\mathbf{c}}_{t-1}, \mathbf{u}_{t-1}^*) \\ L_{\text{mse}}(I_1, I_2) = \|I_1 - I_2\|_2 \end{aligned} \tag{10}$$

$$I_{\text{s},t} = \mathcal{R}(\bar{\mathbf{c}}_t^{sty}) = \mathcal{R}(\mathcal{A}(\bar{\mathbf{c}}_t, \mathbf{v}_t^*)) \tag{11}$$

where $I_{\text{c},t}^*$ is unstylized rendered results affected by our new object velocity field $\mathbf{u}_{t-1}^*$. $I_{\text{c},t}$ is the unstylized render result without being affected. L_{SSIM} is the Structure Similarity Index Measure, commonly used in computer vision. λ_{TSDN}, λ_{mse} and λ_{guided} are the hyperparameters adjusted according to each scene.

The regularization function in Eq. 9 represents a joint optimization process. It incorporates the stylization guiding velocity field from the previous frame and regularizes the delta object velocity field $\delta\mathbf{u}_{t-1}$, trying to find a suitable one that can extract stylized features while maintaining similar stylized strucutre with the guided synthesis image $I_{\text{guide},t}$. At the same time, it constrains that the rendered result of the NeRF field in the previous frame $\bar{\mathbf{c}}_{t-1}$ advect by the new object velocity field $\mathbf{u}_{t-1}^*$ are similar to that of the original $\bar{\mathbf{c}}_t$ by the factor λ_{mse}.

We adapt the example-based image synthesis algorithm in [FJL*16] to synthesize $I_{\text{guide},t}$, inspired by [JST*19]. $I_{\text{guide},t}$ is composed by setting $I_{\text{s},t-1}$ as the example and using $I_{\text{c},t-1}$ and $I_{\text{c},t}$ as the feature guidance. This algorithm runs by finding the best matching pixel q in $I_{\text{s},t-1}$ and assigning to another pixel p in $I_{\text{guided},t}$ by minimizing the following energy:

$$\sum_{q \in I_{\text{guided}}} \min_{p \in I_{\text{s},t-1}} E_{\text{example}}\left(I_{\text{s},t-1}, I_{\text{guide},t}, I_{\text{c},t}, I_{\text{c},t-1}, p, q\right) \tag{12}$$

where E_{example} refers to the "example-based synthesis energy function" in [FJL*16], and is defined as below:

$$\begin{aligned} &E_{\text{example}}\left(I_{\text{s},t-1}, I_{\text{guided},t}, I_{\text{c},t}, I_{\text{c},t-1}, p, q\right) \\ &= \left\|I_{\text{s},t-1}(p) - I_{\text{guided},t}(q)\right\|^2 + \lambda_{\text{guided}} \left\|I_{\text{c},t}(p) - I_{\text{c},t-1}(q)\right\|^2 \end{aligned}$$

The optimization of Eq. 12 is solved by EM-like iterations with uniform usage of source patches as in [FJL*16]. For more information on how the optimization is solved, we refer readers to [FJL*16]. Such example-based synthesis is similar to [JST*19], while we only use color for guidance.

Also, we advect the change of the delta object velocity field from the previous frame to the current frame, providing it as the initial value $\delta\mathbf{u}_{t-1}$ of the optimization of Eq. 9:

$$\delta\mathbf{u}_{t-1} = \mathcal{A}(\delta\mathbf{u}_{t-2}, \mathbf{u}_{t-1}) \tag{13}$$

The delta object velocity field $\delta\mathbf{u}_{t-1}$ is then used to derive the initial value $\hat{\mathbf{v}}_t$:

$$\hat{\mathbf{v}}_t = \mathcal{G}_{\text{TSDN}}(\mathbf{v}_{t-1}, \mathbf{u}_{t-1}, \delta\mathbf{u}_{t-1}) = \mathcal{A}(\mathbf{v}_{t-1}, \mathbf{u}_{t-1} + \delta\mathbf{u}_{t-1}) \tag{14}$$

where $\mathbf{u}_{t-1}$ is the object velocity we obtained from Sec. 4.2.1.

Consequently, we found that our method is effective and capable of maintaining stylized features between frames.

However, applying such a regularization function is not easy because recording the full graph of the entire field advection and rendering the whole image of $I_{\text{s},t}$ (in L_{TSDN}) to get the stylized loss consumes huge GPU memory. Prior works [ZKB*22] also highlight this issue. To overcome such a problem, we propose a two-stage deferred back-propagation strategy, inspired by [ZKB*22].

We first advect the $\mathbf{v}_{t-1}$ with $\mathbf{u}^*_{t-1}$ and use the result $\mathbf{v}^*_t$ to render the images $I_{s,t}$ without recording the gradient. Then we use the images to calculate L_{Reg} and derive the gradient $\nabla_{I_{s,t}}\mathcal{L}_{\text{TSDN}}$. After that, we set $\mathbf{v}^*_t$ as the leaf node in the auto-differentiation and batched render the images and backpropagat $\nabla_{I_{s,t}}\mathcal{L}_{\text{TSDN}}$ to $\mathbf{v}^*_t$ to obtain $\nabla_{\mathbf{v}^*_t}\mathcal{L}_{\text{TSDN}}$. Finally, we advect the $\mathbf{v}_{t-1}$ with $\mathbf{u}_{t-1}+\delta\mathbf{u}_{t-1}$ again to calculate $\mathbf{v}^*_t$ and pass $\nabla_{\mathbf{v}^*_t}\mathcal{L}_{\text{TSDN}}$ to update $\delta\mathbf{u}_{t-1}$.

Algorithm 1: Transport-based Stylization for Dynamic NeRF

Input: NeRF fields $\bar{\mathbf{c}}$, deformation field T, frame count n_{frames}, regularization substeps count n_{reg}, style transfer substep count n_{sty}, Leraning rate η.
Output: stylization guiding velocity field $\mathbf{v}_t$

1 $\mathbf{v}_t \leftarrow 0, \delta\mathbf{u}_t \leftarrow 0$
2 **for** $t \leftarrow 0, f_{frames}$ **do**
3 Adjust η, n_{sty}, according to t
4 $\mathbf{u}_{t-1} \leftarrow -\frac{\mathbf{T}(\mathbf{x}_G,t)-\mathbf{T}(\mathbf{x}_G,t-1)}{\Delta t}$ (Sec. 4.2.1)
5 Extrapolate and smooth $\mathbf{u}_{t-1}$
6 $\delta\mathbf{u}_{t-1} \leftarrow \mathcal{A}(\delta\mathbf{u}_{t-2}, \mathbf{u}_{t-1})$ (Sec. 4.2.2)
7 **for** $m \leftarrow 0, n_{reg}$ **do**
8 Calculate regularization $\mathcal{R}_{\text{reg}} \leftarrow L_{\text{Reg}}(\delta\mathbf{u}_{t-1}, \bar{\mathbf{c}}, \mathbf{u}_{t-1}, \mathbf{v}_{t-1})$ (Sec. 4.2.2)
9 Update $\delta\mathbf{u}_{t-1}$ using $\nabla_{\delta\mathbf{u}_{t-1}}\mathcal{R}_{\text{reg}}$
10 **end**
11 $\mathbf{v}_t \leftarrow \mathcal{T}(\mathbf{v}_{t-1}, \mathbf{u}_{t-1}+\delta\mathbf{u}_{t-1})$
12 **for** $m \leftarrow 0, n_{sty}$ **do**
13 Calculate loss $\mathcal{L}_{\text{loss}} \leftarrow L_{\text{TSDN}}(\mathbf{v}_t, \bar{\mathbf{c}}_t)$ (Sec. 4.1)
14 Smooth $\nabla_{\mathbf{v}_t}\mathcal{L}_{\text{loss}}$
15 Update velocity field $\mathbf{v}_t$ using $\nabla_{\mathbf{v}_t}\mathcal{L}_{\text{loss}} * \eta$
16 **end**
17 **end**

4.3. Optimized pipeline with substeps

In this section, we first analyze the difference in optimization targets between NeRF and fluid, that makes the stylization optimization procedure of dynamic NeRF difficult. And then introduce the new optimized pipeline for dynamic NeRF.

In TNST, the stylized output would be an image of the transmittance value from the fluid density field, as in Eq. 4.3. In most camera rays, the transmittance value does not accumulate to 1. This means making stylized patterns on it by advection is easy because manipulating the density field is all the optimization needs. However, compared to dynamic NeRF, although they all use the same framework for volume rendering, the density field d needs to cooperate with the radiance field $\mathbf{c}$ to form the stylized pattern as shown in Eq. 15. Also, the transmistance image of the NeRF field is almost 1 everywhere due to containing hard surfaces.

$$T(t) = \exp\left(-\int_{t_n}^{t} d(\mathbf{r}(s))ds\right), \mathbf{C}(\mathbf{r}) = \int_{t_n}^{t_f} T(t)d(\mathbf{r}(t))dt$$
$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$
$$\mathbf{C}(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}_t(\mathbf{r}(t), \mathbf{d})dt \tag{15}$$

Due to the difference in the optimization target, as the above states, naively applying the original TNST pipeline may not be suitable in dynamic NeRF. We discovered that adapting the original TNST pipeline to dyanmic NeRF leads to features being washed out, resulting in poor stylized results. Additionally, we found the stylization gradient of $\mathbf{v}_t$ is noisy due to the original density field σ and radiance field $\mathbf{c}$ being noisy, leading to poor stylized results.

To overcome such a problem, we implemented a new pipeline that takes substeps during traversal. Our new pipeline doesn't take a few iterations to traverse the whole sequence like TNST, but only traverses the sequence once. In the first frame, we set a normal learning rate η and take multiple substeps n_{sty} to optimize an almost converged stylization guiding velocity field. For the subsequent frames, we first use the above regularization function (Eq. 8) to find the best object velocity field that can present features with a few substeps n_{reg}. After that, we advect the stylization guiding velocity field from the previous frame with the object velocity field we found above. Then we reduce the learning rate η and refine the results with fewer optimization substeps n_{sty}. After each step of optimization, we also adapt a gaussian blur to the gradient of the stylized field. This helps reduce the noise introduced by the color field and sigma field.

In summary, our algorithm can be viewed in Algorithm 1. In lines 3-5, we adapt the method in Sec. 4.2.1 to derive a smoothed object velocity field. Then, in lines 6-10, we regularize the deformation field using the novel regularization function to keep it from degeneration in Sec. 4.2.2. In line 11, we use the regularized object velocity field to advect our stylization guiding velocity field to find a suitable initial value $\hat{\mathbf{v}}_t$ to maintain temporal coherency. Finally, we adapt the optimization strategy in Sec. 4.1 to stylize the NeRF field using the stylization guiding velocity field with $\hat{\mathbf{v}}_t$ as the initial value.

4.4. Detailed geometric brushstrokes and object element transfer

To get detailed stylized geometric brushstrokes, we apply the NNFM loss from [ZKB*22, KKP*22].

$$L_{\text{NNFM}}(I_1, I_2) = \frac{1}{N}\sum_{i,j}\min_{i',j'} D\left((\Phi(I_1))(i,j), (\Phi(I_2))\left(i',j'\right)\right)$$
$$D(\mathbf{v}_1, \mathbf{v}_2) = 1 - \mathbf{v}_1^T\mathbf{v}_2 / \sqrt{\mathbf{v}_1^T\mathbf{v}_1\mathbf{v}_2^T\mathbf{v}_2}$$

where N is the size in the latent feature, and D measures the distance between two latent features. Solely applying NNFM loss without TSDN leads to brushstroke features appearing on the surface of objects, leading to macro features only. After integrating it with our stylization guiding velocity field, the dimension of stylization is increased, broadening the ability to stylize. The geometry

starts to be transformed, leading to detailed, geometrically stylized features. Additionally, the increasing stylization dimenison enables the freezing of another dimenison of color, leading to geometry manipulating only stylization results.

5. Experiments

In this section, we demonstrate a thorough evaluation of our stylization algorithm on dynamic NeRF. We adapt D-NeRF [PCPMMN20] and DAVIS [PPTM*16] for the synthesis and real-world dataset, respectively. Our algorithm is tested on a single NVIDIA RTX 4090 GPU. We refer readers to the supplementary material for video results.

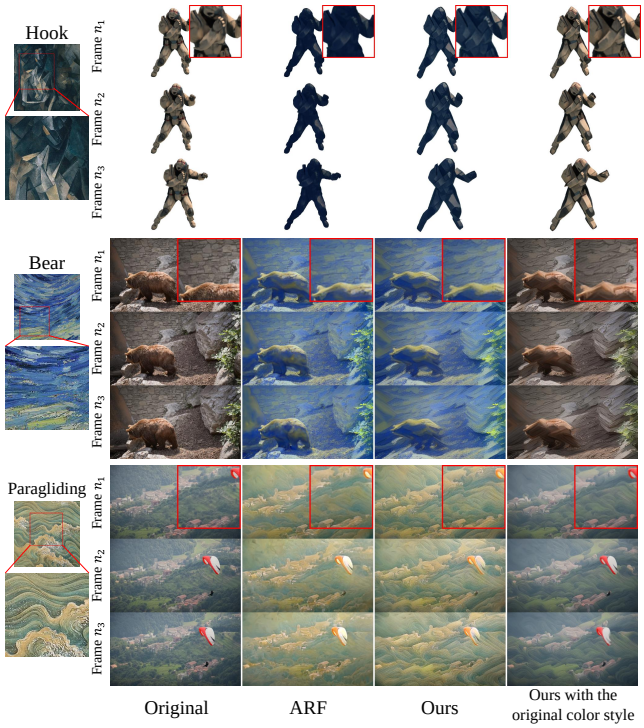

Figure 2: *Qualitative comparison between ARF [ZKB*22] and ours on D-NeRF [PCPMMN20] and DAVIS [PPTM*16] datasets.*

5.1. Qualitative comparison

We compare our results with the static NeRF stylization method ARF [ZKB*22] and two video stylization techniques: CAP-VSTNet [WGZ23] and CCPL [WZDB22]. **Compare with the NeRF stylization method.** We compare TSDN with the NeRF stylization method ARF [ZKB*22]. The qualitative results are shown in Fig. 2. Because the original ARF is designed for static NeRF, applying it directly to dynamic scenes will produce furry results. To give a fair comparison, we reimplement ARF in TiNeuVox [FYW*22], the same framework we used to compare, and stylize each frame in sequence. As the figure shows, TSDN is able to produce more faithful geometrically stylized features from the target images. In the Hook scene, we are able to transfer the unique shape from the target image to the shape of the object, while ARF [ZKB*22] only transfers the color features. In the Bear scene and Paragliding scene, we are able to better transfer the oil painting brushstroke and wave elements in the target images separately than [ZKB*22]. **Compare with video stylization methods.** We compare TSDN with CAP-VSTNet [WGZ23] and CCPL [WZDB22]. The qualitative results are shown in Fig. 3. Compared to CAP-VSTNet [WGZ23], although it is able to produce slightly better temporal coherency results than us in some scenes, CAP-VSTNet is not able to produce faithful stylized geometric features or stylize scenes while maintaining the original color style, like TSDN. For example, in the Bear scene, we are able to transfer the repeatedly appearing shapes in the target image to the stylized results, whereas CAP-VSTNet is only able to transfer the color style of the target image. In the Paragliding scene, we are able to faithfully transfer the cloud elements, while CAP-VSTNet only produces blurry results. While CCPL [WZDB22] is able to produce stylized geometric features, it can't produce faithful and stable geometric stylized features like TSDN. Also, it's not able to produce stylized results while maintaining original color styles, like TSDN. For instance, in the Bus scene, we can style the scene with a hand-drawn brushstroke in the target image, while CCPL only produces dot-like flickering results. In the Rhino scene, we are able to faithfully transfer the vector geometry element. CCPL can also slightly produce such geometry features, but it suffers more flickering.

5.2. Quantitative comparison

5.2.1. User study

To quantify our method's results, we performed a user study to compare TSDN with previous methods in two dimensions: less flickering (temporal coherency) and better geometrical stylization. To compare our method with one of the previous methods on a certain dimension, we design a question showing the original video, the target style image, the stylized results by TSDN, and the stylized results of the previous methods. We ask the user to choose the answer based on a specific dimension in each question. We totally designed 36 questions and randomly picked 20 questions from them for each volunteer. There are 251 valid survey results collected in our user study. Results are shown in Tab. 1.

5.2.2. Metrics evaluation

In order to quantify the degree of stylization achieved by our method, we compute the average VGG Gram Matrix([GEB16]), NNFM([KKP*22, ZKB*22]), and DINO embedding similarity([CTM*21]) scores between the stylized results and the style image, based on the exhibited scenes in Fig. 2 and Fig. 3. The VGG Gram matrix score is the first metric used in the field of style transfer to quantify the degree of stylization. NNFM is a recently proposed style metric that has been shown to be better aligned with the target style image. We use the VGG Gram matrix and NNFM score to help measure the overall stylization of the stylized results. In addition, we use the DINO embedding similarity to assess the geometrical style similarity between the stylized results and the style images. This is because the DINO embedding similarity is sensitive to the structure, which is crucial for measuring the geometrical

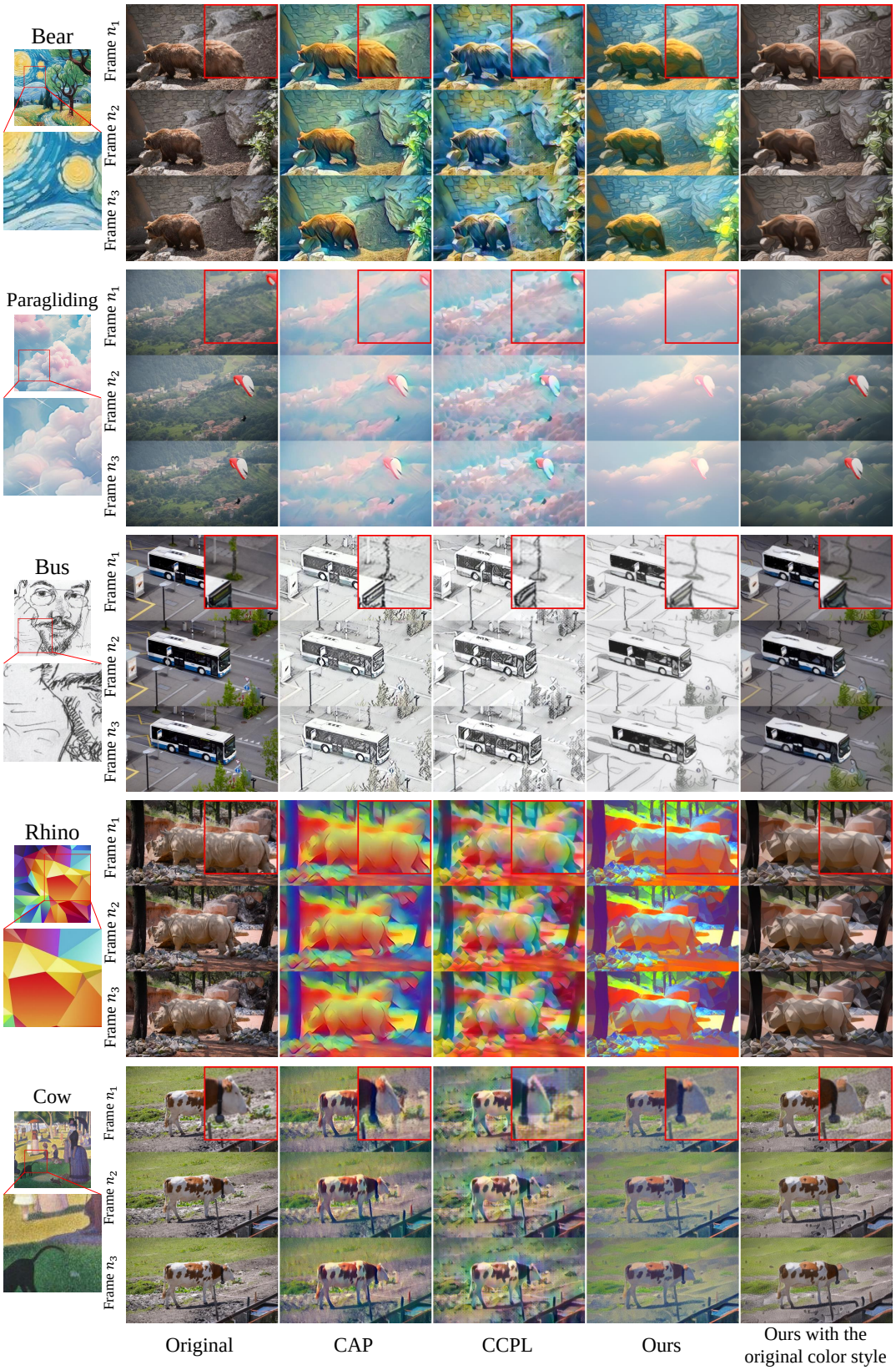

Figure 3: *Qualitative comparison among CAP-VSTNet [WGZ23], CCPL [WZDB22] and ours on DAVIS [PPTM*16] dataset.*

Table 1: *We conducted a user study to compare our method with previous methods. The results are displayed as percentages based on the choices made by the volunteers.*

	Better Temporal Coherency	Better Geometrical Stylization
User Choices: ARF [ZKB*22] vs Ours:		
ARF	47.9%	23.4%
Ours	52.1%	76.6%
User Choices: CAP [WGZ23] vs Ours:		
CAP	36.3%	28.2%
Ours	63.7%	71.8%
User Choices: CCPL [WZDB22] vs Ours:		
CCPL	11.8%	21.9%
Ours	88.2%	78.1%

Table 2: *Quantitative results averaged across **the scenes in Fig. 2**. The best are **bold**.*

Methods	VGG Gram matrix↓			NNFM↓			DINO↑
	Layer 2&4	Layer 6&8	Layer 12&14	Layer 2&4	Layer 6&8	Layer 12&14	
ARF [ZKB*22]	0.0011	0.0133	0.0038	0.3669	0.6870	0.4457	0.6151
Ours	**0.0008**	**0.0098**	**0.0020**	**0.3521**	**0.5734**	**0.3975**	**0.7769**

Table 3: *Quantitative results averaged across **the scenes in Fig. 3**. The best are **bold**.*

Methods	VGG Gram matrix↓			NNFM↓			DINO↑
	Layer 2&4	Layer 6&8	Layer 12&14	Layer 2&4	Layer 6&8	Layer 12&14	
CAP [WGZ23]	0.0040	0.0047	0.0029	0.2532	0.5289	0.4683	0.6278
CCPL [WZDB22]	**0.0004**	**0.0045**	0.0028	0.2302	0.5015	0.4457	0.6452
Ours	0.0006	0.0086	**0.0022**	**0.1851**	**0.3567**	**0.3550**	**0.7141**

stylization. It was also adapted as a metric by [KYO24, TGBD23]. The metric comparison of our methods and the NeRF stylization method ARF is shown in Tab. 2. As the results show, we are always the best in all metric. This is because we introduce another dimension of stylization in the geometry, leading to better fitness. The comparison of CAP, CCPL, and our methods is shown in Tab. 3. We did not evaluate CAP and CCPL on the D-NeRF dataset because these video stylization methods are not capable of handling multi-view image input. As shown in the data, although we are not always the best under all layers in terms of VGG, we are the best in terms of NNFM and DINO score all the time. These indicate that we are better aligned with the target style images and geometrically stylized, as reflected by the NNFM and DINO embedding similarity scores.

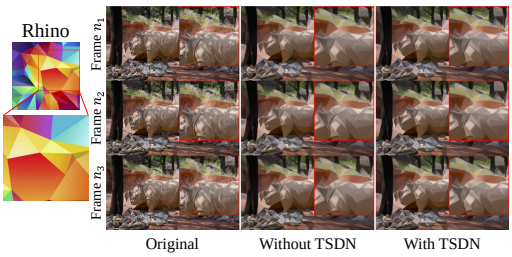

Figure 4: *Ablation study for feature alignment using TSDN.*

5.3. Ablation study

5.3.1. Aligning geometrical stylized features with TSDN

To verify the effectiveness of TSDN, we compare it with a naive implementation that optimizes each frame independently with a stylization guiding velocity field. As illustrated in Fig. 4. As time evolves (from frame n_1 to n_2 and n_2 to n_3), the features stylized by the naive implementation change, leading to flickering results. In contrast, in TSDN, features remain stable across frames, leading to coherent results. It may not be intuitive to observe such a difference in a few frames, so we refer readers to the video for a better demonstration.

5.3.2. Advection scheme and regularization function

This section verifies that our advection scheme and regularization function work well for preserving geometry features during advec-

tion. We expect the stylized features at frame n_1 to be preserved when advected to n_2 (from top to bottom). In other words, stylized features are expected to be still clear and stable across frames. We compare this ability with a naive transfer scheme, our advection scheme without regularization function, and our advection schemes with regularization function. The results are shown in Fig. 5. Results show that as time evolves, in a naive advection, stylized features become blurry. After adapting our advection scheme for dynamic NeRF, as time evolves, stylized features become smoother compared to the naive advection. Finally, implementing our additional regularization function brings back the geometric features.

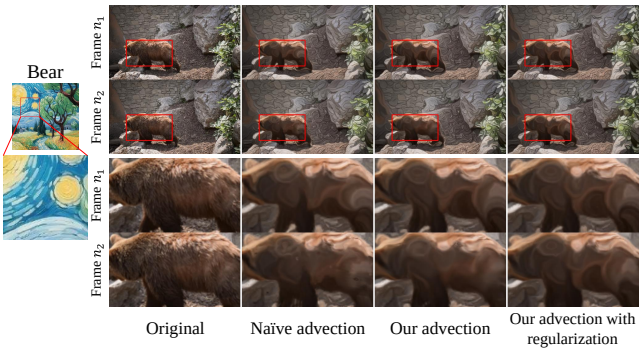

Figure 5: *Ablation study for advection schemes. The red box areas are enlarged and place below to better demonstrate the difference.*

5.3.3. Optimized pipeline targeting NeRF fields

To justify our design choice for the pipeline, we compare results with the original TNST pipeline and our pipeline without and with smooth gradient strategy in Fig. 6. It shows that when adapting the original TNST pipeline, the stylized geometry gets washed out during iteration. In other words, the original TNST pipeline fails to maintain features, whereas our pipeline succeeds in doing so. Additionally, applying the smooth gradient strategy leads to noisy-free results.

Figure 6: *Ablation study for our pipeline.*

5.3.4. Detailed geometric brushstrokes and object element transfer

We use L_{NNFM} to replace the original F_{VGG} that is used in [AONA22]. Although the ability of L_{NNFM} compared to F_{VGG} had been studied by [ZKB*22, KKP*22], we still include a comparison here for our TSDN. Fig. 7 demonstrates a comparison of using L_{VGG} and L_{NNFM}. As the figure shows, with L_{VGG}, the stylized results are not able to present the brushstrokes and object features. With L_{NNFM}, the stylized results show geometrically stylized brushstrokes from the style images.

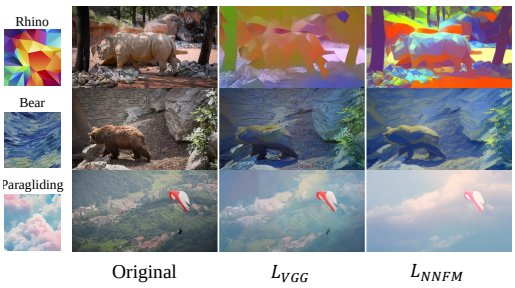

Figure 7: *Ablation study for the style loss we adapted.*

5.4. Novel views

Thanks to the ability of dynamic NeRF, we are able to synthesize stylized novel views, as shown in Fig. 8, while it is not possible with previous video stylization methods like CAP-VSTNet [WGZ23] and CCPL [WZDB22]. We refer readers to the video for a better demonstration.

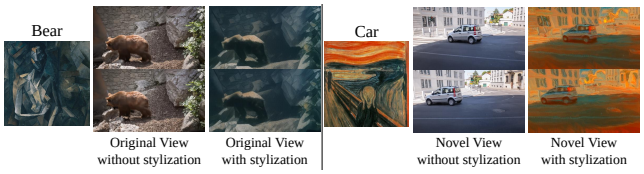

Figure 8: *Example of novel view synthesis and stylization.*

5.5. Implementation details

Our algorithm is built on TiNeuVox [FYW*22] and RoDynRF [LGM*23]. For the synthesis datasets, we directly reconstruct the dynamic NeRF using TiNeuVox [FYW*22]. For most of the real datasets, we first use RoDynRF [LGM*23] to evaluate camera poses. Then we use camera poses to train a dynamic NeRF using TiNeuVox [FYW*22]. This is because we found applying TSDN to VM-decomposition fields [LGM*23] introduces coupling, resulting in bad geometrical stylization. To get colored results, we adapt a simple strategy because we primarily focus on geometry stylization. In the first frame, we simply fine-tune the radiance network while keeping the feature grid unchanged. And we give a very low learning rate for the radiance network in the following frames. This strategy is simple but helps maintain color coherency between frames. To get better color alignment with the style target, we apply the same color transfer algorithm in [ZKB*22] before extracting latent features from images. The stylization guiding velocity fields we use are pure Euler grids because we need to advect these fields explicitly. We use the stylization guiding velocity to advect NeRF

fields to get stylized results with a first-order Euler semi-lagrangian method, as in [AONA22]. However, the NeRF fields are implicit and can't be advected explicitly. To overcome such a challenge, we look up the value of stylization guiding velocity during sampling and use the semi-lagrangian method to find the sample position in the NeRF fields. Such a scheme avoids the need to explicitly advect the NeRF field and is able to produce the desired stylized results.

6. Limitations and future work

One main limitation of our method comes from the quality of the dynamic NeRF. The capacity of the information that dynamic NeRF recorded is limited to the length of frames and the intensity of the motion. For long videos or videos with complex motion, the dynamic NeRF is not able to faithfully reconstruct such scenes, which prevents further stylization with TSDN. As a result, the artifact from the dynamic NeRF leads to flickering stylized results. A solution would be to adapt methods like VM-decomposition [CXG*22] to model fields in dynamic NeRF to improve the representation ability of dynamic NeRF. However, we find that such representations introduce coupling in fields, leading to poor stylization results. We believe that future work could focus on finding a solution to eliminate the drawback of the coupling. Additionally, when targeting large deformations with few training data, the deformation field does not always follow the movement of the real object, leading to incorrect feature alignment. Sometimes it would misclassify the shifting shadows or changes in lighting as moving objects. A future work would be to add some pre-knowledge to the deformation field regularization. Also, we can't use previous optical-flow-based temporal coherency measures to quantify our results. This is due to the fact that our algorithm heavily deforms the geometry, whereas previous measures relied on the optical flow from the geometry. After deformation, the optical flow is no longer valid. A future work may be to find a new metric for us to measure temporal coherency. Another limitation arises from the expenditure of time. As our method works in 3D, we need to do the time-consuming volume rendering. For a regular scene, TSDN takes almost 20-30 seconds per frame. For a scene with complex motion, we need to perform more iterations for the regularization function, resulting in times that can reach nearly 180-300 seconds. Compared to the previous method, the ARF reimplemented in TiNeuVox only needs 10-20 seconds per frame. The video stylization methods, as they don't operate in 3D space, are extremely fast, resulting in a frame stylization in less than one second. Future work may concentrate on using a forward network to regularize the deformation field through data-driven methods, potentially reducing the computational time required for the regularization function. Furthermore, we believe that developing a 2D version that leverages a pseudo-deformation field in image space from optical flows to advect geometrical changes could speed up our methods by eliminating the time-consuming volume rendering process.

7. Conclusion

This paper introduces a novel method for applying stylization to dynamic NeRF. We bring the concepts of advection and velocity field from computational fluid dynamics, enabling geometric stylization and stylized feature alignment between frames. Also, we are able to stylize videos while maintaining their original color style unchanged, leading to a new type of stylization.

8. Acknowledgments

This work was supported by the National Key Project of China (Project Number GJXM92579) and Sichuan Science and Technology Program (Project Number 2023YFG0122).

References

[AONA22] AURAND J., ORTIZ R., NAUER S., AZEVEDO V. C.: Efficient Neural Style Transfer for Volumetric Simulations. *ACM Trans. Graph. 41*, 6 (Nov. 2022). Place: New York, NY, USA Publisher: Association for Computing Machinery. doi:10.1145/3550454.3555517. 3, 10, 11

[CTM*21] CARON M., TOUVRON H., MISRA I., JÉGOU H., MAIRAL J., BOJANOWSKI P., JOULIN A.: Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)* (2021). 3, 7

[CTT*22] CHIANG P.-Z., TSAI M.-S., TSENG H.-Y., LAI W.-S., CHIU W.-C.: Stylizing 3D Scene via Implicit Representation and HyperNetwork. In *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* (Waikoloa, HI, USA, Jan. 2022), IEEE, pp. 215–224. doi:10.1109/WACV51458.2022.00029. 2

[CXG*22] CHEN A., XU Z., GEIGER A., YU J., SU H.: TensoRF: Tensorial Radiance Fields. In *ECCV* (2022). 11

[DTD*20] DENG Y., TANG F., DONG W., SUN W., HUANG F., XU C.: Arbitrary Style Transfer via Multi-Adaptation Network. In *Proceedings of the 28th ACM International Conference on Multimedia* (New York, NY, USA, 2020), MM '20, Association for Computing Machinery, pp. 2719–2727. event-place: Seattle, WA, USA. doi:10.1145/3394171.3414015. 2

[DZY*21] DU Y., ZHANG Y., YU H.-X., TENENBAUM J. B., WU J.: Neural Radiance Flow for 4D View Synthesis and Video Processing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021). 2

[FJL*16] FIŠER J., JAMRIŠKA O., LUKÁČ M., SHECHTMAN E., ASENTE P., LU J., SÝKORA D.: Stylit: Illumination-guided example-based stylization of 3d renderings. *ACM Trans. Graph. 35*, 4 (July 2016). doi:10.1145/2897824.2925948. 5

[FLNP*24] FISCHER M., LI Z., NGUYEN-PHUOC T., BOZIC A., DONG Z., MARSHALL C., RITSCHEL T.: Nerf analogies: Example-based visual attribute transfer for nerfs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024), pp. 4640–4650. 3

[FYW*22] FANG J., YI T., WANG X., XIE L., ZHANG X., LIU W., NIESSNER M., TIAN Q.: Fast Dynamic Radiance Fields with Time-Aware Neural Voxels. In *SIGGRAPH Asia 2022 Conference Papers* (2022). 2, 7, 10

[GCD*22] GUO X., CHEN G., DAI Y., YE X., SUN J., TAN X., DING E.: Neural Deformable Voxel Grid for Fast Optimization of Dynamic View Synthesis. In *Proceedings of the Asian Conference on Computer Vision (ACCV)* (2022). 2

[GEB16] GATYS L. A., ECKER A. S., BETHGE M.: Image Style Transfer Using Convolutional Neural Networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 2414–2423. doi:10.1109/CVPR.2016.265. 2, 7

[GSKH21] GAO C., SARAF A., KOPF J., HUANG J.-B.: Dynamic View Synthesis from Dynamic Monocular Video. In *Proceedings of the IEEE International Conference on Computer Vision* (2021). 2

[HHK*23] HOFFMAN J., HU T., KANYUK P., MARSHALL S., NGUYEN G., SCHROERS H., WITTING P.: Creating Elemental Characters: From Sparks to Fire. In *ACM SIGGRAPH 2023 Talks*. 2023, pp. 1–2. 3

[JNS*24] JUNG H., NAM S., SARAFIANOS N., YOO S., SORKINE-HORNUNG A., RANJAN R.: Geometry transfer for stylizing radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024), pp. 8565–8575. 3

[JST*19] JAMRIŠKA O., SOCHOROVÁ V., TEXLER O., LUKÁČ M., FIŠER J., LU J., SHECHTMAN E., SÝKORA D.: Stylizing video by example. *ACM Trans. Graph. 38*, 4 (jul 2019). doi:10.1145/3306346.3323006. 5

[KAGS19] KIM B., AZEVEDO V., GROSS M., SOLENTHALER B.: Transport-based neural style transfer for smoke simulations. *ACM Transactions on Graphics 38* (Nov. 2019), 1–11. doi:10.1145/3355089.3356560. 2, 3

[KAGS20] KIM B., AZEVEDO V. C., GROSS M., SOLENTHALER B.: Lagrangian Neural Style Transfer for Fluids. *ACM Trans. Graph. 39*, 4 (Aug. 2020). Place: New York, NY, USA Publisher: Association for Computing Machinery. doi:10.1145/3386569.3392473. 3

[KAOT23] KANYUK P., AZEVEDO V., ORTIZ R., TANG J.: Singed Silhouettes and Feed Forward Flames: Volumetric Neural Style Transfer for Expressive Fire Simulation. In *ACM SIGGRAPH 2023 Talks* (New York, NY, USA, 2023), SIGGRAPH '23, Association for Computing Machinery. event-place: Los Angeles, CA, USA. doi:10.1145/3587421.3595435. 3

[KKP*22] KOLKIN N., KUCERA M., PARIS S., SYKORA D., SHECHTMAN E., SHAKHNAROVICH G.: Neural Neighbor Style Transfer, Mar. 2022. arXiv:2203.13215 [cs]. URL: http://arxiv.org/abs/2203.13215, doi:10.48550/arXiv.2203.13215. 2, 6, 7, 10

[KKSS20] KIM S. S., KOLKIN N., SALAVON J., SHAKHNAROVICH G.: Deformable style transfer. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVI 16* (2020), Springer, pp. 246–261. 2

[KPL23] KUMAR M., PANSE N., LAHIRI D.: S2RF: Semantically Stylized Radiance Fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), pp. 2952–2957. 2

[KYO24] KIM G., YOUWANG K., OH T.-H.: FPRF: Feed-forward photorealistic style transfer of large-scale 3D neural radiance fields. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2024). 9

[LCW*24] LI X., CAO Z., WU Y., WANG K., XIAN K., WANG Z., LIN G.: S-dyrf: Reference-based stylized radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2024), pp. 20102–20112. 3

[LGM*23] LIU Y.-L., GAO C., MEULEMAN A., TSENG H.-Y., SARAF A., KIM C., CHUANG Y.-Y., KOPF J., HUANG J.-B.: Robust Dynamic Radiance Fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023). 2, 10

[LSZ*22] LI T., SLAVCHEVA M., ZOLLHOEFER M., GREEN S., LASSNER C., KIM C., SCHMIDT T., LOVEGROVE S., GOESELE M., LV Z.: Neural 3d video synthesis. In *CVPR* (2022). 2

[LZC*23] LIU K., ZHAN F., CHEN Y., ZHANG J., YU Y., EL SADDIK A., LU S., XING E.: StyleRF: Zero-Shot 3D Style Transfer of Neural Radiance Fields. pp. 8338–8348. doi:10.1109/CVPR52729.2023.00806. 2

[MST*20] MILDENHALL B., SRINIVASAN P. P., TANCIK M., BARRON J. T., RAMAMOORTHI R., NG R.: NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV* (2020). 2

[NPLX22] NGUYEN-PHUOC T., LIU F., XIAO L.: SNeRF: Stylized Neural Implicit Representations for 3D Scenes. *ACM Trans. Graph. 41*, 4 (July 2022). Place: New York, NY, USA Publisher: Association for Computing Machinery. doi:10.1145/3528223.3530107. 2

[PCPMMN20] PUMAROLA A., CORONA E., PONS-MOLL G., MORENO-NOGUER F.: D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020). 7

[PPTM*16] PERAZZI F., PONT-TUSET J., MCWILLIAMS B., VAN GOOL L., GROSS M., SORKINE-HORNUNG A.: A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 724–732. doi:10.1109/CVPR.2016.85. 7, 8

[PSB*21] PARK K., SINHA U., BARRON J. T., BOUAZIZ S., GOLDMAN D. B., SEITZ S. M., MARTIN-BRUALLA R.: Nerfies: Deformable Neural Radiance Fields. *ICCV* (2021). 2

[PSH*21] PARK K., SINHA U., HEDMAN P., BARRON J. T., BOUAZIZ S., GOLDMAN D. B., MARTIN-BRUALLA R., SEITZ S. M.: HyperNeRF: A Higher-Dimensional Representation for Topologically Varying Neural Radiance Fields. *ACM Trans. Graph. 40*, 6 (Dec. 2021). Publisher: ACM. 2

[RDB16] RUDER M., DOSOVITSKIY A., BROX T.: Artistic Style Transfer for Videos. In *German Conference on Pattern Recognition* (2016), pp. 26–36. 3

[SCD*23] SONG H., CHOI S., DO H., LEE C., KIM T.: Blending-NeRF: Text-Driven Localized Editing in Neural Radiance Fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023). 2

[TFK*20] TEXLER O., FUTSCHIK D., KUČERA M., JAMRIŠKA O., ŠÁRKA SOCHOROVÁ, CHAI M., TULYAKOV S., SÝKORA D.: Interactive video stylization using few-shot patch-based training. *ACM Transactions on Graphics 39*, 4 (2020), 73. 2

[TGBD23] TUMANYAN N., GEYER M., BAGON S., DEKEL T.: Plug-and-play diffusion features for text-driven image-to-image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), pp. 1921–1930. 9

[WGZ23] WEN L., GAO C., ZOU C.: CAP-VSTNet: Content Affinity Preserved Versatile Style Transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), pp. 18300–18309. 7, 8, 9, 10

[WHSX21] WU X., HU Z., SHENG L., XU D.: StyleFormer: Real-time Arbitrary Style Transfer via Parametric Style Composition. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (2021), pp. 14598–14607. doi:10.1109/ICCV48922.2021.01435. 2

[WJC*23] WANG C., JIANG R., CHAI M., HE M., CHEN D., LIAO J.: Nerf-art: Text-driven neural radiance fields stylization. *IEEE Transactions on Visualization and Computer Graphics* (2023). Publisher: IEEE. 2

[WXZ*20] WANG W., XU J., ZHANG L., WANG Y., LIU J.: Consistent Video Style Transfer via Compound Regularization. *Proceedings of the AAAI Conference on Artificial Intelligence 34*, 07 (Apr. 2020), 12233–12240. doi:10.1609/aaai.v34i07.6905. 2

[WYXL20] WANG W., YANG S., XU J., LIU J.: Consistent Video Style Transfer via Relaxation and Regularization. *IEEE Transactions on Image Processing 29* (2020), 9125–9139. doi:10.1109/TIP.2020.3024018. 2

[WZDB22] WU Z., ZHU Z., DU J., BAI X.: CCPL: Contrastive Coherence Preserving Loss for Versatile Style Transfer. In *European Conference on Computer Vision* (2022), Springer, pp. 189–206. 7, 8, 9, 10

[XHKK21] XIAN W., HUANG J.-B., KOPF J., KIM C.: Space-time neural irradiance fields for free-viewpoint video. In *CVPR* (2021). 2

[XLSL23] XU S., LI L., SHEN L., LIAN Z.: DeSRF: Deformable Stylized Radiance Field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), pp. 709–718. 2

[YYZ*23] YANG G., YANG S., ZHANG J. Z., MANCHESTER Z., RAMANAN D.: Physically Plausible Reconstruction from Monocular Videos. In *ICCV* (2023). 5

[ZHX*23] ZHANG Y., HE Z., XING J., YAO X., JIA J.: Ref-npr: Reference-based non-photorealistic radiance fields for controllable scene stylization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), pp. 4242–4251. 3

[ZKB*22] ZHANG K., KOLKIN N., BI S., LUAN F., XU Z., SHECHTMAN E., SNAVELY N.: Arf: Artistic radiance fields. In *European Conference on Computer Vision* (2022), Springer, pp. 717–733. 2, 5, 6, 7, 9, 10