StyleRF-VolVis: Style Transfer of Neural Radiance Fields for Expressive Volume Visualization

Kaiyuan Tang and Chaoli Wang

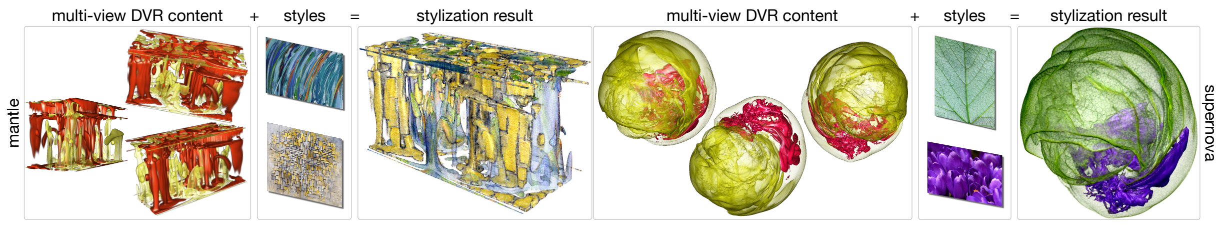


Fig. 1: Given a set of multi-view DVR images produced from a volumetric dataset, StyleRF-VolVis transfers arbitrary styles from reference images to the reconstructed 3D scene, synthesizing high-quality novel views with visual content consistency.

Abstract—In volume visualization, visualization synthesis has attracted much attention due to its ability to generate novel visualizations without following the conventional rendering pipeline. However, existing solutions based on generative adversarial networks often require many training images and take significant training time. Still, issues such as low quality, consistency, and flexibility persist. This paper introduces StyleRF-VolVis, an innovative style transfer framework for expressive volume visualization (VolVis) via neural radiance field (NeRF). The expressiveness of StyleRF-VolVis is upheld by its ability to accurately separate the underlying scene geometry (i.e., content) and color appearance (i.e., style), conveniently modify color, opacity, and lighting of the original rendering while maintaining visual content consistency across the views, and effectively transfer arbitrary styles from reference images to the reconstructed 3D scene. To achieve these, we design a base NeRF model for scene geometry extraction, a palette color network to classify regions of the radiance field for photorealistic editing, and an unrestricted color network to lift the color palette constraint via knowledge distillation for non-photorealistic editing. We demonstrate the superior quality, consistency, and flexibility of StyleRF-VolVis by experimenting with various volume rendering scenes and reference images and comparing StyleRF-VolVis against other image-based (AdaIN), video-based (ReReVST), and NeRF-based (ARF and SNeRF) style rendering solutions.

Index Terms—Style transfer, neural radiance field, knowledge distillation, volume visualization

1 Introduction

Since the early 1990s, volume visualization (VolVis) has been a central topic in scientific visualization research. One of the most popular techniques for VolVis is direct volume rendering (DVR). It works by casting rays from the image plane to the 3D volume, gathering samples along each ray, mapping them to visual quantities (i.e., colors and opacities) via transfer function (TF) lookups, and compositing the final color for each pixel in the rendering image. Thanks to the astonishing advances in graphics hardware, DVR can be efficiently implemented to achieve interactive framerates and high-quality visualizations, providing superior capability for users to explore volumetric datasets interactively. This practice has been the norm for the past two decades.

The recent surge of deep learning for scientific visualization research [55] has sprouted new opportunities for VolVis. Leveraging the capabilities of generative adversarial networks (GANs), one can train a network to synthesize rendering results under novel viewpoints, TFs, or other parameters, eliminating the need to access the original volumetric data and bypassing the conventional rendering pipeline [5, 23]. These seemingly impossible advances could have shocked many researchers just several years ago but are widely understood by the research community nowadays. After all, generative AI has swept across many fields, culminating in its extraordinary power to synthesize novel images and videos from text prompts and level the playing field for the masses.

• K. Tang and C. Wang are with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, USA. E-mail: {ktang2, chaoli.wang}@nd.edu.

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

Inspired by this fantastic trend, in this paper, we aim to design a deep learning framework to accomplish *expressive visualization* via *style transfer* for VolVis. Similar to prior work [5,17,21,23], we are supposed to be given a set of 2D images captured from the sample viewpoints and rendered with a fixed, reasonably good, yet unknown TF. Also, we assume no access to the 3D volumetric dataset during training or inference. We aspire to develop a new model that achieves expressive visualization by meeting the following goals (**G1** to **G3**) for end users. First, they can freely explore the DVR scene in excellent visual quality from previously unseen viewpoints (**G1**). Second, they can further flexibly recolor the rendering results *photo-realistically* by editing the underlying colors, opacities, and lighting effects (**G2**). Third, and most importantly, they can even intuitively modify the rendering results *non-photorealistically* by transferring styles (e.g., salient brushstrokes) from any image or painting of interest to the DVR scene (**G3**).

Realizing **G1** is relatively straightforward. Unlike GAN-based techniques, state-of-the-art solutions based on *neural radiance field* (NeRF) can deliver high-resolution, high-quality novel view synthesis results using a set of sample images. Nevertheless, significant challenges exist for achieving **G2** and **G3** due to the considerable gaps between the given 2D images and the 3D scene we aim to reconstruct and edit. For **G2**, the main challenge lies in accurately extracting the contributing color, opacity, and lighting information for faithful downstream edits to ensure visual content consistency. For **G3**, the grand challenge is to move beyond a fixed number of colors decoded from the original rendering images and adapt to abundantly rich color patterns or textures in the reference images.

We introduce StyleRF-VolVis, achieving style transfer of NeRFs for expressive VolVis. For large volumetric data, NeRF-based scene representations are space efficient and thus have implications for altering renderings in a compressive manner. At its core, we highlight three key components of StyleRF-VolVis to realize the stated goals (**G1** to **G3**).

First, we employ a *base NeRF model* to learn the density representation (i.e., content) of the 3D volumetric data from the collection of 2D training images (**G1**). This component accurately extracts the underlying scene geometry, paving the way for successful subsequent style editing. Second, we design a new *palette color network* that classifies regions of the radiance field (RF) and extracts a color palette (i.e., appearance) from multi-view images (**G2**). This component enables *photorealistic style editing* by modifying the original rendering's color, opacity, and lighting while maintaining visual content consistency across the views. Third, to remove the color palette constraint, we propose a novel *knowledge distillation* solution that transfers the color information from the palette color network (i.e., *teacher model*) to an *unrestricted color network* (i.e., *student model*) (**G3**). This component allows users to assimilate a wide spectrum of color patterns and textures from any reference image, supporting expressive visualization of the DVR scene via *non-photorealistic style editing*. Figure 1 highlights the capability of StyleRF-VolVis on two datasets, each using two styles extracted from different reference images. The contributions of our work are:

- We revisit style transfer for VolVis by presenting an innovative NeRF-based solution that supports expressive photorealistic and non-photorealistic style editing.
- StyleRF-VolVis represents a significant leap forward for visualization synthesis of volumetric data, advancing the state-of-the-art solutions in quality, consistency, and flexibility.
- We show the consistency and flexibility of StyleRF-VolVis on various combinations of DVR scenes and reference images.
- We compare StyleRF-VolVis with other image-based (AdaIN), video-based (ReReVST), and NeRF-based (ARF and SNeRF) style rendering solutions to demonstrate its superior quality.

2 Related Work

This section discusses related work on deep learning for VolVis, style transfer, NeRF for stylization, and knowledge distillation.

Deep learning for VolVis. Over the past few years, deep learning has emerged as a promising solution for improving the DVR process. One application is to use deep learning models to replace the traditional DVR pipeline. For instance, Berger et al. [5] constructed a GAN to synthesize volume-rendered images by investigating the image space of volume rendering under various TFs and viewing parameters. Hong et al. [23] proposed a GAN to synthesize high-resolution images from volume data under the desired rendering effect without knowing the TF. He et al. [21] developed InsituNet, a surrogate model that correlates simulation and visualization parameters with visualization outcomes, allowing users to preview visualization results under different simulation settings with a trained model. Shi et al. [46] built a view-dependent neural-network-latent-based surrogate model that supports producing high-resolution visualization results. Han and Wang [17] presented CoordNet, a coordinate-based neural network to synthesize rendering images under novel viewpoints given a set of DVR images for training. Another application employs a scene representation network (SRN) to represent volumetric data, minimizing disk storage requirements and enabling interactive neural rendering without direct access to volume data. For example, Weiss et al. [59] designed a dense-grid encoding method fV-SRN that directly renders from compressed representation with no additional memory for storing volume data during rendering. Wu et al. [60] leveraged multiresolution hash grid encoding and achieved fast volume encoding and real-time rendering. Wurster et al. [61] developed APMGSRN, an adaptively placed multi-grid SRN with a domain decomposition training approach for efficient VolVis. Other deep learning works in the context of VolVis focus on data or visualization generation [14, 15, 19, 51, 58], compression [13, 36, 37, 50], translation [20, 62], and completion [16]. Our work represents the first step in applying style transfer techniques to volume rendering results using a NeRF model. Compared with existing works [5, 17, 21, 23], StyleRF-VolVis takes a smaller set of DVR images for training and infers higher-quality results, maintaining content consistency across different viewpoints and supporting flexible style transfer.

Style transfer. Ever since Gatys et al. [11] demonstrated the capability of convolutional neural networks (CNN) in applying diverse artistic styles to natural images, neural style transfer [27] has emerged as a trending topic in both academia and industry. Numerous techniques have been developed to improve or extend the original algorithm. Johnson et al. [28] introduced a feed-forward network to solve the slow optimization problem and achieved real-time style transfer. Huang et al. [25] further maintained speed comparable to [28] without sacrificing the flexibility of transferring inputs to arbitrary new styles. This was realized using an adaptive instance normalization layer that aligns the statistics information of the content and style features. Ruder et al. [45] extended style transfer to video sequences by computing optical flow to achieve consistent style transfer across frames. Wang et al. [57] presented ReReVST that relaxes the objective function of style loss to make the transfer more robust to motions in content video.

In VolVis, Bruckner and Gröller [6] introduced a style TF using the lit sphere, which assigns the optical properties of voxel values with rendering styles instead of simple colors and opacities. However, extracting or designing the lit sphere requires artist involvement, and the rendering relies on a traditional pipeline. Our StyleRF-VolVis adopts an advanced NeRF representation, supporting end-to-end style transfer and rendering without accessing the original volume.

NeRF for stylization. Since the groundbreaking work of Mildenhall et al. [39] in 2020, NeRF has been widely used for novel view synthesis. We refer readers to recent survey papers [7, 53] to follow the roadmap of NeRF-based applications. While extensive studies [3, 10, 24, 29, 40] emphasize the quality or speed enhancement of NeRF, there is also a growing body of work focusing on editing a base NeRF to perform 3D style transfer of a scene. Generally, editing NeRF for style transfer can be classified into *photorealistic* and *non-photorealistic*. One main task for *photorealistic editing* is recoloring the scene. For example, Kuang et al. [32] and Gong et al. [12] utilized optimizable base colors in a palette to fit the scene and modify the base colors to recolor the scene during inference. Lee et al. [33] modified the essential weights in the color multilayer perceptron (MLP) of a trained NeRF to perform local recoloring. Unlike photorealistic editing, *non-photorealistic editing* transfers more abstract information, such as textures or brushstrokes of an artistic work. For instance, Chiang et al. [8] utilized a hypernetwork to transfer style features into the NeRF representation. Huang et al. [26] proposed a mutual learning strategy to integrate 2D stylization with NeRF geometry consistency. Zhang et al. [63] designed a nearest neighbor-based loss to stylize a trained NeRF, which performs better stylization quality compared with standard Gram matrix-based loss. Liu et al. [34] transformed the grid features within RFs to match a reference style. Even though the quality is less impressive than [63], it supports zero-shot transfer to an unseen reference image. StyleRF-VolVis differs from all these works in that it supports photorealistic and non-photorealistic editing in a single framework. We will show that conducting photorealistic editing before non-photorealistic editing yields better quality and flexibility for stylization results.

Knowledge distillation. Hinton et al. [22] introduced *knowledge distillation* (KD), which optimizes a small network to match the predictions of a large model. KD has been used in model compression and optimization [18, 22, 38]. In the NeRF space, Reiser et al. [44] leveraged a large global pre-trained NeRF to speed up the training of multiple small local NeRFs. Barron et al. [4] improved the rendering quality with an online distillation strategy. Wang et al. [56] designed R2L to learn the light fields of a pre-trained NeRF model effectively. Fang et al. [9] proposed progressive volume distillation, which provides a systematic KD method that allows the distillation between explicit and implicit NeRF architectures. Instead of applying KD to optimize a small network for model compression or optimization purposes, we utilize KD to transfer the knowledge of a photo-realistically edited RF to improve the quality of later non-photorealistic editing.

3 StyleRF-VolVis

The objective of StyleRF-VolVis is to edit the *style* (i.e., appearance) of the DVR scene represented by a NeRF model without losing the original *content* (i.e., geometry) information. The style can be *photorealistic* (such as adjusting the original rendering's *color*, *opacity*, and *lighting* attributes) or *non-photorealistic* (such as altering the *texture* to resemble

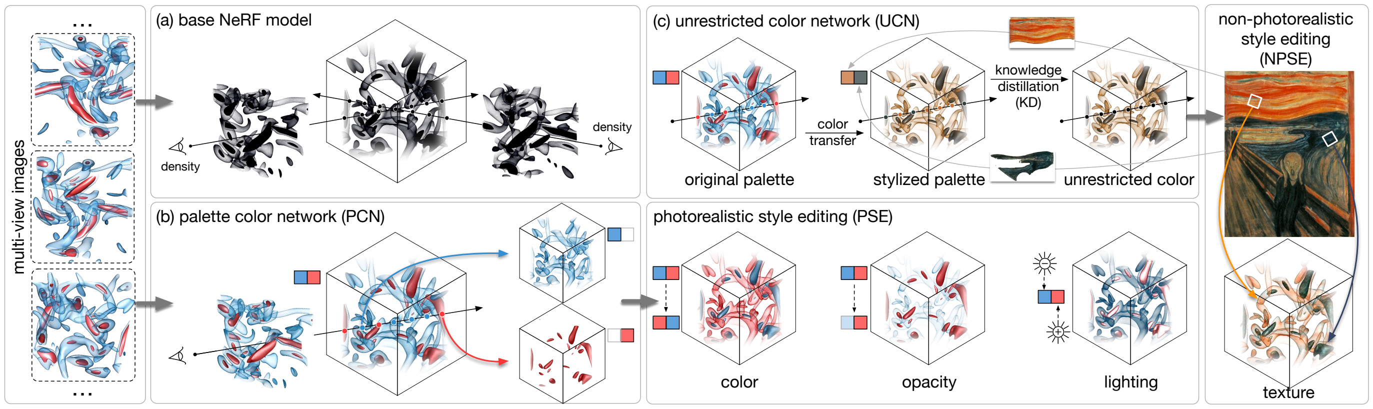

multi-view images
(a) base NeRF model
density
density
(b) palette color network (PCN)
(c) unrestricted color network (UCN)
color transfer
knowledge distillation (KD)
original palette
stylized palette
unrestricted color
non-photorealistic style editing (NPSE)
photorealistic style editing (PSE)
color
opacity
lighting
texture

Fig. 2: The workflow of StyleRF-VolVis. (a) Given a collection of multi-view images, we first optimize a base NeRF model for accurate density representation. (b) Then, we train a PCN for PSE, allowing color, opacity, and lighting changes. (c) Next, we utilize KD to optimize a UCN with no color palette constraint. Optimized with the stylization loss, the trained UCN can produce NPSE results.

the brushstroke of an artistic image). Distinct from conventional 3D neural stylization methods, our StyleRF-VolVis aims to unify two types of style editing into a single framework. Additionally, from a visualization perspective, users design TFs to classify the regions of interest in the DVR process. These TFs map distinct colors and opacities to each region users specify, highlighting relevant data ranges through visual mapping. Such visual content is critical in obtaining helpful visualization. Our method must keep the original visual content consistent across the views to ensure the stylized visualization remains meaningful, facilitating accurate interpretation and analysis.

Figure 2 illustrates the three-stage workflow of StyleRF-VolVis: (a) a *base NeRF model* for optimizing the density representation of the scene, (b) a *palette color network* (PCN) for supporting *photorealistic style editing* (PSE) and maintaining visual content consistency during style editing, and (c) an *unrestricted color network* (UCN) for enabling *non-photorealistic style editing* (NPSE). Accordingly, our method is structured into three optimization stages, each dedicated to training the parameters of one network while others are frozen for stability.

The first stage focuses on training a base NeRF model to reconstruct the scene geometry for subsequent style editing. The parameters of the trained base NeRF are then frozen to ensure that the following style editing does not affect the underlying scene geometry.

The second stage optimizes the PCN. Similar to TFs that differentiate volumetric regions with user-defined colors and opacities, the goal here is to classify regions of the RF with a color palette extracted from multi-view images. To this end, the PCN is optimized to fit input images with a weighted linear combination of extracted palette colors, given the density representation obtained from the previous stage. By learning to represent the colors of the RF, our model achieves PSE, including modifying the original rendering's color, opacity, and lighting.

The number of palette colors inherently restricts the network's ability to utilize more diverse color patterns for NPSE, where each style often consists of multiple colors. To address this issue, in the third stage, we start with stylizing the palette colors to match the target style colors. Leveraging KD, we then transfer the color information from the PCN (i.e., *teacher model*) to a UCN (i.e., *student model*), representing the colors of the RF without the color palette constraint. In addition, we can apply different styles to different visual regions extracted from the PCN. As a result, we stylize the visualization scene with NPSE while maintaining the original visual content.

The rest of Section 3 is structured as follows. In Section 3.1, we provide the preliminary knowledge of NeRF representation [39] and how we optimize the base NeRF model. In Sections 3.2 and 3.3, we discuss how we adapt NeRF's palette-based recoloring strategy [12, 32, 54] to fit PCN and achieve PSE. The utilization of KD [9] to extract UCN was described in Section 3.4, followed by the NPSE details that combine stylization loss [31, 63] and unsupervised segmentation techniques [30] in Section 3.5. Finally, we brief our interactive interface design in Section 3.6.

3.1 Base NeRF Model

StyleRF-VolVis is built upon the scene representation from a base NeRF model. In general, a NeRF model [39] represents the scene with two neural functions: a *density function* $\sigma(\mathbf{x})$ that maps any 3D position $\mathbf{x}$ to a density value σ and a *color function* $\mathbf{c}(\mathbf{x},\mathbf{d})$ that outputs RGB color $\mathbf{c}$ given an arbitrary 3D position $\mathbf{x}$ and viewing direction $\mathbf{d}$. NeRF samples rays from the camera origin $\mathbf{o}$ to the rendering pixels for each training camera view $\mathbf{d}$. For one sample ray $\mathbf{r} = (\mathbf{o},\mathbf{d})$, if we sample M points along ray $\mathbf{r}$ with 3D positions $\mathbf{x}_{1\ldots M}$ at depths $\mathbf{t}_{1\ldots M}$. The predicted pixel color $\hat{\mathbf{c}}(\mathbf{r})$ of ray $\mathbf{r}$ is computed as

$$\hat{\mathbf{c}}(\mathbf{r}) = \sum_{i=1}^{M} \alpha_i(1-\omega_i)\mathbf{c}(\mathbf{x}_i,\mathbf{d}), \tag{1}$$

where $\omega_i = \exp(-(\mathbf{t}_i - \mathbf{t}_{i-1})\sigma(\mathbf{x}_i))$ represents the transmittance along the ray between $\mathbf{x}_i$ and $\mathbf{x}_{i-1}$. $\alpha_i = \prod_{j=1}^{i-1}\omega_i$ denotes the attenuation of the ray from its origin $\mathbf{o}$ to $\mathbf{x}_i$. We select the advanced architecture from Instant-NGP [40] to optimize the RF, ensuring fast convergence and efficient rendering. Specifically, Instant-NGP comprises a multiresolution hash-grid encoding and a tiny MLP for efficient optimization. During the optimization of the base NeRF, we optimize with a loss $\mathcal{L}_{\text{BASE}}$ defined as

$$\mathcal{L}_{\text{BASE}} = ||\mathbf{c}(\mathbf{r}) - \hat{\mathbf{c}}(\mathbf{r})||_2^2, \tag{2}$$

where $\mathbf{c}(\mathbf{r})$ and $\hat{\mathbf{c}}(\mathbf{r})$ are the ground-truth (GT) and predicted pixel colors corresponding to ray $\mathbf{r}$. After optimizing the base NeRF, we freeze its model parameters in the subsequent stages to avoid style editing influencing the scene geometry. Note that the colors of the base NeRF will not be used in rendering the stylized RF; we employ the PCN for PSE and the UCN for NPSE.

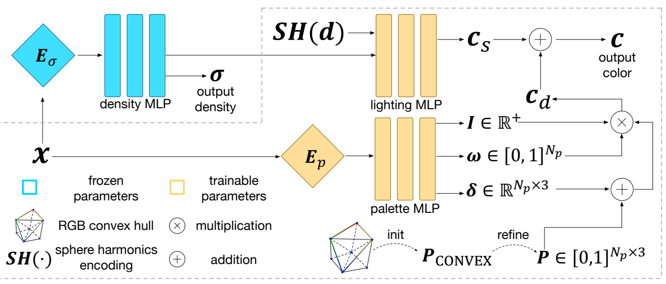


Fig. 3: The PCN architecture. The final predicted color is obtained by summing the view-dependent specular color output from the lighting MLP and the view-independent diffuse color output from the palette MLP. After training, palette weights ω are used for classifying points within the RF.

3.2 Palette Color Network (PCN)

After training the base NeRF model for accurate density representation, we optimize a PCN that represents the scene appearance with colors from a palette. The trained PCN can classify any 3D position within the RF with its palette color. The classification of the PCN preserves region classification from the user-defined TF, ensuring visual content consistency of the downstream style editing. Figure 3 shows the PCN architecture. The network is composed of two branches: one branch uses the lighting MLP to predict the *view-dependent* specular color $\mathbf{c}_s$ and the other branch uses the palette MLP to output the *view-independent* diffuse color $\mathbf{c}_d$.

For the specular color branch, we use a structure similar to the Instant-NGP's color function. The lighting MLP also ingests geometry features output from the density MLP and *spherical harmonics* (SH) encoded view direction as input. The only difference is that the output of the lighting MLP is a grayscale color instead of the original RGB color. Before optimization, we leverage the hidden parameters of the base NeRF's color function to initialize the shallow layers of the palette MLP to improve the prediction of $\mathbf{c}_s$ at the early stage.

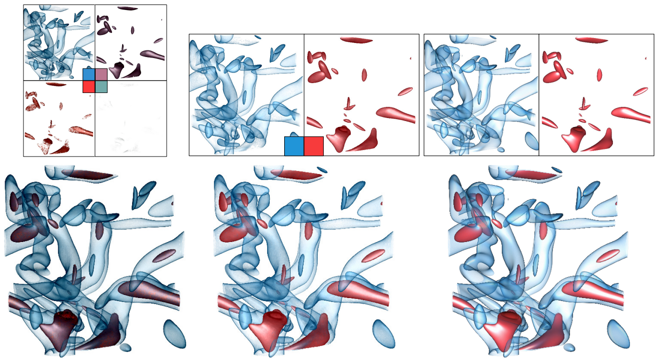


Fig. 4: Classification and rendering results without and with palette refinement. Top: classification results for each palette color. Bottom: rendering results when considering all palette colors.

For the diffuse color branch, we design a palette MLP that captures the weights ω of palette colors $\mathbf{P}$ for any points within the RF. This way, we can use ω to classify points in the RF according to their primary colors, preserving visual content consistency across different views during style editing. Similar to the recent NeRF's palette-based recoloring strategy [12, 32, 54], we apply the RGB convex hull method [49] to extract colors $\mathbf{P}_{\text{CONVEX}}$ from training images. However, unlike previous palette recoloring methods that directly employ $\mathbf{P}_{\text{CONVEX}}$ to initialize the palette, we further refine $\mathbf{P}_{\text{CONVEX}}$ as it usually contains many similar colors that may significantly hinder the classification. We remove similar colors when the distance between two colors is below a certain threshold. This distance is measured by the *hue* component of colors in the hue-saturation-brightness (HSB) space. The *saturation* and *brightness* components of palette colors are not considered in measuring the distance as they can be easily optimized during training. Figure 4 gives an example that compares classification and rendering results without and with palette refinement. Refer to the appendix for more details about the refinement algorithm.

Once the palette colors are initialized and the number of palette colors N_P is determined, we construct and optimize the PCN. During the optimization, the density MLP and hash-grid encoder $\mathbf{E}_\sigma$ learned in the first stage are fixed for invariant density representation. The palette MLP takes features output from the palette hash-grid encoder $\mathbf{E}_p$ as input and outputs three values: an intensity value I shared by all palette colors to counteract the range shift caused by the RGB convex hull color normalization, a set of weights ω that indicate the contribution of each palette color to $\mathbf{c}_d$, and a color offset vector δ to enhance the expressiveness of palette colors. Given the refined palette colors $\mathbf{P}$ and specular color $\mathbf{c}_s$, the output color $\mathbf{c}$ associated with position $\mathbf{x}$ and view direction $\mathbf{d}$ is computed as

$$\mathbf{c}(\mathbf{x},\mathbf{d}) = \mathbf{c}_s(\mathbf{x},\mathbf{d}) + I(\mathbf{x})\sum_{i=1}^{N_P}\omega_i(\mathbf{x})(P_i+\delta_i(\mathbf{x})), \tag{3}$$

where $\mathbf{I}(\mathbf{x})$ is the intensity value at position $\mathbf{x}$, $\omega_i(\mathbf{x})$ and $\delta_i(\mathbf{x})$ denote the weight and color offset for palette color P_i at $\mathbf{x}$, respectively. We then predict pixel color $\hat{\mathbf{c}}(\mathbf{r})$ using Equation 1 for $\mathbf{c}(\mathbf{x},\mathbf{d})$, and optimize network parameters and palette colors with loss $\mathscr{L}_{\text{PALETTE}}$ defined as

$$\mathscr{L}_{\text{PALETTE}} = ||\mathbf{c}(\mathbf{r})-\hat{\mathbf{c}}(\mathbf{r})||_2^2 + \lambda_\delta \sum_{i=1}^{M}\sum_{j=1}^{N_P}||\delta_j(\mathbf{x}_i)||_2^2. \tag{4}$$

The first term of $\mathscr{L}_{\text{PALETTE}}$ is the MSE loss between the GT pixel color $\mathbf{c}(\mathbf{r})$ and predicted pixel color $\hat{\mathbf{c}}(\mathbf{r})$. The second term is the offset regularization loss for sample points along ray $\mathbf{r}$, aiming to suppress the magnitudes of color offsets and avoid significant palette color shiftings. We set $\lambda_\delta = 0.1$ to control the regularization strength.

3.3 Photorealistic Style Editing (PSE)

When inferring the PCN, the output palette weights $\omega(\mathbf{x})$ can classify any 3D position within the RF according to its palette color. We then tune the network values to support PSE. Interactive PSE is achieved using Instant-NGP's fast inference ability.

We use the HSB space for recoloring following [32]. Given the target palette colors $\mathbf{P}'$, we compute the difference $\Delta\mathbf{P}$ between the original palette colors $\mathbf{P}$ and $\mathbf{P}'$ in the HSB space. ΔP_i are then added to $P_i+\delta_i(\mathbf{x})$ in Equation 3. To achieve opacity or lighting editing, we multiply $\sigma(\mathbf{x}_{P_i})$ or $\mathbf{c}_s(\mathbf{x}_{P_i},\mathbf{d})$ by a scalar value, where $\mathbf{x}_{P_i}$ denotes a point position receiving its primary color contribution from P_i.

3.4 Unrestricted Color Network (UCN)

Although the PCN can represent the colors of the RF with PSE, it prohibits using diverse colors in NPSE. To address this issue, we utilize KD to optimize a UCN (student model) that produces similar diffuse colors within the PCN (teacher model) with no color palette constraint.

Before distillation, we initially extract the average palette colors $\bar{\mathbf{P}}$ from each reference style to replace the original palette colors $\mathbf{P}$. This color transfer step ensures the distilled student network can match with styles, reducing optimization effort in NPSE. Our UCN comprises one hash-grid encoder $\mathbf{E}_u$ following an unrestricted MLP that outputs color $\mathbf{c}_u$. During distillation, we first randomly sample camera views from the scene. Then we optimize $\mathbf{E}_u$ with volume-aligned loss $\mathscr{L}_{\text{VOLUME}}$ adopted from [9] for each view. $\mathscr{L}_{\text{VOLUME}}$ is defined as

$$\mathscr{L}_{\text{VOLUME}} = ||\mathbf{E}_p(\mathbf{x})-\mathbf{E}_u(\mathbf{x})||_2^2, \tag{5}$$

where $\mathbf{E}_p(\mathbf{x})$ and $\mathbf{E}_u(\mathbf{x})$ are the output features from the palette and unrestricted encoders for each sample point $\mathbf{x}$. We optimize $\mathscr{L}_{\text{VOLUME}}$ for 150 iterations to initialize $\mathbf{E}_u(\mathbf{x})$ to accelerate the subsequent distillation. For the following 500 iterations, we optimize both $\mathbf{E}_u(\mathbf{x})$ and unrestricted MLP with color loss $\mathscr{L}_{\text{COLOR}}$, which is defined as

$$\mathscr{L}_{\text{COLOR}} = ||\bar{\mathbf{c}}_d(\mathbf{x})-\mathbf{c}_u(\mathbf{x})||_2^2, \tag{6}$$

where $\bar{\mathbf{c}}_d(\mathbf{x})$ and $\mathbf{c}_u(\mathbf{x})$ are the diffuse colors output from the PCN with the average palette colors $\bar{\mathbf{P}}$ and the UCN at $\mathbf{x}$.

3.5 Non-Photorealistic Style Editing (NPSE)

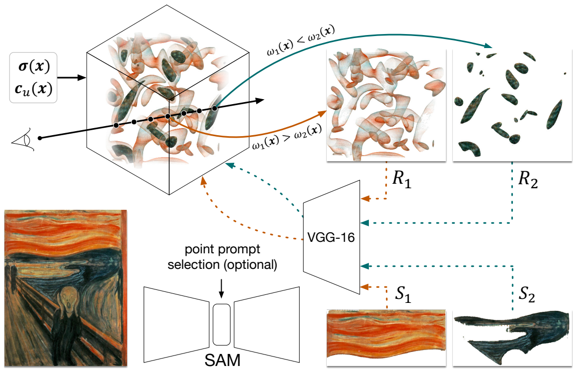
Given one or multiple reference images, we first extract N_P desired reference styles $\mathbf{S} = \{S_1, S_2, \cdots, S_{N_P}\}$ for N_P regions of the RF. When the number of reference images is less than N_P, or users are only interested in local patterns of one image, we can leverage unsupervised segmentation techniques such as the segment anything model (SAM) [30] to automatically or manually (with point prompt selection) divide a reference image into localized style regions for flexible stylization.

Figure 5 shows our training process for non-photorealistic style optimization. We use the optimizable unrestricted color $\mathbf{c}_u(\mathbf{x})$ (Section 3.4) and frozen density $\sigma(\mathbf{x})$ of the base NeRF (Section 3.1) to construct a new RF. During training, we first compute N_P rendering images

$\mathbf{R} = \{R_1, R_2, \cdots, R_{N_P}\}$ from the new RF for each camera view. For one sample point $\mathbf{x}$, it only contributes to R_i if its palette weight $\omega_i(\mathbf{x})$ is greater than other palette weights. Once we have N_P renderings $\mathbf{R}$ and styles $\mathbf{S}$, a pre-trained VGG-16 model [47] computes a stylization loss based on a user-specified or randomly assigned style mapping. Specifically, we use the *nearest neighbor feature matching* (NNFM) loss [31, 63] for each rendering and style, which is formulated as

$$\mathscr{L}_{\text{NNFM}}(\mathbf{R}, \mathbf{S}) = \frac{1}{(N_F N_P)} \sum_{i=1}^{N_P} \sum_{f_j \in \phi(R_i)} \min_{f_k \in \phi(S_i)} d(f_j, f_k), \tag{7}$$

where $\phi(R_i)$ and $\phi(S_i)$ are the rendering and style feature vectors extracted from the VGG-16 model ϕ. N_F is the number of feature vectors for $\phi(R_i)$, and d is the cosine distance between feature vectors f_j and f_k.



$\sigma(x)$
$c_u(x)$
$\omega_1(x) < \omega_2(x)$
$\omega_1(x) > \omega_2(x)$
R_1
R_2
VGG-16
point prompt selection (optional)
SAM
S_1
S_2

Fig. 5: The process of NPSE. We start segmenting one reference image into styles (S_1, S_2) via SAM. Each rendering (R_1, R_2) showing separate content is then stylized by one assigned style.

Unlike previous NeRF-based non-photorealistic stylization methods [26, 34, 63], our style transfer process involves three unique designs for the VolVis scenario. First, different from traditional NeRF-based stylization that directly discards the view-dependent color to maintain visual content consistency across the views, we optimize $\mathbf{c}_u(\mathbf{x})$ (i.e., diffuse color) during training and output $\mathbf{c}_u(\mathbf{x}) + \mathbf{c}_s(\mathbf{x}, \mathbf{d})$ (i.e., diffuse and specular colors) during inference. The rationale behind this design is to prevent the view-dependent color from influencing visual content consistency and preserve the specular lighting effect in the final output. Second, when rendering $\mathbf{R}$ in training, we observe that varying background colors could impact the stylization results (refer to an example shown in Figure 6). This phenomenon is caused by the inherent transparency in the DVR scene, which is not often present in natural scenes. Consequently, different background colors could lead to distinct renderings using the identical network, potentially incurring unsatisfactory stylization. To address this issue, we extract the luminance value from each style S_i as the corresponding background color for each rendering R_i. Third, we omit the content loss between the original and stylized images due to suboptimal stylization outcomes.

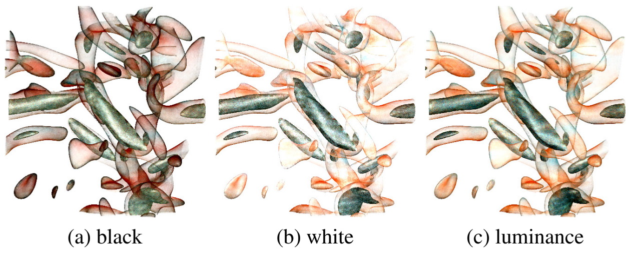
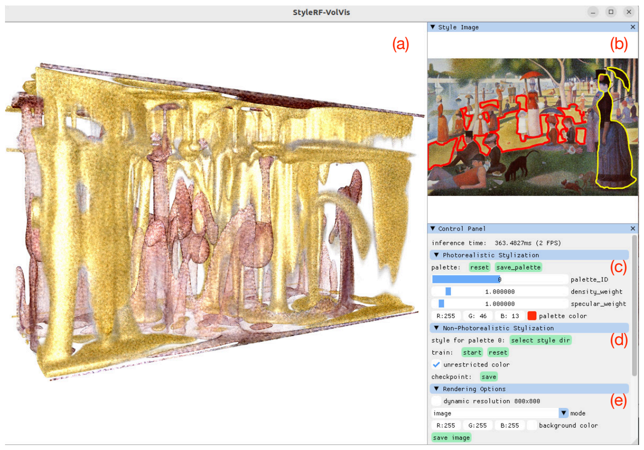


Fig. 6: Inference results using different background colors to optimize the NNFM loss. (c) yields the best stylization outcome.



Control Panel

inference time: 363.4827ms (2 FPS)

Photorealistic Stylization

palette: reset save_palette

0 palette_ID

1.000000 density_weight

1.000000 specular_weight

R:255 G: 46 B: 13 🟥 palette color

Non-Photorealistic Stylization

style for palette 0: select style dir

train: start reset

✓ unrestricted color

checkpoint: save

Rendering Options

dynamic resolution 800x800

image mode

R:255 G:255 B:255 background color

save image

Fig. 7: The screenshot of StyleRF-VolVis interface. (a) the stylization result of the DVR scene; (b) the reference image with selected styles highlighted with segmentation boundaries; (c) the PSE panel for color, density, and lighting editing; (d) the NPSE panel for style selection and UCN parameter training, resetting, and saving; (e) rendering options for adjusting the background color or saving stylized images, etc.

3.6 Interactive interface

Figure 7 shows our visual interface for users to conduct PSE and NPSE for a given DVR scene. For PSE, users select a palette ID and interactively modify the corresponding scene region's color, opacity, and lighting. For NPSE, users first specify the target style for each scene region from single or multiple reference images. They then start optimizing UCN. After NPSE, users can still change the opacity and lighting of the stylized DVR scene but cannot modify the color as the stylized scene is represented with UCN. Refer to the accompanying video for the recorded interaction with the interface.

Table 1: The datasets and their respective settings.

dataset	volume resolution	# visible ranges	# training images	# inference images	image resolution
aneurysm	$256{\times}256{\times}256$	1	92	181	800×800
combustion	$480{\times}720{\times}120$	3	92	181	800×800
earthquake	$256{\times}256{\times}96$	3	92	181	800×800
five jets	$128{\times}128{\times}128$	3	92	181	800×800
mantle	$360{\times}201{\times}180$	2	92	181	800×800
rotstrat	$4096{\times}4096{\times}4096$	2	92	181	800×800
solar plume	$128{\times}128{\times}512$	2	92	181	800×800
supernova	$432{\times}432{\times}432$	2	92	181	800×800
vortex	$128{\times}128{\times}128$	2	92	181	800×800

4 Results and Discussion

4.1 Datasets, Training, Baselines, and Metrics

Datasets and network training. To show the quality, consistency, and flexibility of StyleRF-VolVis, we evaluate it using the datasets listed in Table 1. The visible ranges are the opacity bumps in the TF specification corresponding to distinct visual contents for generating the original rendering. All reference images used are copyright-free, coming from WikiArt [2, 42] and Pexels [1]. We implemented StyleRF-VolVis using PyTorch and ran experiments on an NVIDIA RTX A4000 graphics card with 16 GB memory. We trained the base NeRF model with 30,000 iterations and PCN with 2,500 iterations using 92 DVR images with cameras evenly placed along a sphere enclosing the volume. The KD of UCN and NPSE does not need GT images. For KD, we optimized UCN with 624 iterations, where each iteration trains on one of the randomly sampled 312 camera views. For NPSE, we optimized UCN using the NNFM loss with 210 iterations, where each iteration trains on one of the uniformly sampled 42 camera views. For all three stages (base NeRF, PCN, and UCN), we applied the Adam optimizer with a learning rate of 0.01 and $\beta_1 = 0.9$, $\beta_2 = 0.999$ for training and set the batch size to 4,096 rays. For the base NeRF model, PCN, and UCN during KD, we decayed the learning rate exponentially for every iteration until it

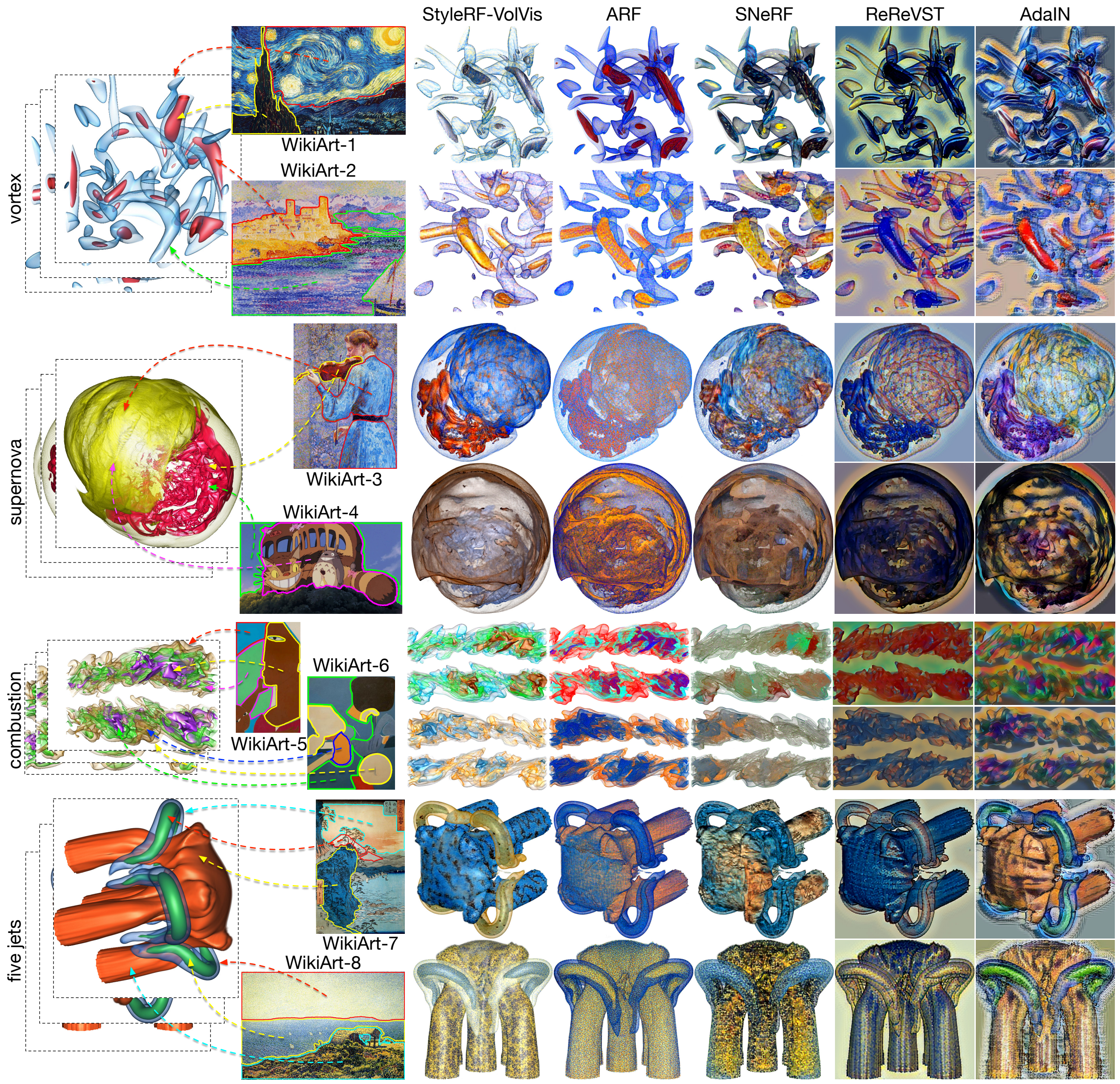


Fig. 8: Comparison of VolVis stylization. We provide two reference images for each dataset. For StyleRF-VolVis, we highlight extracted style regions and their correspondence with the DVR scene regions. None of the other methods support customized style selection and content mapping.

reached 0. During NPSE, we fixed the learning rate and followed the setting of ARF [63]. We used the *conv*3 block of VGG-16 to extract features for computing the NNFM loss. *Deferred backpropagation* was employed to update UCN's parameters for the entire image under limited GPU memory. During inference, we generated 181 stylized images with novel views, with cameras evenly placed along a trajectory to capture the full 360-degree view.

Baselines. We compare four baseline methods for NPSE:

- ARF [63] is a NeRF-based method that trains a NeRF model to represent the scene and then updates the pre-trained NeRF parameters with NNFM loss to generate stylized novel view images.
- SNeRF [41] is a NeRF-based method that first constructs the scene with a NeRF model and then alternatively finetunes the parameters of the pre-trained NeRF and an additional image stylization module to achieve stylization.
- ReReVST [57] is a video-based stylization method. It first performs novel view synthesis with a NeRF model, then arranges the novel view images as a video sequence, and finally performs video stylization to yield the result.
- AdaIN [11] is an image-based stylization method that optimizes a NeRF model for generating novel view images of the scene and then applies stylization to the NeRF synthesized images.

Note that we did not compare StyleRF-VolVis with the more recent StyleRF [34] work as it mainly focuses on the ability of zero-shot transfer to unseen reference images. StyleRF trains on all reference images from the WikiArt dataset to ensure better generalization for unseen reference images. However, this could lead to inferior stylization results compared to optimization methods like ARF and SNeRF. Existing visualization generation methods in scientific visualization [5, 17, 21, 23] were also excluded because they were not designed for style transfer.

Evalutation metrics. Since 3D style transfer is relatively novel and scarcely explored, limited metrics are available for quantitatively assessing stylization quality. Consequently, our evaluation focuses on *cross-view consistency* of stylization results. Specifically, we generated testing videos in which each frame represents a rendered image from one of the 181 novel viewpoints of our stylized scene. Then, we warped one view to the other using *softmax splatting* [43] according to the optical flow estimated with RAFT [52]. Finally, we computed the MSE and LPIPS [64] scores to measure the cross-view consistency. Similar to [34, 41, 57], we calculated the *long-range* consistency for faraway views (with an interval of 10 among the 181 inference views) and *short-range* consistency for adjacent views, respectively.

4.2 Qualitative Comparison

We compare StyleRF-VolVis against baseline methods using DVR scenes of vortex, supernova, combustion, and five jets. Each scene showcases two stylization results using distinct reference images. Figure 8 shows the qualitative comparison. Unlike all baseline methods, StyleRF-VolVis supports customized style selection and mapping to distinct visual contents, which we highlight in the DVR scene.

Overall, StyleRF-VolVis produces clearer and more consistent stylization results than other methods. In contrast to video- and image-based stylization methods (ReReVST and AdaIN), NeRF-based stylization methods (StyleRF-VolVis, ARF, and SNeRF) preserve better geometry consistency and are capable of separating foreground objects from the background during stylization. Among NeRF-based methods, SNeRF captures the overall style of the reference image but cannot assign distinct styles to different visual regions of the original scene. Thus, SNeRF's stylization is blurry and mixed, making it hard to identify the difference between individual visual contents in the original DVR scene. For the vortex dataset, ARF shows better stylization than SNeRF in preserving visual content distinction. However, the uncontrollable ARF style selection process could let the model choose a negligible local style that does not match the overall style (e.g., the WikiArt-1 case) or select one style for multiple visual regions (e.g., the WikiArt-8 case). Both drawbacks could lead to poor stylization results. In addition, in the stylization process, ARF and SNeRF discard view directions in their input to ensure cross-view stylization consistency. Consequently, they cannot preserve the view-dependent lighting of the DVR scene. Unlike ARF and SNeRF, StyleRF-VolVis utilizes the lighting MLP to preserve lighting information of visual content while ensuring cross-view consistency during NPSE by removing view direction input in UCN optimization. Moreover, StyleRF-VolVis allows explicit assigning of different styles to different DVR regions. Via color transfer, we ensure that the region color always aligns with the overall style color, avoiding the disadvantage of NNFM loss, which focuses on negligible local style.

Table 2: Comparing the total training time and per-image inference time for NeRF-based methods. The best ones are shown in bold.

dataset	method	training time	inference time
vortex	ARF	**14.6 m**	204 ms
	SNeRF	3.6 h	**173 ms**
	StyleRF-VolVis	29.7 m	373 ms
combustion	ARF	**20.1 m**	210 ms
	SNeRF	3.8 h	**195 ms**
	StyleRF-VolVis	1.2 h	564 ms

4.3 Quantitative Comparison

Training and inference time. We report the training and inference time for all NeRF-based methods in Table 2. The training of StyleRF-VolVis is slower than ARF but faster than SNeRF. Unlike ARF, StyleRF-VolVis requires additional steps to optimize PCN and apply KD to UCN before NPSE. Moreover, optimizing on a single camera view during NPSE requires rendering different regions independently. This process maps a style to a region, slowing the stylization process. Although the alternative training process of SNeRF is relatively straightforward, it demands significant time to train the image stylization module, leading to the longest training time. For inference, StyleRF-VolVis needs additional feedforward steps of the lighting MLP to provide lighting information, which takes longer than ARF or SNeRF.

Short- and long-range consistency. In Table 3, we compare short- and long-range cross-view consistency for different methods. We use MSE and LPIPS as the metrics. For each metric, we report the average value over eight stylization cases (WikiArt-1 to WikiArt-8) shown in Figure 8. StyleRF-VolVis achieves the best cross-view consistency, while AdaIN performs the worst as this image-based stylization method treats each image independently. ReReVST leads to comparable or even better MSEs than the NeRF-based methods because it produces stylization with relatively uniform colors, resulting in smaller pixel-wise errors. However, under the image-level LPIPS metric, the consistency of ReReVST significantly lags behind the NeRF-based methods.

Table 3: Average short- and long-range cross-view consistency for stylization cases shown in Figure 8.

	short-range		long-range	
method	MSE↓	LPIPS↓	MSE↓	LPIPS↓
AdaIN	0.087	0.171	0.118	0.217
ReReVST	0.049	0.096	**0.075**	0.137
ARF	0.053	0.056	0.093	0.105
SNeRF	0.046	0.060	0.082	0.106
StyleRF-VolVis	**0.045**	**0.054**	0.076	**0.092**

Table 4: The votes of 14 participants on better solutions gathered from eight stylization cases (WikiArt-1 to WikiArt-8).

	vortex		supernova		combustion		five jets		
method	1	2	3	4	5	6	7	8	sum
ARF	8	8	2	5	11	10	11	**19**	74
SNeRF	**23**	12	**27**	15	6	9	10	8	110
StyleRF-VolVis	11	**22**	13	**22**	**25**	**23**	**21**	15	**152**

Voting results from a user study. As the consistency metrics do not necessarily reflect the perceived style transfer quality, we conducted a user study to measure user preference for different stylizations following the University's IRB protocol. Per Figure 8, we included four datasets, each with two reference images, and considered three leading methods (ARF, SNeRF, and StyleRF-VolVis) for pairwise comparison. This leads to 24 (4×2×3) image pairs organized into eight (4×2) groups. We recruited 14 students from a visualization class of undergraduate, Master's, and Ph.D. students in computer science and engineering, aerospace and mechanical engineering, and psychology majors.

The participants were briefed on the evaluation criteria before proceeding to the study. A full-screen display shows each pair. At the top of each display, the original DVR and reference images (with extracted style regions highlighted in different color boundaries) are shown. At the bottom, the stylized images of two methods, randomly placed on the left and right sides and labeled 'A' and 'B,' are presented. The participants were asked to decide which one ('A' or 'B') achieved the better stylization outcome, and no tie was allowed. We asked the participants to take their time, as we did not record how much time they spent on each pair, each group, or the entire study. During the evaluation, they could go back and forth to update their votes. We advised them that many factors should be considered, including the overall impression, content preservation, style application, and visuals (color, opacity, and lighting). They could decide how to weigh them, and their criteria should be consistent throughout the study. The entire study was completed in the classroom within 10 minutes.

The voting results are shown in Table 4. We can see that StyleRF-VolVis wins for all stylization cases except WikiArt-1, WikiArt-3, and WikiArt-8. This may be attributed to the fact that some participants prefer the stylization result of ARF that provides a detailed local pattern for the WikiArt-8 case or the stylization results of SNeRF, which contain a more global pattern in WikiArt-1 (i.e., the dark tone and contrasting colors of *The Starry Night*) and WikiArt-3 cases. However, StyleRF-VolVis wins for all other cases and gains the most overall preference. The cross-view consistency of StyleRF-VolVis is also better than ARF and SNeRF, as shown in Table 3.

4.4 Flexibility of NPSE

Our NPSE strategy allows users to define which style should be applied to which part of a DVR scene (analogous to using the TF), which

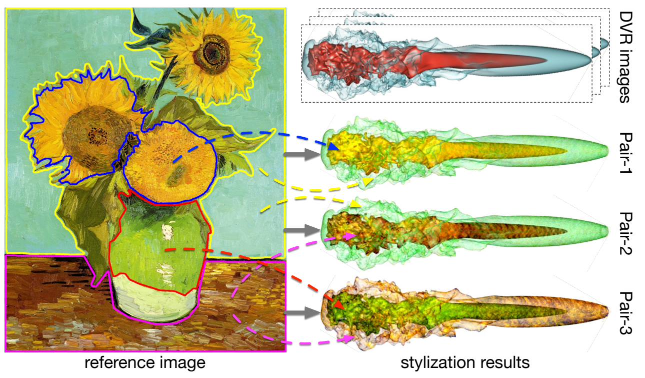


Fig. 9: StyleRF-VolVis results on the solar plume dataset where three style pairs from one reference image are used for flexible stylization.

provides more flexibility than other NPSE methods. Specifically, users can apply styles obtained from one reference image for multiple stylizations, leverage styles extracted from multiple reference images for one stylization, or only stylize one part of the DVR scene. All these additions are the unique features of StyleRF-VolVis.

One reference image for multiple stylizations. Unlike conventional NeRF-based stylization methods, which only produce one stylized scene from a single reference image, StyleRF-VolVis can generate various stylized outcomes with only one reference image. Thanks to the advanced segmentation of PCN and SAM for the RF and reference image, we can extract multiple distinct styles and apply them to different scene regions. By pairing styles and regions in various combinations, StyleRF-VolVis achieves controllable and diverse stylization outcomes. Figure 9 shows multiple stylization results of the solar plume DVR scene using the same reference image. By comparing Pair-1 and Pair-2, we can see that when the style of one region remains unchanged, altering the style of another region keeps the unchanged style region intact. Further comparison of Pair-2 and Pair-3 reveals that applying the same style to different regions consistently transfers the expected texture. These favorable features allow StyleRF-VolVis to offer users flexible stylization.

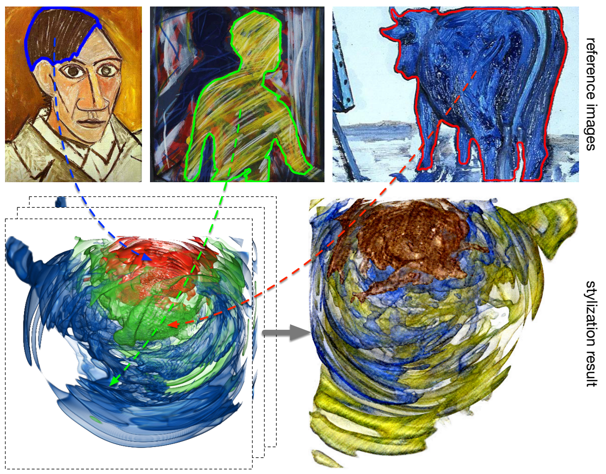
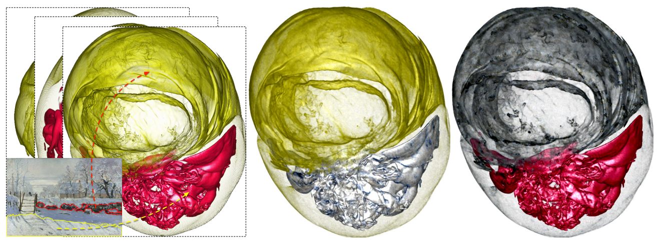


Fig. 10: StyleRF-VolVis result on the earthquake dataset where three styles from different reference images are used for flexible stylization.

Multiple reference images for one stylization. Previous approaches like StyleRF [34] suggest leveraging extra supervision from large models pre-trained on extensive image datasets for segmenting scene elements to support style transfer from multiple reference images to a single scene. However, such a strategy cannot be directly applied to VolVis scenes. One reason lies in the inherent difference between DVR and natural scenes. DVR images exhibit more complex geometric relationships, with different visual contents often nested in layers with varied transparencies, rendering 2D segmentation methods futile. Another reason is that large models are typically pre-trained on natural images. Adapting such models to process DVR images would require extensive DVR images and substantial hardware resources for fine-tuning, a demand that is impractical for user-oriented VolVis scene stylization. Our model can segment various regions via PCN, even though separate GT rendering results for different regions are missing during optimization and additional supervision from a 2D segmentation method is lacking. As illustrated in Figure 10, StyleRF-VolVis achieves stylized results for different regions using styles from multiple reference images. With a collection of reference images, StyleRF-VolVis could create vast combinations of different stylization results for one DVR scene.



(a) DVR scene and styles (b) partial stylization 1 (c) partial stylization 2

Fig. 11: StyleRF-VolVis partial stylization results on the supernova dataset.

Partial stylization. In some scenarios, users may wish to stylize only certain regions of the scene while keeping others unchanged. Unlike previous NeRF-based stylization methods, StyleRF-VolVis achieves such partial stylization by combining the color representations of PCN and UCN. After completing the NPSE of each region, PCN can still utilize the original palette colors to represent the DVR scene. Users can thus specify which regions should maintain the NPSE texture and which should keep the original appearance. During rendering, regions that preserve the original scene's appearance utilize colors represented by PCN, while those stylized regions are processed through UCN. Figure 11 shows an example of partial stylization, providing extra freedom for users to achieve desirable results.

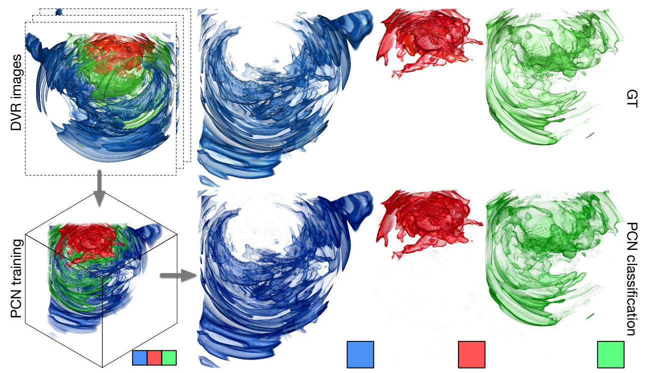


Fig. 12: Comparision of StyleRF-VolVis PCN classification rendering results with GT for each region of the earthquake dataset.

4.5 PCN Classification and PSE

As discussed in Section 4.4, the PCN classification outcomes are crucial for style editing. Visual contents in a DVR scene are often nested with each other. However, as demonstrated in Figure 12, even for a complex scene like the earthquake, PCN can still produce region classification closely resembling GT, even though only DVR images are used for training. Although the colors and lighting of PCN classification results are not perfect, the boundary of each region is clear, and even fine details maintain consistency with GT. This ensures precise PSE and NPSE of each region in subsequent stages.

In Figure 13, we showcase the PSE outcomes for various DVR scenes. With the assistance of PCN classification, we can perform a distinct PSE in each region. Such flexibility further assists users in achieving desired stylization results.

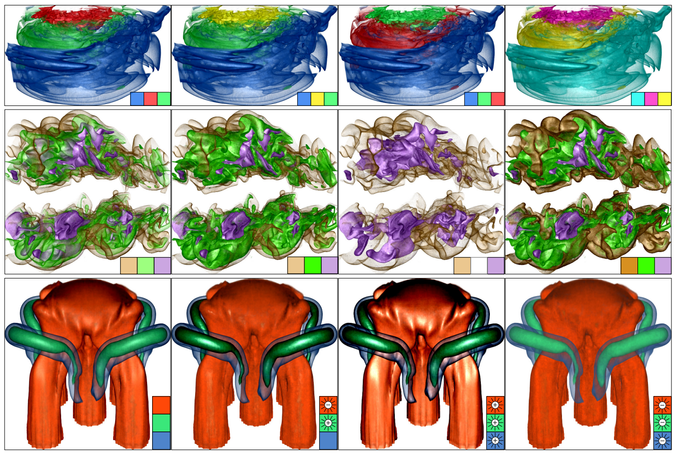
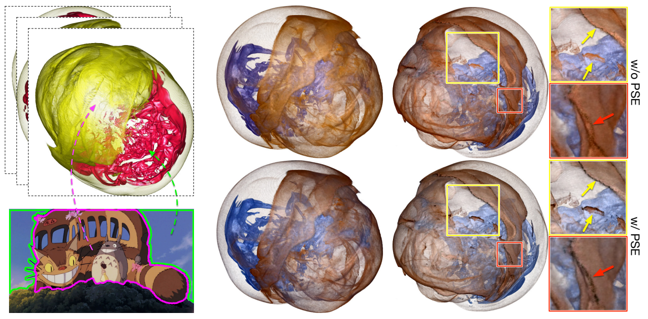


Fig. 13: StyleRF-VolVis PSE of color, opacity, and lighting on earthquake, combustion, and five jets datasets, respectively.

4.6 Ablation Study

Benefit of PSE to NPSE. At the beginning of UCN optimization, we perform the color transfer in PCN (teacher network) to ensure that the color representation of UCN (student network) matches the selected styles before NPSE. This PSE step is essential for speeding up the convergence and improving the stylization quality of UCN during NPSE optimization. Figure 14 compares the style transfer process without and with color transfer. Comparing the stylization results of 42 and 168 iterations, we can see that NPSE without color transfer suffers from inaccurate style color matching at an early stage and shows unclear stylization (i.e., black borders, see arrows) at 168 iterations. In contrast, NPSE with color transfer matches the overall style color well at the early stage and reveals well-stylized texture at 168 iterations.



(a) DVR scene and styles (b) 42 iterations (c) 168 iterations

Fig. 14: Comparison of the style transfer process of StyleRF-VolVis without and with PSE for color transfer on the supernova dataset.

View-dependent lighting in NPSE. Recent NeRF-based stylization methods [8, 26, 34, 41, 63] discard view directions in the input to ensure cross-view stylization consistency. However, they ignore that view-dependent lighting of the original scene is essential for maintaining consistency between the original content and the stylized scene. In contrast, StyleRF-VolVis utilizes the additional lighting MLP optimized in the PCN training stage to preserve the DVR lighting during NPSE. In Figure 15, we show examples of vortex and supernova datasets to compare the effect of DVR lighting on NPSE outcomes. The results show that the stylized scene with lighting is more consistent with the original DVR scene than that without.

4.7 Limitations

Although StyleRF-VolVis achieves flexible, high-quality stylization of DVR scenes, it has the following limitations. First, given an initial TF, NeRF cannot represent the value ranges where opacity equals

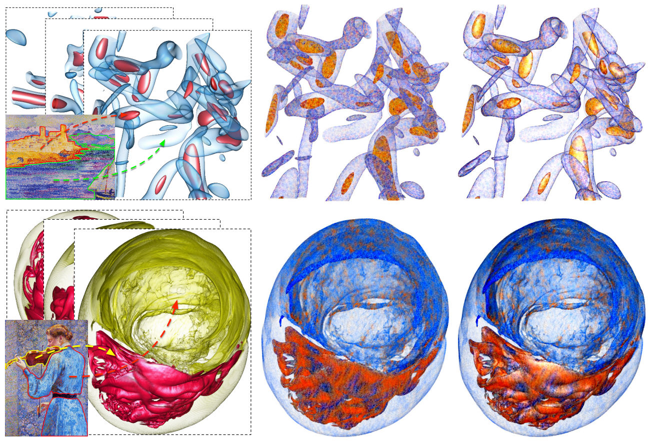

(a) DVR scene and styles (b) w/o lighting (c) w/ lighting

Fig. 15: Comparison of the stylization results of StyleRF-VolVis without and with lighting on vortex and supernova datasets.

zero. Consequently, StyleRF-VolVis cannot perform PSE or NPSE for regions with zero opacity in the initial TF. Second, StyleRF-VolVis supports interactive PSE but not interactive NPSE. Third, even though users can adjust the lighting *intensity* for the DVR scene, they cannot modify the lighting *direction* because no normal information is available. Finally, if the colors among different regions within the input DVR scene are similar, color refinement may not separate these regions correctly. In such cases, users may need to manually adjust the number of palette colors and their RGB values to obtain the desired results.

5 Conclusions and Future Work

We have presented StyleRF-VolVis, the first work in VolVis that targets style transfer in the NeRF space. The crux of our approach lies in the accurate extraction of content and appearance information separately from the given DVR scene and the bridging between photorealistic and non-photorealistic style editing via knowledge distillation. With these innovations, StyleRF-VolVis achieves high-quality, consistent, and flexible 3D style transfer outcomes with novel view synthesis. The efficacy of StyleRF-VolVis is demonstrated with various combinations of DVR scenes and reference images. Moreover, we compare StyleRF-VolVis against other image-based (AdaIN), video-based (ReReVST), and NeRF-based (ARF and SNeRF) solutions via objective and subjective evaluation to showcase its superior quality performance.

In the future, we will extend StyleRF-VolVis to handle dynamic DVR scenes produced from time-varying datasets. The challenge is maintaining a consistent appearance over timesteps to achieve temporally coherent stylization. We will also explore StyleRF-VolVis for multivariate or ensemble VolVis, where different variables or ensembles could be mapped to visually distinct styles beyond colors for better differentiation. It remains to be seen what the appropriate number of styles and their mixing should be to leverage the human's visual capacity best while maintaining observation clarity.

To ease the difficulty for non-professionals using StyleRF-VolVis, we will explore integrating natural language interaction into the current graphical user interface and broaden the selection of reference images from the WikiArt collection to images created by generative AI. The success of StyleRF-VolVis will unfold exciting opportunities for VolVis beyond the originated scientific domain. We envision that such a solution will enable citizen science by fusing diverse disciplines, such as science and art, for the general public's exploration, understanding, and appreciation, which we would like to pursue.

Acknowledgments

This research was supported in part by the U.S. National Science Foundation through grants IIS-1955395, IIS-2101696, OAC-2104158, and IIS-2401144, and the U.S. Department of Energy through grant DE-SC0023145. The authors would like to thank the anonymous reviewers for their insightful comments.

References

[1] Pexels - the best free stock photos, royalty free images & videos shared by creators. https://www.pexels.com/. 5

[2] WikiArt - visual art encyclopedia. https://www.wikiart.org/. 5

[3] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan. Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of IEEE/CVF International Conference on Computer Vision*, pp. 5835–5844, 2021. doi: 10.1109/ICCV48922.2021.00580 2

[4] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5460–5469, 2022. doi: 10.1109/CVPR52688.2022.00539 2

[5] M. Berger, J. Li, and J. A. Levine. A generative model for volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 25(4):1636–1650, 2019. doi: 10.1109/TVCG.2018.2816059 1, 2, 6

[6] S. Bruckner and M. E. Gröller. Style transfer functions for illustrative volume rendering. *Computer Graphics Forum*, 26(3):715–724, 2007. doi: 10.1111/j.1467-8659.2007.01095.x 2

[7] Y. Chen, G. Shao, K. C. Shum, B.-S. Hua, and S.-K. Yeung. Advances in 3D neural stylization: A survey. *arXiv preprint arXiv:2311.18328*, 2023. doi: 10.48550/arXiv.2311.18328 2

[8] P.-Z. Chiang, M.-S. Tsai, H.-Y. Tseng, W.-S. Lai, and W.-C. Chiu. Stylizing 3D scene via implicit representation and hypernetwork. In *Proceedings of IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 215–224, 2022. doi: 10.1109/WACV51458.2022.00029 2, 9

[9] S. Fang, W. Xu, H. Wang, Y. Yang, Y. Wang, and S. Zhou. One is all: Bridging the gap between neural radiance fields architectures with progressive volume distillation. In *Proceedings of AAAI Conference on Artificial Intelligence*, pp. 597–605, 2023. doi: 10.1609/aaai.v37i1.25135 2, 3, 4

[10] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5491–5500, 2022. doi: 10.1109/CVPR52688.2022.00542 2

[11] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2414–2423, 2016. doi: 10.1109/CVPR.2016.265 2, 6

[12] B. Gong, Y. Wang, X. Han, and Q. Dou. RecolorNeRF: Layer decomposed radiance fields for efficient color editing of 3D scenes. In *Proceedings of ACM International Conference on Multimedia*, pp. 8004–8015, 2023. doi: 10.1145/3581783.3611957 2, 3, 4

[13] P. Gu, D. Z. Chen, and C. Wang. NeRVI: Compressive neural representation of visualization images for communicating volume visualization results. *Computers & Graphics*, 116:216–227, 2023. doi: 10.1016/J.CAG.2023.08.024 2

[14] J. Han and C. Wang. TSR-TVD: Temporal super-resolution for time-varying data analysis and visualization. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):205–215, 2020. doi: 10.1109/TVCG.2019.2934255 2

[15] J. Han and C. Wang. SSR-TVD: Spatial super-resolution for time-varying data analysis and visualization. *IEEE Transactions on Visualization and Computer Graphics*, 28(6):2445–2456, 2022. doi: 10.1109/TVCG.2020.3032123 2

[16] J. Han and C. Wang. VCNet: A generative model for volume completion. *Visual Informatics*, 6(2):62–73, 2022. doi: 10.1016/J.VISINF.2022.04.004 2

[17] J. Han and C. Wang. CoordNet: Data generation and visualization generation for time-varying volumes via a coordinate-based neural network. *IEEE Transactions on Visualization and Computer Graphics*, 29(12):4951–4963, 2023. doi: 10.1109/TVCG.2022.3197203 1, 2, 6

[18] J. Han, H. Zheng, and C. Bi. KD-INR: Time-varying volumetric data compression via knowledge distillation-based implicit neural representation. *IEEE Transactions on Visualization and Computer Graphics*, 2023. Accepted. doi: 10.1109/TVCG.2023.3345373 2

[19] J. Han, H. Zheng, D. Z. Chen, and C. Wang. STNet: An end-to-end generative framework for synthesizing spatiotemporal super-resolution volumes. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):270–280, 2022. doi: 10.1109/TVCG.2021.3114815 2

[20] J. Han, H. Zheng, Y. Xing, D. Z. Chen, and C. Wang. V2V: A deep learning approach to variable-to-variable selection and translation for multivariate time-varying data. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1290–1300, 2021. doi: 10.1109/TVCG.2020.3030346 2

[21] W. He, J. Wang, H. Guo, K.-C. Wang, H.-W. Shen, M. Raj, Y. S. G. Nashed, and T. Peterka. InSituNet: Deep image synthesis for parameter space exploration of ensemble simulations. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):23–33, 2020. doi: 10.1109/TVCG.2019.2934312 1, 2, 6

[22] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. doi: 10.48550/arXiv.1503.02531 2

[23] F. Hong, C. Liu, and X. Yuan. DNN-VolVis: Interactive volume visualization supported by deep neural network. In *Proceedings of IEEE Pacific Visualization Symposium*, pp. 282–291, 2019. doi: 10.1109/PACIFICVIS.2019.00041 1, 2, 6

[24] W. Hu, Y. Wang, L. Ma, B. Yang, L. Gao, X. Liu, and Y. Ma. Tri-MipRF: Tri-mip representation for efficient anti-aliasing neural radiance fields. In *Proceedings of IEEE/CVF International Conference on Computer Vision*, pp. 19717–19726, 2023. doi: 10.1109/ICCV51070.2023.01811 2

[25] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of IEEE/CVF International Conference on Computer Vision*, pp. 1510–1519, 2017. doi: 10.1109/ICCV.2017.167 2

[26] Y.-H. Huang, Y. He, Y.-J. Yuan, Y.-K. Lai, and L. Gao. StylizedNeRF: Consistent 3D scene stylization as stylized NeRF via 2D-3D mutual learning. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18321–18331, 2022. doi: 10.1109/CVPR52688.2022.01780 2, 5, 9

[27] Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, and M. Song. Neural style transfer: A review. *IEEE Transactions on Visualization and Computer Graphics*, 26(11):3365–3385, 2020. doi: 10.1109/TVCG.2019.2921336 2

[28] J. Johnson, A. Alahi, and F.-F. Li. Perceptual losses for real-time style transfer and super-resolution. In *Proceedings of European Conference on Computer Vision*, pp. 694–711, 2016. doi: 10.1007/978-3-319-46475-6_43 2

[29] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. 3D Gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):139:1–139:14, 2023. doi: 10.1145/3592433 2

[30] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023. doi: 10.48550/arXiv.2304.02643 3, 4

[31] N. Kolkin, M. Kucera, S. Paris, D. Sykora, E. Shechtman, and G. Shakhnarovich. Neural neighbor style transfer. *arXiv preprint arXiv:2203.13215*, 2022. doi: 10.48550/arXiv.2203.13215 3, 5

[32] Z. Kuang, F. Luan, S. Bi, Z. Shu, G. Wetzstein, and K. Sunkavalli. PaletteNeRF: Palette-based appearance editing of neural radiance fields. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20691–20700, 2023. doi: 10.1109/CVPR52729.2023.01982 2, 3, 4

[33] J.-H. Lee and D.-S. Kim. ICE-NeRF: Interactive color editing of NeRFs via decomposition-aware weight optimization. In *Proceedings of IEEE/CVF International Conference on Computer Vision*, pp. 3468–3478, 2023. doi: 10.1109/ICCV51070.2023.00323 2

[34] K. Liu, F. Zhan, Y. Chen, J. Zhang, Y. Yu, A. E. Saddik, S. Lu, and E. Xing. StyleRF: Zero-shot 3D style transfer of neural radiance fields. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8338–8348, 2023. doi: 10.1109/CVPR52729.2023.00806 2, 5, 6, 7, 8, 9

[35] A. Lu, C. J. Morris, D. S. Ebert, P. Rheingans, and C. D. Hansen. Non-photorealistic volume rendering using stippling techniques. In *Proceedings of IEEE Visualization Conference*, pp. 211–218, 2002. doi: 10.1109/VISUAL.2002.1183777 12, 13

[36] Y. Lu, P. Gu, and C. Wang. FCNR: Fast compressive neural representation of visualization images. In *Proceedings of IEEE VIS Conference (Short Papers)*, 2024. Accepted. 2

[37] Y. Lu, K. Jiang, J. A. Levine, and M. Berger. Compressive neural representations of volumetric scalar fields. *Computer Graphics Forum*, 40(3):135–146, 2021. doi: 10.1111/CGF.14295 2

[38] P. Luo, Z. Zhu, Z. Liu, X. Wang, and X. Tang. Face model compression by distilling knowledge from neurons. In *Proceedings of AAAI Conference on Artificial Intelligence*, pp. 3560–3566, 2016. doi: 10.1609/aaai.v30i1.10449 2

[39] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proceedings of European Conference on Computer Vision*, pp. 405–421, 2020. doi: 10.1007/978-3-030-58452-8_24 2, 3

[40] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4):102:1–102:15, 2022. doi: 10.1145/3528223.3530127 2, 3

[41] T. Nguyen-Phuoc, F. Liu, and L. Xiao. SNeRF: Stylized neural implicit representations for 3D scenes. *ACM Transactions on Graphics*, 41(4):142:1–142:11, 2022. doi: 10.1145/3528223.3530107 6, 7, 9

[42] K. Nichol and W. Kan. Painter by numbers - does every painter leave a fingerprint? `https://kaggle.com/competitions/painter-by-numbers`, 2016. 5

[43] S. Niklaus and F. Liu. Softmax splatting for video frame interpolation. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5436–5445, 2020. doi: 10.1109/CVPR42600.2020.00548 7

[44] C. Reiser, S. Peng, Y. Liao, and A. Geiger. KiloNeRF: Speeding up neural radiance fields with thousands of tiny MLPs. In *Proceedings of IEEE/CVF International Conference on Computer Vision*, pp. 14315–14325, 2021. doi: 10.1109/ICCV48922.2021.01407 2

[45] M. Ruder, A. Dosovitskiy, and T. Brox. Artistic style transfer for videos. In *Proceedings of German Conference on Pattern Recognition*, pp. 26–36, 2016. doi: 10.1007/978-3-319-45886-1_3 2

[46] N. Shi, J. Xu, H. Li, H. Guo, J. Woodring, and H.-W. Shen. VDL-Surrogate: A view-dependent latent-based model for parameter space exploration of ensemble simulations. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):820–830, 2023. doi: 10.1109/TVCG.2022.3209413 2

[47] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image regnition. In *Proceedings of International Conference on Learning Representation*, 2015. 5

[48] M. Stone. *A Field Guide to Digital Color*. AK Peters, 2003. 12

[49] J. Tan, J. Echevarria, and Y. Gingold. Efficient palette-based decomposition and recoloring of images via RGBXY-space geometry. *ACM Transactions on Graphics*, 37(6):262:1–262:10, 2018. doi: 10.1145/3272127.3275054 4

[50] K. Tang and C. Wang. ECNR: Efficient compressive neural representation of time-varying volumetric datasets. In *Proceedings of IEEE Pacific Visualization Conference*, pp. 72–81, 2024. doi: 10.1109/PACIFICVIS60374.2024.00017 2

[51] K. Tang and C. Wang. STSR-INR: Spatiotemporal super-resolution for time-varying multivariate volumetric data via implicit neural representation. *Computers & Graphics*, 119:103874, 2024. doi: 10.1016/J.CAG.2024.01.001 2

[52] Z. Teed and J. Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *Proceedings of European Conference on Computer Vision*, pp. 402–419, 2020. doi: 10.1007/978-3-030-58536-5_24 7

[53] A. Tewari, J. Thies, B. Mildenhall, P. Srinivasan, E. Tretschk, W. Yifan, C. Lassner, V. Sitzmann, R. Martin-Brualla, S. Lombardi, et al. Advances in neural rendering. *Computer Graphics Forum*, 41(2):703–735, 2022. doi: 10.1111/cgf.14507 2

[54] K. Tojo and N. Umetani. Recolorable posterization of volumetric radiance fields using visibility-weighted palette extraction. *Computer Graphics Forum*, 41(4):149–160, 2022. doi: 10.1111/cgf.14594 3, 4

[55] C. Wang and J. Han. DL4SciVis: A state-of-the-art survey on deep learning for scientific visualization. *IEEE Transactions on Visualization and Computer Graphics*, 29(8):3714–3733, 2023. doi: 10.1109/TVCG.2022.3167896 1

[56] H. Wang, J. Ren, Z. Huang, K. Olszewski, M. Chai, Y. Fu, and S. Tulyakov. R2L: Distilling neural radiance field to neural light field for efficient novel view synthesis. In *Proceedings of European Conference on Computer Vision*, pp. 612–629, 2022. doi: 10.1007/978-3-031-19821-2_35 2

[57] W. Wang, S. Yang, J. Xu, and J. Liu. Consistent video style transfer via relaxation and regularization. *IEEE Transactions on Image Processing*, 29:9125–9139, 2020. doi: 10.1109/TIP.2020.3024018 2, 6, 7

[58] S. Weiss, M. Chu, N. Thuerey, and R. Westermann. Volumetric isosurface rendering with deep learning-based super-resolution. *IEEE Transactions on Visualization and Computer Graphics*, 27(6):3064–3078, 2021. doi: 10.1109/TVCG.2019.2956697 2

[59] S. Weiss, P. Hermüller, and R. Westermann. Fast neural representations for direct volume rendering. *Computer Graphics Forum*, 41(6):196–211, 2022. doi: 10.1111/cgf.14578 2

[60] Q. Wu, D. Bauer, M. J. Doyle, and K.-L. Ma. Interactive volume visualization via multi-resolution hash encoding based neural representation. *IEEE Transactions on Visualization and Computer Graphics*, 2023. Accepted. doi: 10.1109/TVCG.2023.3293121 2

[61] S. W. Wurster, T. Xiong, H.-W. Shen, H. Guo, and T. Peterka. Adaptively placed multi-grid scene representation networks for large-scale data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):965–974, 2024. doi: 10.1109/TVCG.2023.3327194 2

[62] S. Yao, J. Han, and C. Wang. GMT: A deep learning approach to generalized multivariate translation for scientific data analysis and visualization. *Computers & Graphics*, 112:92–104, 2023. doi: 10.1016/J.CAG.2023.04.002 2

[63] K. Zhang, N. Kolkin, S. Bi, F. Luan, Z. Xu, E. Shechtman, and N. Snavely. ARF: Artistic radiance fields. In *Proceedings of European Conference on Computer Vision*, pp. 717–733, 2022. doi: 10.1007/978-3-031-19821-2_41 2, 3, 5, 6, 9

[64] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018. doi: 10.1109/CVPR.2018.00068 7

APPENDIX

1 IMPLEMENTATION DETAILS

Palette refinement algorithm. When initializing palette colors $\mathbf{P}$ before PCN training, we refine $\mathbf{P}_{\text{CONVEX}}$ extracted from the RGB convex hull method to eliminate similar colors within the palette. Specifically, for any two colors in $\mathbf{P}_{\text{CONVEX}}$, if they are similar (i.e., their L1 distance of *hue* value in the HSB space is below a threshold T_h), we remove one of them. Furthermore, there is a small chance that $\mathbf{P}_{\text{CONVEX}}$ may contain gray colors corresponding to the light color in the DVR scene. Such gray colors are unnecessary in $\mathbf{P}$ as the lighting MLP has represented lighting components. Therefore, we remove any color in $\mathbf{P}_{\text{CONVEX}}$ if its *brightness* value in the HSB space is less than a threshold T_b. For normalized color component values in the HSB space, we empirically set $T_h = 0.1$ and $T_b = 0.2$ to obtain refined $\mathbf{P}$.

Luminance background. Before NPSE, we calculate each style's luminance value L to compute the NNFM loss using VGG-16. Following the calculation steps given in [48], for each average RGB color of the selected style, we normalize each component, such as R, and convert it into the linear-scale counterpart R_{lin}

$$R_{\text{lin}} = R^{2.2}. \tag{8}$$

L of the selected style can be computed as

$$L = 0.2126 \times R_{\text{lin}} + 0.7152 \times G_{\text{lin}} + 0.0722 \times B_{\text{lin}}, \tag{9}$$

where the RGB component coefficients reflect the average spectral sensitivity of lighting perceived by humans. We use L as the corresponding background color for each style rendering. This way, the stylization of StyleRF-VolVis can better match the overall brightness of the selected style.

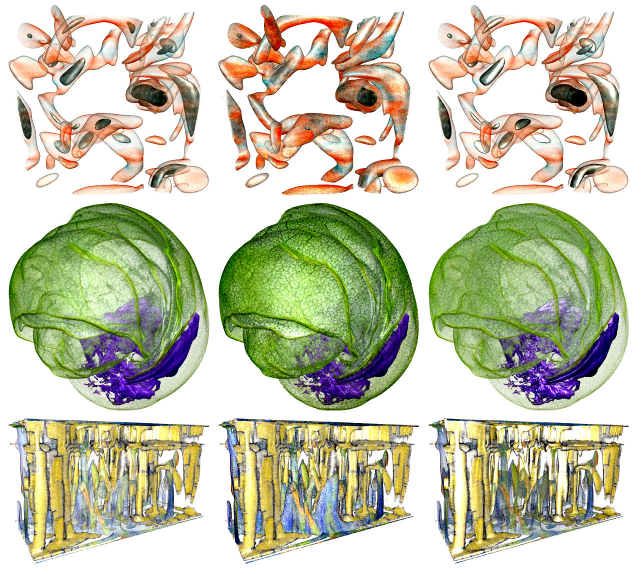

(a) NPSE scene (b) opacity PSE (c) lighting PSE

Fig. 1: Applying various opacity or lighting PSEs to the stylized NPSE scenes.

2 ADDITIONAL RESULTS

PSE after NPSE. After NPSE, users can still apply further PSE to the stylized scene. When doing so, one limitation is that if a region of the NPSE scene utilizes the UCN to represent the color term, users cannot apply a color PSE to the region as the PCN does not represent the color. However, users can modify the opacity and lighting of each region without restriction. Figure 1 shows examples of applying various opacity or lighting PSEs to the scene after NPSE.

Choice of λ_δ. When optimizing the PCN to avoid palette color shiftings, we include an offset regularization loss and use λ_δ to control the regularization strength. We conduct an ablation study to investigate the effect of λ_δ on PCN performance. After optimizing the PCN with different λ_δ, we compare PCN rendering results with GT using PSNR, LPIPS, and SSIM, as shown in Table 1. Figure 2 presents the rendering results. We can see that the offset regularization loss is essential for PCN training. When offset regularization is missing ($\lambda_\delta = 0$), the PCN does not predict correctly, as it tries to focus more on leveraging offsets instead of palette colors to represent the DVR scene. However, the PCN may leverage more on palette colors instead of offsets to represent colors when λ_δ gets larger, resulting in less accurate scene reconstruction. Therefore, we choose $\lambda_\delta = 0.1$ for a good control of the regularization strength.

Table 1: Averaging PSNR (dB), LPIPS, and SSIM values across all PCN-inferred images with the combustion dataset. The best ones are shown in bold.

λ_δ	PSNR↑	LPIPS↓	SSIM↑
0.0	20.47	0.079	0.954
0.1	**23.93**	**0.056**	**0.969**
1.0	22.99	0.058	0.967

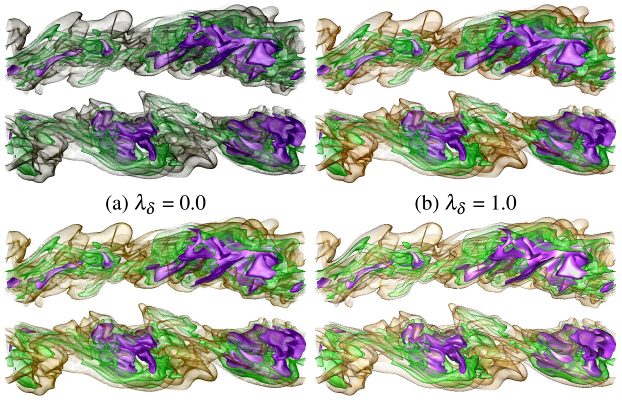

(a) λ_δ = 0.0 (b) λ_δ = 1.0 (c) λ_δ = 0.1 (d) GT

Fig. 2: Comparison of PCN rendering results under different λ_δ on the combustion dataset.

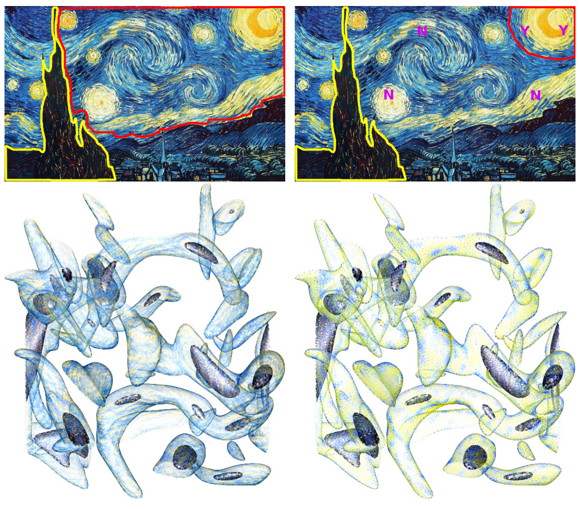
Iterative style refinement for NPSE. For NPSE, StyleRF-VolVis does not support direct control over the stylization process. However, users can indirectly achieve their desired stylization by iterative refining the selection in the reference image. Figure 3 shows such an example. Based on the style selection in (a), we apply several negative point prompts for SAM in (b) to exclude undesired styles and retrain the UCN to achieve the desired stylization.

Comparison with NPR. Conventional *non-photorealistic rendering* (NPR) of a DVR scene relies on predefined generation rules to mimic a reference style. Due to the complexity of hyperparameter settings, the effectiveness of these methods depends on the choice of hyperparameters to some extent. Moreover, such an approach limits the stylization results to a single style. In contrast, StyleRF-VolVis can transfer arbitrary styles according to different reference images within a unified framework, offering higher flexibility and robustness. In Figure 4, we compare the stylization results of StyleRF-VolVis and a conventional NPR method for stipple drawing style [35]. We use the stipple drawing in [35] as a reference image and adjust the TF and camera pose to align the viewpoint for comparison. StyleRF-VolVis matches the overall texture of the reference style more closely.

Comparison with DVR. In DVR, users can adjust the TF to explore the scene. StyleRF-VolVis can also achieve similar objectives through the PSE of PCN. However, PSE cannot adjust invisible parts of the scene constructed via training the multiview images. Despite this limitation, PCN supports random access to any position within the scene during the rendering's sampling process. Compared to DVR, PCN may achieve faster render speed and require smaller memory footprints for large volumes under the same TF. In Table 2, we compare DVR and PCN regarding the average rendering time per image, CPU

Table 2: Average rendering time, CPU memory, and GPU memory footprints for DVR and PCN as well as average PSNR (dB) and LPIPS values of PCN under different volume and image resolutions of the rotstrat dataset. Since E3 cannot be rendered in our local test machine using DVR due to out-of-memory (OOM), we record its performance on a high-performance cluster and highlight it in bold for reference.

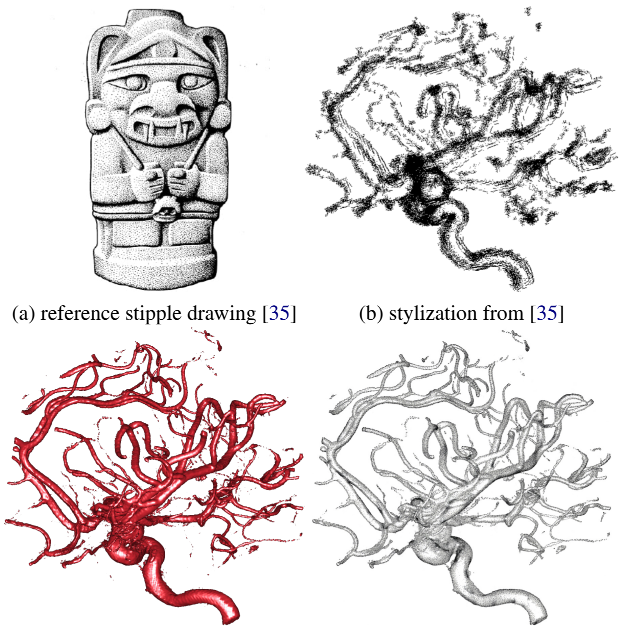
experiment	volume		DVR				PCN				
ID	resolution	size	resolution	time	CPU mem	GPU mem	PSNR↑	LPIPS↓	time	CPU mem	GPU mem
E1	512^3	0.5 GB	800^2	111 ms	3.3 GB	1.4 GB	28.71	0.0326	145 ms	4.9 GB	4.6 GB
E2	1024^3	4.1 GB	800^2	388 ms	11.3 GB	6.5 GB	26.68	0.0462	153 ms	4.9 GB	4.6 GB
E3	2048^3	32.8 GB	800^2	–/**3526 ms**	OOM/**74.8 GB**	OOM/**45.5 GB**	25.28	0.0469	166 ms	4.9 GB	4.6 GB
E4	1024^3	4.1 GB	400^2	323 ms	11.2 GB	6.5 GB	25.05	0.0369	76 ms	4.7 GB	3.1 GB
E2	1024^3	4.1 GB	800^2	388 ms	11.3 GB	6.5 GB	26.68	0.0462	153 ms	4.9 GB	4.6 GB
E5	1024^3	4.1 GB	1200^2	510 ms	11.4 GB	6.9 GB	26.89	0.0503	314 ms	5.2 GB	7.2 GB



(a) before refinement (b) after refinement

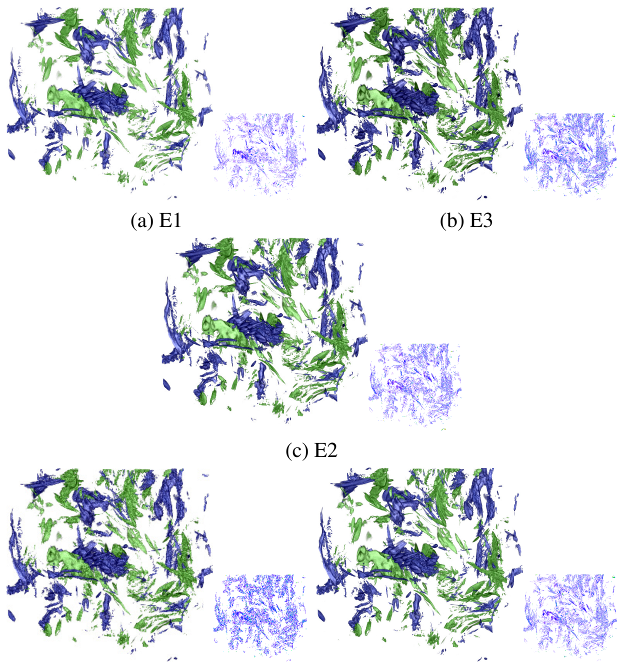
Fig. 3: Iterative style refinement on the vortex dataset. Y/N shows a positive/negative point prompt to include/exclude a certain selection.

memory, and GPU memory requirement for the rotstrat dataset with different volume and image resolutions (denoted by experiment IDs). We run DVR using the open-source software ParaView with NVIDIA IndeX plugins. All models converge around four minutes and occupy a storage of 168 MB. PSNR and LPIPS values reported in Table 2 and the difference images (with respect to the GT rendering image) shown in Figure 5 suggest that PCN achieves acceptable accuracy but requires a smaller rendering memory footprint and faster rendering speed compared to DVR. Note that the NeRF-based representation is independent of the volume resolution. Storing the training images and the network model for large data is more space-efficient than the original volume. Therefore, StyleRF-VolVis provides an efficient means for altering renderings of large volumes.


(a) reference stipple drawing [35] (b) stylization from [35]

(c) DVR (d) stylization from StyleRF-VolVis

Fig. 4: Comparison of the stylization results generated by a NPR method [35] and StyleRF-VolVis on the aneurysm dataset.


(a) E1 (b) E3

(c) E2

(d) E4 (e) E5

Fig. 5: PCN rendering results with different volume and image resolutions on the rotstrat dataset.