*Research Article*

# Video style transfer via implicit 4D representation

**Yinji ShenTu · Tianrun Chen · Bojian Wu**

Received: date / Accepted: date

**Abstract** Video style transfer is an important area of research in computer vision with many potential applications in the fields of art, entertainment, and video production. While previous methods for video style transfer relied on constraints such as optical flow to maintain temporal coherence, they lacked the desired level of generalizability. To address this limitation, we propose a novel video style transfer approach based on an implicit 4D representation, which enables us to preserve the style, temporal dynamics, and spatial information of the input video. Unlike previous methods that extended 2D image style transfer techniques to videos, our approach decouples temporal and spatial consistency into a distinct 4D reconstruction task, followed by the stylization of the scene appearance. This extension builds upon the foundation of 4D reconstruction techniques from computer graphics and rendering. Importantly, Decoupling the 4D representation brings additional benefits, including support for style transfer for a single training view video and facilitating view synthesis to ensure consistent style across different viewpoints, which we believe will bring new possibilities for future video production tools. By proposing a flexible framework instead of a model, we can easily replace the 4D model to get better results, which is conducive to subsequent work and updates.

**Keywords** Video style transfer · Neural radiance field · Neural Rendering · Dynamic implicit 3D representation · Neural Graphic Primitive

## 1 Introduction

Editing digital content has always been an intriguing problem. When it comes to image editing, comprehensive tools like PhotoShop provide users with the ability to make almost any desired changes to an image. However, video editing, on the other hand, is not as commonly demanded, resulting in video editing tools being less advanced compared to their image editing counterparts. Consequently, video editing remains a challenging endeavor. There are two primary challenges associated with video editing. Firstly, maintaining temporal consistency across all frames is crucial, which is not a concern in image editing. For instance, if an object moves within a video, we must ensure that the edited object maintains consistent positioning across all frames, even when occlusions occur. Secondly, the editing process should be intuitive, a challenge that has already been addressed in image editing. Manually editing each frame of a video is a non-intuitive approach that demands extensive expertise and a significant amount of effort, representing the traditional method employed by video editing tools. Ideally, we should be able to automatically propagate edited content throughout the entire video by simply editing a single frame, akin to the simplicity of image editing. The question then arises: How can we overcome these aforementioned challenges?

The aforementioned challenges can be addressed through both 2D and 3D approaches. In 2D approaches, users modify keyframes, and these modifications are propagated throughout the entire video using frame-to-frame tracking or feature similarity [1]. Alternatively,

All Authors are with Zhejiang University, Hangzhou, CHINA. (Email: styj549194748@gmail.com, tianrun.chen@zju.edu.cn, ustcbjwu@gmail.com).
Corresponding author: Bojian Wu.

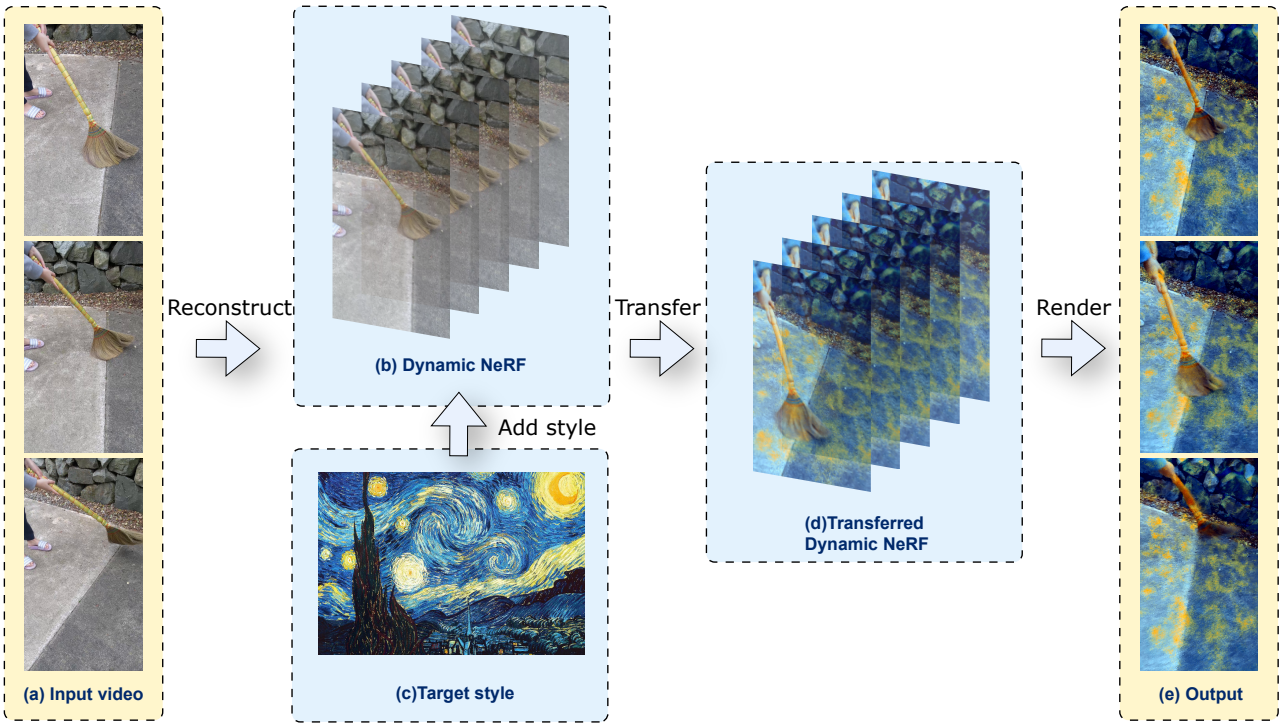



**Fig. 1** Our approach aims to perform style transfer on the visual appearance of a dynamic 3D NeRF scene by considering spatial and temporal information, rather than directly transferring the rendered image without such awareness. In order to achieve this, we utilize training videos (a) to reconstruct a dynamic NeRF representation (b) of the 4D scene. Subsequently, we employ an optimization-based style transfer process to generate a newly transferred dynamic NeRF (d). Finally, we are able to render novel style transferred results at novel views and times as shown in (e).

recent studies have tackled this issue through layer-based methods [2], which have made significant progress in ensuring consistency. On the other hand, 3D approaches typically rely on strong 3D priors or assume static scenes [3,4]. However, these methods only apply to local parts of the scene. In 2D approaches, it is necessary to distinguish between foreground and background, while 3D representations require strong 3D priors or are limited to static scenes. The task becomes more challenging if the goal is global editing, such as style transfer as presented in this paper.

To address this, we propose a novel video editing pipeline based on a 4D implicit representation. Unlike 3D representations, our approach incorporates the time variable into the model, enabling global editing of dynamic scenes. Similar to NeRF [5], we simplified HyperNeRF [6] as our baseline, which employs MLPs to model the time variable. It is important to note that HyperNeRF can be substituted with any superior model within our proposed pipeline to achieve enhanced editing effects. The advantage of our method lies in the 4D representation's ability to intuitively capture the entire video scene, enabling comprehensive content alterations through intuitive editing. Furthermore, we offer edited videos generated from different viewpoints, providing additional spatial consistency constraints. Our method contributes in the following ways: 1) We introduce a video editing pipeline based on a 4D representation, effectively addressing the challenges of temporal consistency and intuitive editing. 2) Our pipeline enables global style editing, a capability lacking in previous methods that focused on dynamic 3D representations and scene editing. 3) Our pipeline incorporates additional spatial consistency constraints and facilitates the synthesis of novel views and time images.

## 2 Related work

**Video Style Transfer** The application of deep learning in computer vision has led to a significant body of research dedicated to video style transfer in recent years. Early studies primarily focused on transferring styles between images. The pioneering work of Gatys et al. [7–9] introduced neural network-based methods for image style transfer, which subsequently became widely adopted. Building upon this foundation, Johnson et al. [10] proposed perceptual loss to enhance the learning of high-level features and address real-time optimization challenges. The introduction of cycleGAN by

Zhu et al. [11] garnered substantial attention, prompting researchers to explore its application in video data. For instance, Wang et al. [12] pioneered video-to-video synthesis by applying cycleGAN to video data, coining it as video-video synthesis. However, despite its initial promise, cycleGAN exhibited limitations, resulting in noticeable temporal inconsistencies, such as flickering effects. Subsequent research efforts primarily aimed to mitigate this issue by incorporating additional constraints, including optical flow [13,14], depth [15], and illumination [16]. Nonetheless, constraints such as optical flow are significantly influenced by occlusion and varying illumination conditions. To address this challenge, Wu et al. [17] proposed a method that preserves temporal consistency using the structural similarity (SSIM) loss. However, obtaining spatial information through 2D video representation remains challenging, as it inherently fails to account for occlusion and the resulting incoherence.

**NeRF** In recent advancements, the neural radiance field (NeRF) [5] has emerged as a highly successful technique for novel view synthesis, surpassing traditional rendering methods in terms of effectiveness. NeRF enables the creation of a 3D representation of a scene using a sequence of scene images, which can then be utilized to generate images from arbitrary viewpoints through ray marching. Several studies [6, 18–22] have extended the capabilities of NeRF to dynamic scenes, yielding impressive outcomes. Noteworthy examples include NSFF [23], which incorporates flow supervision to capture scene motion, DyNeRF [24], which employs time-conditional neural radiance field to ensure temporal consistency, and D-NeRF [25], Nerfies [19], and HyperNeRF [6], which leverage canonical space and a series of deformation fields to represent dynamic scenes. While these investigations present intriguing findings, there is currently a dearth of research exploring editing and style transfer specifically tailored to dynamic scenes.

**NeRF based editing** In addition to the success of NeRF in novel view synthesis, there have been endeavors to apply NeRF to scene editing. For instance, NeuMesh [26] enables precise appearance editing within NeRF scenes. Additionally, a set of inverse rendering studies [27–29] explores physically-based relighting and material editing by recovering object materials and environmental illumination. In terms of style transfer, a series of NeRF-based investigations [30–33] have emerged, allowing for style modification within NeRF scenes through adjustments in 2D pre-trained networks. These works primarily employ optimization methods to ensure similarity between the generated RGB image and the target style image. However, it is important to note that all the aforementioned editing and style transfer studies are focused on static scenes, with no integration of dynamic scenes.

## 3 Method

### 3.1 Overview

Our method takes a video and a style transfer target image as input and outputs a stylized free-view video. Specifically, we build a NeRF model that can represent dynamic scenes as our basic representation. The neural radiance field models the scene implicitly as a continuous and differentiable 5D function base on volume rendering, which is parameterized by a multi-layer perception (MLP). It can be defined as: $F(\mathbf{x} \in \mathbb{R}^3, \mathbf{d} \in \mathbb{S}^2) \rightarrow (\sigma \in \mathbb{R}^+, \mathbf{c} \in \mathbb{R}^3)$ where $\mathbf{x} \in \mathbb{R}^3$ represents the position $(x, y, z)$, $\mathbf{d} \in \mathbb{S}^2$ represents the direction $(\theta, \phi)$, $\sigma$ represents the density, and $\mathbf{c}$ represents the color.

As for dynamic NeRF, which has a more complicated structure, it has to encode the time as a conditional variance $\mathbf{t}$ to the model. It can also be viewed as extending NeRF to a higher dimensional space. And our baseline is to map all the coordinates $\mathbf{x}$ of the observation space to the canonical space. Then we use the optimization method to stylize the entire model, thereby obtaining a stylized free-view video. In summary, our work is a 2-stage work. The first stage is to build a 4D model of a dynamic scene, and the second stage is to perform style transfer on this model. The overall pipeline is shown in Figure 2 We will introduce how we build a dynamic scene model in the next section 3.2, and how we perform style transfer in the following section 3.3. And there is also some other helpful designs shall be introduced in other sections 3.3 As shown in the following formula, where $\mathbf{c}'$ represents the color after style transfer, and **style** represents the style transfer image.

$$\begin{aligned} F : \left(\mathbf{x} \in \mathbb{R}^3, t \in \mathbb{R}, \mathbf{d} \in \mathbb{S}^2\right) \\ \mapsto \left(\sigma \in \mathbb{R}^+, \mathbf{c} \in \mathbb{R}^3\right) \\ \overset{\textbf{style}}{\mapsto} \left(\sigma \in \mathbb{R}^+, \mathbf{c}' \in \mathbb{R}^3\right). \end{aligned} \tag{1}$$

### 3.2 Video to NeRF

Most dynamic NeRF [18, 20–22] based on multi-view video input to build a dynamic scene model. However, some works [6,19,34] have made normal but not fix-view video input enough to build a dynamic scene model. In our method, we choose to simplify the dynamic scene model based on HyperNeRF [6]. Because compared with

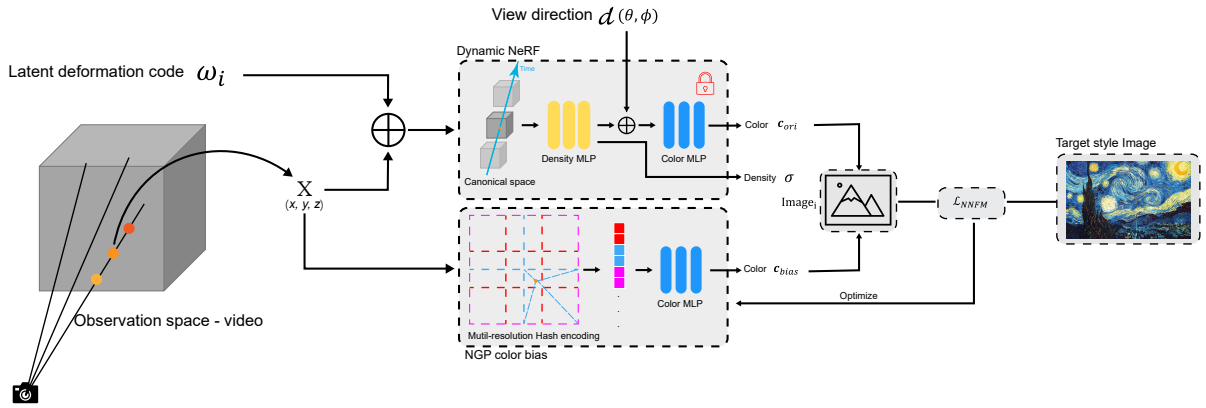



**Fig. 2** In this pipeline we first assume we have already obtained the well-reconstruct dynamic NeRF since the reconstruction from a video is not the main task of this paper.(1) First we encode the Time $t_i$ to deformation code $\omega_\mathbf{i}$ and use the dynamic NeRF to render the corresponding $Image_i$ . (2) At the same time, we add a Neural Graphic Primitive (NGP) to as the part that to be optimized for style transfer, where the Multi-resolution Hash encoding only encode the position X *(x,y,z)* and pass the feature to the MLP to get the bias color. (3) Then we add the bias to the rendered $Image_i$ . (4) Finally, we calculate the NNFM loss between $Image_i$ and the target style image and optimize iteratively until we get the style transfer result we want.

other methods, although they all use a deformation field to encode the time variable, the HyperNeRF structure has the advantage that it does not separate static and dynamic scenes like other methods. This advantage ensures we can better keep the consistency of the appearance when we perform style transfer, and avoid some boundary problems between static and dynamic scenes. But for the video input, we will definitely make certain restrictions, such as the position of the camera must not be fixed, otherwise the entire scene cannot be modeled well. After all, there is no satisfying general method for reconstructing a single-view fixed camera pose till the moment we set the experiment.

Many previous methods [35–38] for processing video information may focus on some constraints such as optical flow and depth estimation. Then these methods gain the information, like depth, from distilling spatial information, rather than a very intuitive representation. While optical flow gain from image matching containing motion information, is also an important representation strongly related to time information. Neither of them is a very intuitive representation of the entire scene information, but an approximation. We have to admit that optical flow is practical and effective for local video editing, but it is not really useful for global video editing, like style transfer. So we thought of using a 4D scene model to represent video information directly. In this way, we can also perform style transfer on the appearance of the entire scene, rather than style transfer on the rendered images of each frame. Make it keep both spatial and temporal consistency.

Our 4D scene model is based on the volume rendering method. It means we regard the scene as the volume density and directional emitted radiance at any point in space. We render the color if any ray passes through the scene using the principle from classical volume rendering [39]. As the following volume rendering equation 2 shows, we can get the color of the ray by integrating the radiance along the ray.

$$\begin{aligned} \mathbf{C}(\mathbf{r}) &= \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt \\ \text{where } T(t) &= \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(t'))dt'\right). \end{aligned} \tag{2}$$

The $\mathbf{r}$ represents a ray in the radiance field and $\mathbf{C}(\mathbf{r})$ is the expected color of the ray. For the ray function $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ with near and far bounds $t_n$ and $t_f$, the transmittance $T(t)$ is the accumulated opacity along the ray. The $\sigma(\mathbf{r}(t))$ is the density of the scene at the point $\mathbf{r}(t)$ and $\mathbf{c}(\mathbf{r}(t), \mathbf{d})$ is the emitted radiance at the point $\mathbf{r}(t)$ in the direction $\mathbf{d}$. Although the origin volume rendering equation is continuous, NeRF [5] discretizes the equation and uses a multi-layer perceptron (MLP) to approximate the density and the emitted radiance. By using a stratified sampling strategy that partition$[t_n, t_f]$ into N samples, the formula can be changed into discretized form as follows:

$$\begin{aligned} \mathbf{C}(\mathbf{r}) &= \sum_{i=1}^{N} T_i(1 - \exp\left(-\sigma_i\delta_i\right))\mathbf{c}_i \\ \text{where } T_i &= \sum_{j=1}^{i-1} \exp\left(-\sigma_j\delta_j\right) \end{aligned} \tag{3}$$

where $\delta_i = t_{i+1} - t_i$ is the distance between adjacent samples and this will reduce the traditional alpha composition with $\alpha_i = 1 - \exp(-\sigma_i \delta_i)$

It should be noted that NeRF does not directly receive input $\mathbf{x}$, but passes $\mathbf{x}$ through a sinusoidal position encoding for better learning at high-frequency features according to Tancik et al. [40]. The position encoding formula:

$$\gamma(x) = \left[\sin(x), \cos(x), ..., \sin(2^{m-1}x), \cos(2^{m-1}x)\right] \quad (4)$$

where $m$ is a hyper-parameter that controls the number of sinusoids for encoding. The parameter $m$ control the smoothness of the learned representation by modifying an interpolating kernel effective bandwidth.

As for the time variable, we implement embedding to it, which is a common method in natural language processing tasks. We encode time $t_i$ as the latent deformation code $\omega_i$, and concatenate it with the input $\mathbf{x}$ as input to dynamic NeRF. The detail of the canonical space is originally represented by a deformation field and ambient field. Both are MLPs while all the specific structures will be introduced in the implementation details.

At the training stage, we must also specify the training loss we need. We choose L2 loss as our training loss.

$$\mathcal{L}_{L2} = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{C}(\mathbf{r}_i) - \mathbf{C}_{gt}(\mathbf{r}_i)\|_2^2 \quad (5)$$

where $\mathbf{C}(\mathbf{r}_i)$ is the color of the ray $\mathbf{r}_i$ and $\mathbf{C}_{gt}(\mathbf{r}_i)$ is the ground truth color of the ray $\mathbf{r}_i$. And since all the computing is based on discrete volume rendering 3, which calculates a ray as a unit. It is not memory efficient to calculate the loss with image size resolution like other 2D vision tasks. But for the style transfer task, to obtain some global features, we have to align the loss calculation and feature extraction in image size resolution. This challenge will be introduced more in the following section.

### 3.3 Style transfer

Style transfer is a very classic computer vision task. Since NeRF is a method that is highly related close to 2D computer vision compared to classical computer graphics, some work about style transfer has been done as we introduced in related work. However, till our experiment, we didn't find any other work focused on dynamic NeRF-style transfer. Thus our main work and experiment are addressing this problem. According to formula (1), we can know that NeRF, which base volume rendering, will mainly output two parts: density $\sigma$ and color $\mathbf{c}$. Our method is to perform style transfer on the color part, which is consistent with the static scene.

Specifically, we choose a method similar to the optimization method of ARF [30], because this work has proved the effectiveness of this method on static scenes. However, the nice result of previous work is based on TensoRF [41], which decouples the scene with an appearance plane that is very suitable for feature-related back-propagation. But from the experiment. we find that directly using the plane decouple method on dynamic NeRF will cause the scene to be distorted and not work for our task.

**Neural graphic primitives(NGP)** So it is very intuitive that we want to optimize the color MLP to achieve style transfer. Although direct optimization of color MLP works in changing the appearance, it is far away from the results by static results. This may due to the MLP may hard to learn the local feature of the image without some specific encoder, which is very important for style transfer. The encoder in the static case is TensorVM decouple and convolution patch in vision tasks, but in the dynamic models, we don't have such a method yet. Thus we would like to find an encoder that makes the model. We choose the Multiresolution hash encoding method introduced in Instant neural graphics primitives with a multiresolution hash Encoding [42]

As introduced in the paper [42], computer graphics primitives are fundamentally represented by mathematical functions, which is a very natural way to represent the world and its quality and performance are vital for visual effects. To achieve good neural graphic primitives, multiresolution hash encoding is proposed, which is independent of the task, adaptive and efficient. The key point is that at the volume rendering equation, we assume the scene has a discrete grid and voxel, which may lead to a lack of local appearance and high-frequency features. The core idea of multiresolution hash encoding is to map a cascade of grids to corresponding fixed-size arrays of feature vectors, while there are hash tables for different fine resolutions. For the latent hash collision risk, the hash table can automatically prioritize the most important features and collision will not happen since the hash table only needs to store those points that have the largest gradients. Unlike prior work, like position encoding, no structure will be updated to the data structure during training.

**Table 1** Multiresolution Hash encoding parameters

| Parameter | Value | Description |
|---|---|---|
| $L$ | 16 | Number of levels |
| $T$ | 19 | Hash table entries per level(log2) |
| $F$ | 2 | Dimensionality of feature vectors |
| $N_{min}$ | 64 | Coarsest resolution |

Multiresolution hash encoding is effective but not memory efficient. So there is a trade-off between the memory and performance. Higher values of $T$ result in higher quality and lower performance. Normally better performance always needs more memory. And since the original work is also accelerated by CUDA implementation, we will not consider this trade-off in our work.

Normally we use MLPs to represent the radiance field, the fully connected neural network $m(y;\phi)$, and the encoding function is position encoding $y = enc(x)$. We can see there are no trainable parameters in the encoding function, which is not suitable for our task. The multiresolution hash encoding $y = (x;\theta)$ provides trainable parameters $\theta$ for the encoding function. The multiresolution hash tables are arranged into $L$ levels, each con-training up to $T$ feature vectors with dimensional $F$. Above mentioned parameters and others' default values are shown in Table 2. For each level resolutions(the different color cubes in the pipeline 2), the levels independently store feature vectors at the vertices of a grid, the resolutions are chosen between coarsest and finest resolution $[N_{min}, N_{max}]$, where $N_{max}$ is chosen to match the finest detail in the training data and not be needed to be specific in the tiny-cuda-nn [43] implementation:

$$N_l = \lfloor N_{min} \cdot b^l \rfloor, b = exp\left(\frac{log\left(N_{max}/N_{min}\right)}{L-1}\right) \quad (6)$$

In summary, we try to use the NGP to encode the position of the scene, which is a grid or voxel in the volume rendering equation. Meanwhile, the encoding has better feature representation ability than the position encoding, which is very important for style transfer. So as we have shown in the pipeline, instead of adding NGP in the training phase, we add it in the style transfer phase. It is more flexible for other dynamic 3D scene representations or normally called better generalization.

**Optimization Objective** The optimization objective of our method is to minimize the following formula:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{content}} + \lambda \mathcal{L}_{\text{style}} \quad (7)$$

where the total loss is combined with content loss and style loss with a weight $\lambda$ that balances the content preservation and the stylization effect. This kind of loss is proposed since the very beginning of style transfer work [8,9]. And this structure will also lead to hard convergence problems, like adversarial training. Due to the style transfer can not be well quantified and evaluated, we need to monitor the results while optimizing.

The style of 2D images is actually not a very explicit feature in many cases, especially in some artworks. So we choose to use the pre-trained VGG19 [44] network to extract style features just like previous work. But we don't use the output of VGG19 directly, but the output of some intermediate layers. Usually, the gram matrix is used to compute the style loss in 2D style transfer tasks. However, according to the work of ARF, we found that the effect of using the gram matrix on dynamic scenes is not as good as NNFM(Nearest Neighbor Feature Matching) Loss.

$$\mathcal{L}_{nnfm}(\mathbf{F}_{\text{style}}, \mathbf{F}_{\text{render}}) = \frac{1}{N}\sum_{i,j} \min_{i',j'} D(\mathbf{F}_{\text{style}}(i,j), \mathbf{F}_{\text{render}}(i',j')) \quad (8)$$

where $\mathbf{F}_{\text{style, render}}$ means the feature of style target and rendered image extract by pre-trained models, $i, j$ means the pixel position at the rendered image,$N$ comes from the pixel number region to be transfer and $D(\boldsymbol{\alpha}, \boldsymbol{\beta})$ is the cosine distance between two vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$.

$$D(\boldsymbol{\alpha}, \boldsymbol{\beta}) = 1 - \frac{\boldsymbol{\alpha} \cdot \boldsymbol{\beta}}{\|\boldsymbol{\alpha}\| \, \|\boldsymbol{\beta}\|} \quad (9)$$

For each feature in $F_{render}$, we minimize the cosine distance 9 between it and the nearest feature in $F_{style}$.

**Deferred back-propagation** All the above-mentioned loss functions are computed in the image space, which is not friendly to the radiance field when we considered GPU memory and computation cost. So we use the strategy called deferred back-propagation. It is a very simple but effective strategy to reduce the memory cost. The basic idea is to break the gradient auto back-propagation into two stages–the image stage and the radiance field stage. The key trade-off is that our GPU cannot store all the rays belonging to one rendered image and their corresponding gradients in memory at the same time. And the NeRF-related method normally processes the back-propagation in the ray unit instead of the image unit. However, since NeRF is an algorithm that is highly related to computer vision, we can not avoid the image unit back-propagation. Thus deferred back-propagation is a good trade-off to reduce the memory cost when we face vision-related gradients processing. As we can see in the 3, we need to disable the auto back-propagation when rendering images. Then we compute the loss and obtain the gradients of each pixel in the rendered image. Finally, we can use the gradient of each pixel to back-propagate to the radiance field patch-wise.

**Match Color** We found that a large amount of color information can be achieved by zero-shot transformation of the RGB space through the reproduction of ARF experiments. This pre-processing can significantly improve the style transfer quality with nearly no extra cost. We apply this method to the rendered image before loss calculation and inference after style transfer. Color transfer is a traditional task that has been addressed since 2001 by Reinhard et al [45]. The characteristics of

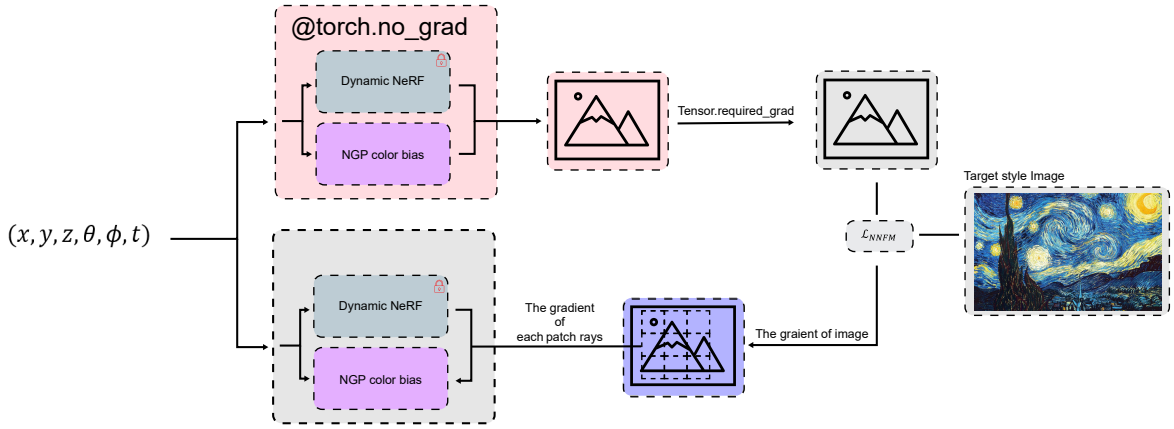



**Fig. 3** This is the illustration of the deferred back-propagation. Please notice that the pink block means the computing in these process is gradients free. The grey block means those tensor computed in these blocks has gradient function and need back-propagation. The blue block means it already contains the descended gradients. The red lock for dynamic NeRF means we do the gradients descended for it but without updating its parameters.

each channel in the color space are independent of each other. According to the statistical analysis of the colored image, a linear transformation is determined, so that the target image and the original image are in $l, \alpha, \beta$ spaces have the same mean and variance. Mathematically, the color transfer can be expressed as:

$$\begin{cases} E\left[\mathbf{Ac}\right] = E\left[\mathbf{s}\right] \\ Cov\left[\mathbf{Ac}\right] = Cov\left[\mathbf{s}\right] \end{cases} \tag{10}$$

Where $E, Cov$ means the mean and covariance of the image, $\mathbf{A}$ is the transformation matrix, $\mathbf{c}$ is the color of the original image, $\mathbf{s}$ is the color of the target image. We just need to solve the linear transformation matrix $\mathbf{A}$ to achieve the color transfer.

# 4 Experiment

## 4.1 Experiment Setup

We conduct our experiments on the datasets of Nerfies [19] and other datasets from HyperNeRF [6]. Since there is no previous work that uses a similar framework as we do, we compare our complete method with some other approaches we have tried. For example, directly optimizing the color MLP in the dynamic NeRF, it does keep better consistency but is hugely restricted by the training view, and the style transfer is highly constrained by the geometry of the scene. Even though we have tried to use TensoVM encoding, where implementation from nerfstudio [46], the results are still not satisfactory, even some details cannot learn like the static scenes.

**Implementation details** All experiments are conducted on a single RTX3090. The environment is based on CUDA 11.1 with its corresponding version pytorch and other package related. Tiny-cuda-nn [43] is used to quickly implement some encodings. The Multiresolution hash encoding-related parameters are introduced in Table 2. We use Adam optimizer with learning rate at 1e-3 and 1e-4 for the initial and the fine-tuning stage respectively. And 1e-3 as the default learning rate when we optimize NGP. The $\lambda$ in the loss function is set to 1 as the default.

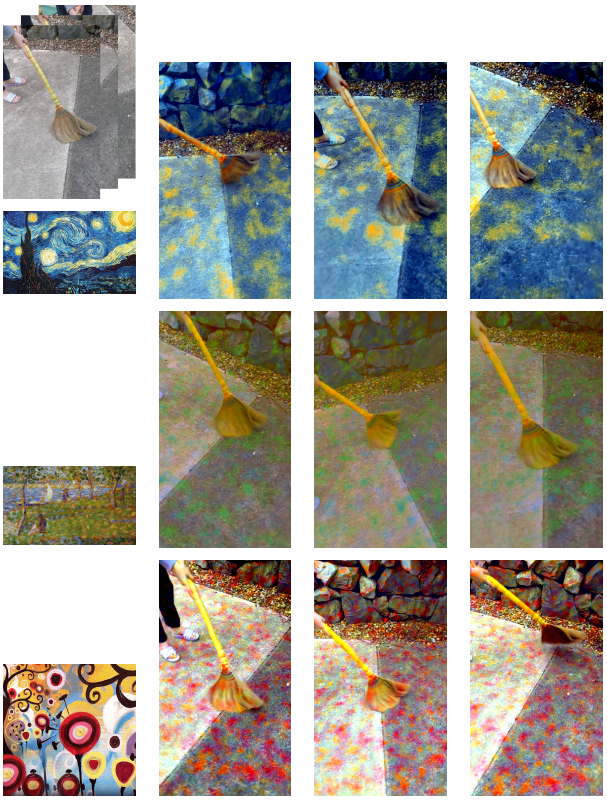
## 4.2 Results

All the results are with default parameters and later we would like to discuss about the ablation study.

## 4.3 Ablation Study

**Encoding methods** As we mentioned in Section 4.1, we have tried different encoding methods, including directly optimizing the color MLP and the TensoVM encoding, but the results are not satisfactory. color MLP is base position encoding. And for the TensoVM encoding, the results show it does not work well. The comparison is shown in the following figure.

**NGP base resolution** During our experiments, we found that the base resolution of the NGP is important for the final results. While the smaller base resolution would likely gain smoother features and the larger base resolution would likely gain more details with the storage



(a) Inputs (b) View1 (c) View2 (d) View3

**Fig. 4** Here are some style transfer result on novel views, the (a) Input includes the training video and target style and (b,c,d) are different novel view and time results we rendered from the transferred dynamic NeRF

(a) NGP (b) color MLP (c) TensoVM

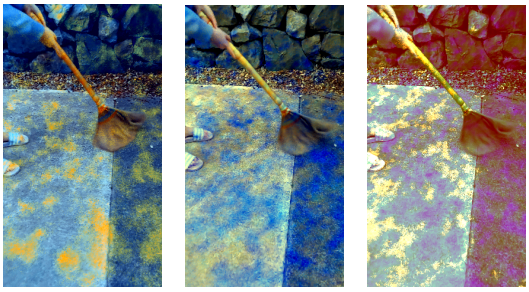
**Fig. 5** Please notice we choose the results according to the NNFM loss, those results are nearly has the best performance in their own optimization process measured by NNFM loss. So it may not with same view and time.

cost increase. We have tested the base resolution in 32, 64, and 128, and taken 64 as the default value. The comparison is shown in the following figure.

**Loss function** The loss function is also important for the final results. We have tried the gram loss function and different layer features in the NNFM loss feature extraction stages. The comparison of nnfm loss and gram loss will be shown in the following figure. The layers feature extracted didn't have as much difference as the results in ARF [30].

(a) $N_{min}$=64 (b) $N_{min}$=32 (c) $N_{min}$=128

**Fig. 6** As we can see, the $N_{min}$=64 has the most style like performance, and the $N_{min}$=128 has better details and $N_{min}$=32 is more smooth. In practice we can use different $N_{min}$ for different scenes.



(a) NNFM loss (b) gram loss (c) gram loss 2

**Fig. 7** First gram loss is work in some case as we shown above, but it is not generalizable enough for different style, we can see the results($c$) on another style is not competitive as the former one

**Color transfer** From the results we did in the following ablation study, color Transfer hugely reduces the learning difficulty making the distance in the color space much smaller. There is one more reason that color transfer is vital in our pipeline we tried to define the style as a color bias. The ability to represent style features is highly related to the original color space.

(a) w ColorTransfer (b) w/o ColorTransfer

**Fig. 8** We can see without color transfer the color bias is hard to be learned rightly.

## 5 Conclusion

Video style transfer has been extensively explored, yielding effective solutions. However, approaching this task from a graphics perspective rather than a vision introduces intriguing possibilities. In this study, we propose a novel method for video style transfer based on dynamic neural radiance fields. Our approach aims to preserve the global appearance of 3D scenes while retaining the original geometry and its deformations, ensuring spatial and temporal consistency in the generated video. Moreover, our method can be readily tested on diverse datasets, as it is a general approach applicable to various data distributions. Nevertheless, our method has certain limitations. A key constraint is a strong emphasis on preserving voxel density corresponding to the scene geometry. As a result, the color output of some dynamic scene NeRF methods may not precisely align with our desired global appearance. Since appearance and density are intertwined within implicit representations, our method may not achieve the same level of effectiveness in video style transfer as previous approaches. For instance, in our experiments showcasing *StarryNight*, while we successfully capture the color and global features, we encounter challenges in accurately learning the stroke texture due to inherent geometry limitations. Consequently, the results may resemble graffiti on the scene, rather than a faithful representation of the target style. Nonetheless, our method represents a valuable attempt, and we anticipate that advancements in implicit dynamic representation techniques will address this issue, much like the progress made in resolving similar challenges faced by NeRF methods in explicit static scenes through approaches like TensoRF [47]. Additionally, local editing of dynamic scenes poses a highly complex task that warrants further exploration, and we look forward to future endeavors tackling this challenge.

## 6 List of abbreviation

**Table 2** List of abbreviations

| Abbreviations | Definitions |
|---|---|
| MLP | Multi-Layer Perception |
| NeRF | Neural radiance field |
| 3/4D | three or four dimensions |
| NGP | Neural Graphic Primitive |
| NNFM | Nearest Neighbor Feature Matching |
| CUDA | Compute Unified Device Architecture |
| HyperNeRF | A Higher-Dimensional Representation for Topologically Varying Neural Radiance Fields |
| ARF | Artistic radiance field |

## 7 Declaration

**Availability of data and material** The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

**Competing interests** The authors have no relevant financial or non-financial interests to disclose.

**Author contributions** All authors contributed to the study conception and design. Material preparation, data collection, and analysis were performed by Yinji ShenTu. The first draft of the manuscript was written by Yinji ShenTu and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

**Funding** This work is not supported by any funding.

**Acknowledgement** We thank all authors and others who provided suggestions and help during this work.

## References

1. Allan Jabri, Andrew Owens, and Alexei Efros. Space-time correspondence as a contrastive random walk. *Advances in neural information processing systems*, 33:19545–19560, 2020.
2. Yoni Kasten, Dolev Ofri, Oliver Wang, and Tali Dekel. Layered neural atlases for consistent video editing. *TOG*, 2021.
3. Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. Consistent video depth estimation. *ACM Transactions on Graphics (ToG)*, 39(4):71–1, 2020.
4. Fanbo Xiang, Zexiang Xu, Milos Hasan, Yannick Hold-Geoffroy, Kalyan Sunkavalli, and Hao Su. Neutex: Neural texture mapping for volumetric neural rendering. In *CVPR*, 2021.

5. Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.
6. Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *TOG*, 2021.
7. Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.
8. Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.
9. Leon A Gatys, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. Preserving color in neural artistic style transfer. *arXiv preprint arXiv:1606.05897*, 2016.
10. Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 694–711. Springer, 2016.
11. Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.
12. Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. *arXiv preprint arXiv:1808.06601*, 2018.
13. Yang Chen, Yingwei Pan, Ting Yao, Xinmei Tian, and Tao Mei. Mocycle-gan: Unpaired video-to-video translation. In *Proceedings of the 27th ACM international conference on multimedia*, pages 647–655, 2019.
14. Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. Artistic style transfer for videos. In *GCPR*, 2016.
15. Songhua Liu, Hao Wu, Shoutong Luo, and Zhengxing Sun. Stable video style transfer based on partial convolution with depth-aware supervision. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 2445–2453, 2020.
16. Yingying Deng, Fan Tang, Weiming Dong, Haibin Huang, Chongyang Ma, and Changsheng Xu. Arbitrary video style transfer via multi-channel correlation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1210–1217, 2021.
17. Xinxiao Wu and Jialu Chen. Preserving global and local temporal consistency for arbitrary video style transfer. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 1791–1799, 2020.
18. Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. 11 2020.
19. Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *ICCV*, 2021.
20. Sida Peng, Yunzhi Yan, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Representing volumetric videos as dynamic mlp maps. In *CVPR*, 2023.
21. Haotong Lin, Sida Peng, Zhen Xu, Yunzhi Yan, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Efficient neural radiance fields with learned depth-guided sampling. In *SIGGRAPH Asia Conference Proceedings*, 2022.
22. Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *ICCV*, 2021.
23. Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*, 2021.
24. Lingjie Liu, Marc Habermann, Viktor Rudnev, Kripasindhu Sarkar, Jiatao Gu, and Christian Theobalt. Neural actor: Neural free-view synthesis of human actors with pose control. *TOG*, 2021.
25. Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *CVPR*, 2020.
26. Bangbang Yang, Chong Bao, Junyi Zeng, Hujun Bao, Yinda Zhang, Zhaopeng Cui, and Guofeng Zhang. Neumesh: Learning disentangled neural mesh-based implicit field for geometry and texture editing. In *ECCV*, 2022.
27. Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *TOG*, 2021.
28. Yuanqing Zhang, Jiaming Sun, Xingyi He, Huan Fu, Rongfei Jia, and Xiaowei Zhou. Modeling indirect illumination for inverse rendering. In *CVPR*, 2022.
29. Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. Physg: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *CVPR*, 2021.
30. Kai Zhang, Nick Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. Arf: Artistic radiance fields. In *ECCV*, 2022.
31. Yi-Hua Huang, Yue He, Yu-Jie Yuan, Yu-Kun Lai, and Lin Gao. Stylizednerf: consistent 3d scene stylization as stylized nerf via 2d-3d mutual learning. In *CVPR*, 2022.
32. Thu Nguyen-Phuoc, Feng Liu, and Lei Xiao. Snerf: stylized neural implicit representations for 3d scenes. *arXiv*, 2022.
33. Kunhao Liu, Fangneng Zhan, Yiwen Chen, Jiahui Zhang, Yingchen Yu, Abdulmotaleb El Saddik, Shijian Lu, and Eric Xing. Stylerf: Zero-shot 3d style transfer of neural radiance fields. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.
34. Hongrui Cai, Wanquan Feng, Xuetao Feng, Yan Wang, and Juyong Zhang. Neural surface reconstruction of dynamic scenes with monocular rgb-d camera. In *Thirty-sixth Conference on Neural Information Processing Systems (NeurIPS)*, 2022.
35. Simon Meister, Junhwa Hur, and Stefan Roth. UnFlow: unsupervised learning of optical flow with a bidirectional census loss. In *CVPR*, 2018.
36. Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-net: CNNs for optical flow using pyramid, warping, and cost volume. In *CVPR*, 2018.
37. Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazırbaş, and Vladlen Koltun. Flownet: Learning optical flow with convolutional networks. In *CVPR*, 2015.
38. Joon Son Chung and Andrew Zisserman. Lip reading using spatiotemporal convolutional networks. *arXiv preprint arXiv:1611.05358*, 2016.
39. James T Kajiya and Brian P Von Herzen. Ray tracing volume densities. *ACM SIGGRAPH computer graphics*, 18(3):165–174, 1984.

40. Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.
41. Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*, pages 333–350. Springer, 2022.
42. Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022.
43. Thomas Müller. tiny-cuda-nn, 4 2021.
44. Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
45. Erik Reinhard, Michael Adhikhmin, Bruce Gooch, and Peter Shirley. Color transfer between images. *IEEE Computer graphics and applications*, 21(5):34–41, 2001.
46. Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH '23, 2023.
47. Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision (ECCV)*, 2022.