ReVolVE: Neural Reconstruction of Volumes for Visualization Enhancement of Direct Volume Rendering

Siyuan Yao[a], Chaoli Wang[a]

[a]*Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, 46556, IN, USA*

Abstract

Due to its ability to create novel views and bypass conventional rendering, visualization generation has become a trending topic of deep learning for scientific visualization research. Given a set of direct volume rendering (DVR) images for training, existing solutions design neural networks to learn the mapping between input parameters and output images. Once learned, the network can synthesize novel views from unseen parameters. However, no attempt is made to reconstruct the underlying volumes for more flexible downstream visualization synthesis. In this paper, we introduce neural reconstruction of volumes for visualization enhancement (ReVolVE), a new framework based on the neural radiance field that reconstructs high-quality density and color volumes from unlit DVR images to enable novel view synthesis with various visualization enhancement options. Leveraging a compact volume representation with an efficient vector-matrix decomposition scheme and a color-based volume segmentation, ReVolVE can effectively separate density and color volumes that cover prominent visual content implicitly represented by a generic transfer function mapping, paving the way for subsequent visualization enhancement via neural rendering or conventional rendering. We demonstrate the effectiveness of ReVolVE across multiple volumetric datasets, showcasing its visual enhancement outcomes and comparing its superior performance against representative state-of-the-art methods (DiffDVR, Plenoxels, 3DGS, TensoRF, and Instant-NGP).

Keywords: Direct volume rendering, neural radiance field, deep learning, visualization enhancement

1. Introduction

In scientific visualization, as a process that transforms 3D volume data into 2D images, volume rendering enables the visualization, analysis, and interpretation of large underlying datasets. This technique has become indispensable in advancing many scientific, engineering, and medical fields. Volume rendering can be primarily categorized into two types: *isosurface rendering* (IR) and *direct volume rendering* (DVR).

IR uses polygonal surfaces to represent volumes, focusing on specific isovalues. This method allows for efficient shading using conventional graphics shaders, making it a popular choice for applications requiring clear surface delineation. In contrast, DVR treats a volume as a semi-transparent medium with physical properties that influence its interaction with light, such as absorption, emission, and scattering [1], which provides a more comprehensive view of the volume's internal structure. Beyond the physics-based optical model, researchers have created a range of enhancement techniques [2, 3, 4, 5, 6] that provide various visual cues (e.g., depth, contour, and silhouette) to improve the perception of regions of interest.

Since DVR relies on volume ray casting, the volume data must be available during rendering. These volume data could be large-scale, making them costly to produce, store, and distribute. Besides data generation [7, 8, 9, 10, 11, 12] and neural compression [13, 14, 15, 16, 17], the recent surge of deep learning techniques [18] for visualization generation provides a promising alternative for creating DVR images [19, 20, 21, 22, 23, 24, 25]. One group of works [22, 24, 25] use *scene representation networks* (SRNs) to represent the raw volume itself, similar to how neural compression works. They require access to the full volume during training. Another group of works [20, 21, 23] are trained on DVR images and can generate new views without needing the original data. The key advantage of this second group of works is that, for large-scale simulations, scientists can render and store DVR images in situ as the simulation runs, avoiding the costly I/O and storage demands associated with saving the raw volume data. As shown in Section 4.4, the memory usage of these methods is not directly influenced by the size of the original volume, as they are only trained on the rendered images. These works also show an additional advantage in scenarios where the original volume data cannot be shared due to privacy or security constraints. However, they are limited to synthesizing DVR images with the same volume appearance as the training images. Although the geometry-aware stylization of DVR images [26] can modify the appearance to match the style of a reference image, the potential of deep learning techniques that apply established visual enhancements to volume visualization remains underexplored.

This paper introduces ReVolVE, neural Reconstruction of Volumes for Visualization Enhancement, an innovative *neural radiance field* (NeRF)-based framework that efficiently reconstructs high-quality density and color volumes from DVR images and creates novel volume visualizations with various visual enhancement options. We assume the original volume data

is unavailable during training or inference. Moreover, the input training DVR images are unlit, only reflecting color and opacity mappings determined by some generic transfer function (TF) without any lighting applied. The camera poses associated with these input training images are known, but the TF is unknown.

ReVolVE employs a hybrid representation of the volume data, which combines a vector-matrix decomposed volume [27] to store features with a compact *multilayer perceptron* (MLP) decoder. The MLP computes color and opacity from the given features to enhance the expressiveness of explicit representation. Upon completing its training, ReVolVE leverages a clustering algorithm to extract representative colors from the volume. These representative colors guide the production of segmentation masks that partition the volume into separate segments to support targeted edits. We provide visualization enhancement options that enable the neural network to infer enhanced DVR images directly. These options streamline the process of *neural rendering*, improving the visual quality of synthesized images. To further the flexibility of ReVolVE, we offer an alternative route that exports segmented density volumes for employing *conventional rendering* on these derived volumes. The contributions of our ReVolVE are the following

- ReVolVE is the first NeRF-based approach that directly enhances visualization without requiring access to the original volume data, using only DVR images. This innovation introduces new possibilities for feature enhancement and visual adjustment.
- ReVolVE outperforms representative state-of-the-art methods (DiffDVR [28], Plenoxels [29], 3DGS [30], TensoRF [27], and Instant-NGP [31]) for scene reconstruction and novel view synthesis of DVR images, both quantitatively and qualitatively.
- We comprehensively showcase visualization enhancement scenarios to highlight the versatility of ReVolVE and conduct ablation and hyperparameter studies to evaluate their impact and determine the optimal setup.

2. Related Work

Differentiable volume rendering. Differentiable renderers [32, 33, 34, 35, 36] are designed to optimize 3D scene representations by leveraging the gradients generated during the rendering process, using 2D supervision from sources like depth maps or multi-view images. Differentiable rendering is seldom explored in volume visualization, but recent research has demonstrated its significant potential. Nimier-David et al. designed Mitsuba 2 [37], a differentiable renderer with a wide range of rendering algorithms with the Markov Chain Monte Carlo sampling method. While its reverse-mode automatic differentiation allows 3D scene optimization, Mitsuba 2 significantly increases GPU memory usage for storing intermediate states. Moreover, unlike a neural network, direct gradient-based optimization on volume is prone to low-quality local minima, requiring careful initialization.

Weiss and Westermann introduced DiffDVR [28] to optimize all continuous parameters in the volume rendering process. DiffDVR recomputes intermediate states instead of storing them, allowing gradient calculation through automatic differentiation while conserving GPU memory for handling larger volumes. It applies to various volume visualization tasks, such as optimal viewpoint selection, TF reconstruction, and volume reconstruction. Notably, DiffDVR also supports density and color volume reconstruction from DVR images, closely matching the setup used in our ReVolVE framework. In Section 4, we render the volumes reconstructed using DiffDVR and compare the results with ReVolVE in terms of rendering quality.

Scene representation networks. In scientific visualization, several SRNs [22, 24, 25] have been effectively used to reduce the size of volume data by learning compact neural representations. These methods enable seamless integration with renderers that utilize the neural network directly for rendering, without requiring full reconstruction of the original volume, unlike traditional data compression techniques, which typically require decompressing the entire volume before visualization.

While ReVolVE may be seen as part of the broader SRN family, it fundamentally differs in that it represents pre-shaded volumetric scenes rather than raw volumetric data. ReVolVE is trained solely on a small number of unlit DVR images, eliminating the need for access to the original volume. This approach significantly reduces storage and data transfer demands, making it well-suited for large-scale or restricted-access scenarios.

Neural radiance fields. NeRF [38] represents a 3D scene by modeling a continuous volume of densities and direction-dependent colors through the differentiable volume rendering process. By utilizing an implicit representation of the 3D scene through a deep, fully connected neural network, NeRF captures and represents complex real-world geometry and appearance with a much more compact model size compared to traditional voxel-based (explicit) approaches. Subsequent research has concentrated on key improvements, such as enhancing rendering quality [39, 40, 41], minimizing the number of training images [42, 43], learning dynamic scenes [44, 45], enabling inverse rendering [46, 47, 48], and integrating compositionality [49, 50]. Apart from these improvements, our focus is on further exploring efficient 3D scene representations to optimize NeRF for volume reconstruction from DVR images.

Efficient 3D scene representations. The original NeRF, as a purely implicit approach that relies entirely on a deep MLP, incurs slow convergence, often requiring hours to days to train. To achieve orders of magnitude speedup, fully explicit methods such as NSVF [51], PlenOctrees [52], and Plenoxels [29] have been developed to represent radiance fields by directly storing densities and colors within the voxels of a 3D grid. Still, although optimization techniques such as skipping and omitting empty voxels have been employed to reduce memory footprints and storage costs, the overall demands still escalate rapidly as the volume resolution increases, echoing the limitations observed in the aforementioned differentiable renderers.

Representing the 3D scene with 3D Gaussians is another explicit approach that does not use neural networks and accelerates training and inference [30, 53, 54]. While 3D Gaussians are optimized to avoid computation and storage costs for empty space, rasterization allows efficient real-time rendering.

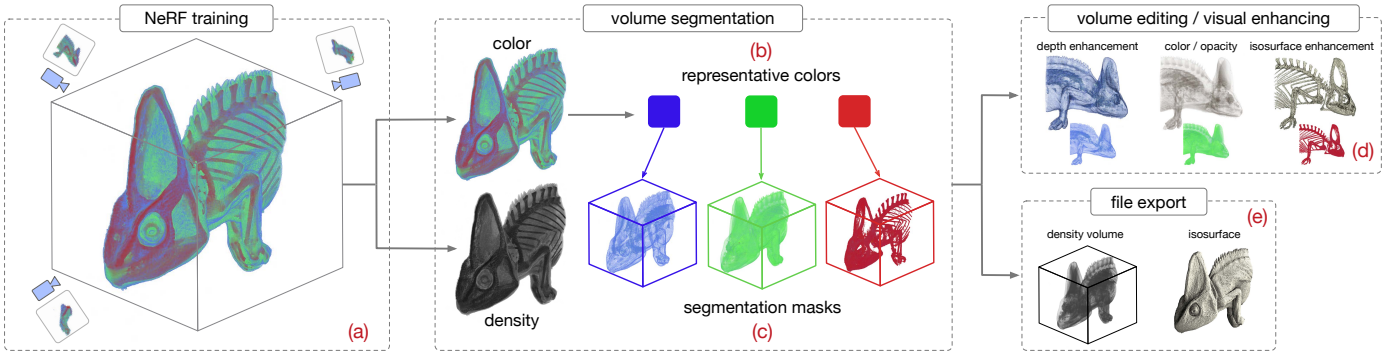


Figure 1: The overview of our ReVolVE framework. ReVolVE first (a) learns a NeRF model from a set of plain, unlit DVR images produced from a generic TF. The reconstructed color volume from the NeRF model is then used to (b) extract representative colors. The representative colors are used to (c) create volume segmentation masks, with each color mapping to a distinct region. With the segmentation masks, ReVolVE (d) enables the editing and enhancement of individual volume segments in the NeRF space. It also offers the capability to (e) export the density volume of an individual segment as scalar field data or its isosurface as a triangle mesh for conventional rendering.

However, since DVR scenes usually contain complex, semi-transparent structures, *3D Gaussian splatting* (3DGS) [30] may not perform effectively without using an excessive number of Gaussians.

Hybrid 3D representations are better suited for volume reconstruction from DVR images, as they accelerate the model by incorporating explicit representations while still utilizing neural networks to maintain 3D scene continuity. Instant-NGP [31] integrates a *multiresolution hash table* (MHT) of trainable feature vectors with a small network, exemplifying an excellent way for hybrid representation. Nevertheless, when applied to volume or surface reconstruction, this method can introduce noticeable noise and artifacts that are difficult to mitigate due to the nature of MHT encoding. An alternative hybrid solution generates an explicit 3D voxel grid representing the 3D scene. As a CNN-based approach, convolutional occupancy networks (CONet) [55] generate the occupancy volume with a 3D U-Net as the volume decoder. This method also implements a multi-plane decoder that decomposes the 3D representation into three planes. The concept of decomposition has been applied in radiance field generation, such as GSN [56] and EG3D [57], which utilize 2D CNNs to generate feature planes.

Based on this concept, TensoRF [27] further incorporates a vector-matrix decomposition to the radiance fields. Unlike generative methods that rely on neural networks to generate the decomposed representation, TensoRF uses an explicit decomposed representation of trainable features, much like the MHT features used in Instant-NGP. TensoRF, similar to Instant-NGP, combines explicit feature grids with a small MLP; however, in TensoRF, the MLP decoder is dedicated solely to computing view-dependent color, while density is directly obtained from the feature grid's voxel values. Like Instant-NGP, TensoRF shows noticeable artifacts due to its vector-matrix decomposed 3D grid representation. ReVolVE employs a hybrid 3D scene representation incorporating vector-matrix decomposition to optimize volume reconstruction. We also provide strategies for ReVolVE to minimize artifacts, resulting in superior renderings and more accurate volume reconstructions.

Neural rendering. Researchers have proposed replacing the whole rendering process with a neural network as an alternative to classical rendering techniques. We refer readers to the recent survey by Tewari et al. [58] for a general overview of neural rendering. As an example, RenderNet [59] replaces the mesh rasterizer with a combination of convolutional and fully connected networks. In scientific visualization, the works by Berger et al. [20] and He et al. [21] fall into the same line of research. GAN-VR [20] trains a network on DVR images and rendering parameters and uses the network to predict new visualizations using only the camera and TF parameters. InSituNet [21] lets a network learn the relationships between the input simulation parameters and rendering output. Then, this network directly synthesizes images from the new input parameters during inference. GAN-VR requires many rendering images (200,000) for training, and InSituNet only works with low-resolution (256×256) images. More recently, Han and Wang designed CoordNet [23], which employs an MLP based on SIREN activation functions [60] to infer high-resolution (1024×1024) DVR images from novel viewpoints. Li et al. [61] presented ParamsDrag, a model inspired by DragGAN [62], which facilitates parameter space exploration through direct interaction with visualizations.

ReVolVE is more robust and flexible than other methods [20, 21, 23] because it not only generates high-fidelity novel views by reconstructing volumes but also enables targeted visualization enhancements to specific regions by creating color-based volume masks. Additionally, our method requires significantly fewer images and trains faster, making it more convenient and efficient.

3. ReVolVE

The primary goal of ReVolVE is to achieve high-fidelity enhancement of DVR images rendered using an emission-absorption model [1], without access to the volume data or knowledge of the TF. As illustrated in Figure 1, ReVolVE starts with learning a NeRF model from a set of plain, unlit DVR images. By leveraging the NeRF model, it represents the 3D scene of these DVR images as a reconstructed volume consisting of densities and colors. ReVolVE further extracts representative colors from the color distribution of the reconstructed color vol-

ume. These colors serve as the basis for segmentation masks, allowing precise identification of different regions within the volume. ReVolVE can apply targeted edits or visual enhancements to specific volume segments with these masks. While ReVolVE includes various visualization enhancement methods that can directly utilize NeRF outputs, users also have the option to export, from the reconstructed volume, high-quality density volumes or isosurfaces from selected segments and follow the conventional rendering workflow.

3.1. Direct Volume Rendering with Radiance Field

Our model begins by taking a camera pose as a transformation matrix. A ray is cast from the camera through each pixel on the screen toward the volume. As the ray traverses the volume, it samples features from matrices and vectors representing the volume along its path, following TensoRF [27]. These sampled features are then processed by a small MLP to determine the density and color of each sample point. The densities and colors sampled along each ray are combined to compute the final color rendered at the corresponding pixel. This rendering process mirrors the method used for rendering DVR images, following the rendering equation

$$\begin{aligned} C(\mathbf{r}) &= \int_{t_n}^{t_f} \mathcal{T}(t)\sigma(\mathbf{r}(t))c(\mathbf{r}(t))dt, \\ \text{where } \mathcal{T}(t) &= \exp\Big(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\Big). \end{aligned} \tag{1}$$

In Equation 1, $C(\mathbf{r})$ denotes the color sampled along the ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$, where $\mathbf{o}$ is the origin, d is the ray direction, and t_n and t_f are near and far bounds, respectively. The accumulated transmittance function $\mathcal{T}(t)$ indicates the probability that no particle obstructs the ray between t_n and t. The volume density function $\sigma(\mathbf{x})$ gives the probability that the ray is intercepted by a particle at location $\mathbf{x}$ by a particle. The color term $c(\mathbf{x})$ corresponds to the particle that terminates the ray at location $\mathbf{x}$.

In practice, we use discrete sampling along the ray to approximate the integral, following the quadrature rule [1]. To estimate the color $C(\mathbf{r})$, n points are sampled along the ray and computed using the formula

$$\begin{aligned} \hat{C}(\mathbf{r}) &= \sum_{i=1}^{n} \mathcal{T}_i(1 - \exp(-\sigma_i\delta_i))c_i, \\ \text{where } \mathcal{T}_i &= \exp\Big(-\sum_{j=1}^{i-1} \sigma_j\delta_j\Big). \end{aligned} \tag{2}$$

In Equation 2, the ray marching step size δ_i equals the distance between the locations of contiguous samples, i.e., $\delta_i = t_{i+1} - t_i$. During ReVolVE training, we use *stratified sampling* along the ray, where points are randomly sampled within evenly spaced intervals, to improve the model convergence during training. We switch to *deterministic sampling* with fixed, evenly spaced points for inference, to ensure consistent results.

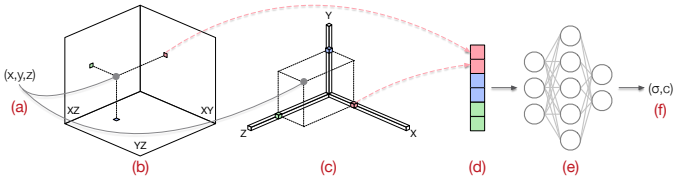


Figure 2: Volume decomposition and feature decoder. ReVolVE takes (a) a location in 3D space and samples on the (b) matrices and (c) vectors to get a feature. These features are (d) concatenated and (e) fed to a neural network decoder to obtain (f) density and color outputs.

3.2. Volume Representation

Vector-matrix decomposition. As discussed in Section 2, using a voxel-based 3D grid to represent the volume is an intuitive method that facilitates fast learning and straightforward volume reconstruction. However, this approach is constrained by substantial memory demands, preventing the reconstruction of high-resolution volumes. Following ViSNeRF [63], we adopt the block-term tensor decomposition [64] to factorize the 3D volume $\mathbf{V} \in \mathbb{R}^{X\times Y\times Z}$ as the sum of vector-matrix outer products [27]

$$\mathbf{V} = \sum_{r=1}^{R_1} \mathbf{M}_r^{XY} \circ \mathbf{v}_r^{Z} + \sum_{r=1}^{R_2} \mathbf{M}_r^{XZ} \circ \mathbf{v}_r^{Y} + \sum_{r=1}^{R_3} \mathbf{M}_r^{YZ} \circ \mathbf{v}_r^{X}, \tag{3}$$

where $\mathbf{M}_r^{XY}$, $\mathbf{M}_r^{XZ}$, and $\mathbf{M}_r^{YZ}$ represent the matrices corresponding to the XY, XZ, and YZ planes, as shown in Figure 2. Similarly, $\mathbf{v}_r^{X}$, $\mathbf{v}_r^{Y}$, and $\mathbf{v}_r^{Z}$ are the vectors along the X, Y, and Z axes, respectively. The term $\mathbf{M}_r^{XY} \circ \mathbf{v}_r^{Z}$ denotes the outer product of the matrix $\mathbf{M}_r^{XY}$ and vector $\mathbf{v}_r^{Z}$. R_1, R_2, and R_3 correspond to the numbers of low-rank components, i.e., $\mathbf{M}_r^{XY} \circ \mathbf{v}_r^{Z}$, $\mathbf{M}_r^{XZ} \circ \mathbf{v}_r^{Y}$, and $\mathbf{M}_r^{YZ} \circ \mathbf{v}_r^{X}$. These numbers can be adjusted based on the complexity and size of the represented volume.

By applying factorization, the memory complexity for representing the radiance field can be reduced from $O(N^3)$ to $O(N^2)$, where N is the resolution and $N \gg R_1 + R_2 + R_3$. We simplify the method by assuming uniform complexity across all three dimensions, using an identical number of components in each dimension, i.e., $R = R_1 = R_2 = R_3$. The factorization can then be formulated as

$$\mathbf{V} = \sum_{r=1}^{R} \mathbf{M}_r^{XY} \circ \mathbf{v}_r^{Z} + \mathbf{M}_r^{XZ} \circ \mathbf{v}_r^{Y} + \mathbf{M}_r^{YZ} \circ \mathbf{v}_r^{X}. \tag{4}$$

Feature decoder. As a trade-off for compactness, the vector-matrix decomposition inevitably sacrifices some expressiveness compared to a voxel-based 3D grid. To enhance the representation's capability, we employ a neural network as a feature decoder. Unlike other radiance field approaches [27, 31], which take point locations and view directions in addition to sampled features from the explicit representation, our decoder uses only sampled features as input. When learning a volume from DVR images, incorporating point locations and view directions into the MLP can cause interference rather than complement the output generation. Relying solely on the features helps improve the robustness of ReVolVE, particularly when working with a limited number of training images.

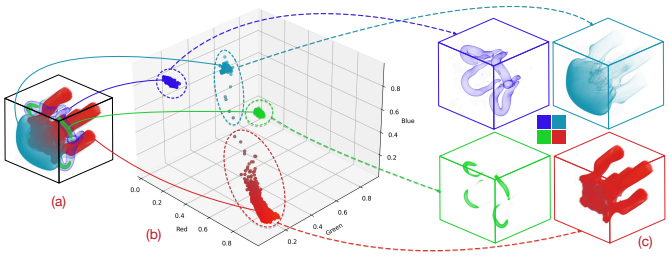


Figure 3: Extraction of representative colors and volume segmentation. (a) RGB values are extracted from the voxels in the color volume. (b) K-means clustering is performed on the sampled colors to identify distinct groups. (c) We use the cluster centers as representative colors, which are used to create masks for volume segmentation.

3.3. Color-Based Volume Segmentation

To deliver visualization enhancement in a flexible and user-friendly way, ReVolVE focuses on enabling the ability to edit different colored regions of unlit DVR images separately. It disentangles the alpha-blended color in DVR images by learning a radiance field, allowing for accurate reconstruction of density and color volumes.

As illustrated in Figure 3, we leverage the color volume to estimate the color distribution within the scene. A clustering algorithm, such as k-means, can be applied to group colors by calculating the Euclidean distances in the RGB space. The centroid of each cluster serves as the representative color. Note that we exclude color samples from points with low density to improve the overall accuracy of the process.

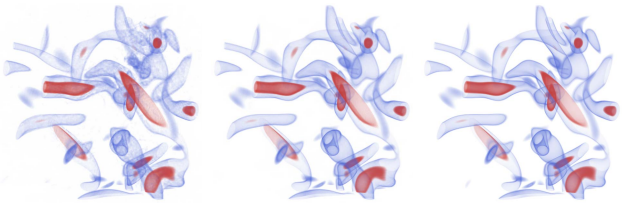
After extracting representative colors, volume segmentation masks are generated by mapping sampled voxels to the closest representative colors. This is determined by calculating the Euclidean distance between the voxel's color and each representative color in the RGB space. As shown in Section 4, our volume segmentation has proven highly effective. Other methods like PaletteNeRF [65] prefer to learn a palette to segment the volume, which would add extra training overhead and may not be an optimal approach for segmenting volumes learned from unlit DVR images.

3.4. Loss Functions

A straightforward way to supervise ReVolVE is to compute a color loss, which measures the difference between the rendered outputs and the training DVR images. For each training batch, a subset of rays $\mathcal{R}$ is randomly selected from the pool of all training image pixels. For each ray $\mathbf{r}$, as described in Equation 2, n samples are queried along the ray, and the corresponding pixel color $\hat{C}(\mathbf{r})$ is predicted. The color loss is then calculated as the mean squared error (MSE) between the predicted colors $\hat{C}(\mathbf{r})$ and the ground truth (GT) colors $C(\mathbf{r})$ for the batch of pixels, and is defined as

$$\mathcal{L}_{\text{RGB}} = \frac{1}{|\mathcal{R}|} \sum_{\mathbf{r}\in\mathcal{R}} \left\| C(\mathbf{r}) - \hat{C}(\mathbf{r}) \right\|_2^2 . \tag{5}$$

However, as shown in Figure 4, without proper regularization, the visualized matrices of ReVolVE and the generated images exhibit significant noise and grainy artifacts. To mitigate



(a) w/o TV (b) with TV (c) GT

Figure 4: TV regularization enables ReVolVE to produce rendering results with less noise and higher accuracy.

them, we leverage a *total variation* (TV) loss as a regularization term alongside the color loss. The TV loss reduces noise in the vectors and matrices by penalizing high variations and is defined as

$$\begin{aligned}
&\mathcal{L}_{\text{TV}} = \mathcal{L}_{\text{TV}_1} + \mathcal{L}_{\text{TV}_2}, \text{where} \\
&\mathcal{L}_{\text{TV}_1} = \frac{1}{|\mathcal{V}||\mathcal{Q}|l} \sum_{\mathbf{v},q,i} (\left\| \mathbf{v}_q^i - \mathbf{v}_q^{i-1} \right\|_2^2) \text{ and} \\
&\mathcal{L}_{\text{TV}_2} = \frac{1}{|\mathcal{M}||\mathcal{Q}|hw} \sum_{\mathbf{M},q,i,j} (\left\| \mathbf{M}_q^{i,j} - \mathbf{M}_q^{i-1,j} \right\|_2^2 \\
&+ \left\| \mathbf{M}_q^{i,j} - \mathbf{M}_q^{i,j-1} \right\|_2^2),
\end{aligned} \tag{6}$$

where $\mathbf{v} \in \mathcal{V}$, $\mathbf{M} \in \mathcal{M}$, and $q \in \mathcal{Q}$. Here, $\mathcal{V}$ represents the set of vectors, $\mathcal{M}$ is the set of matrices, $\mathcal{Q}$ denotes the feature channels, l is the length of the vector, and h and w are the height and width of the plane, respectively. The complete loss function is given by

$$\mathcal{L} = \mathcal{L}_{\text{RGB}} + \lambda \mathcal{L}_{\text{TV}}, \tag{7}$$

where λ is the weight of the TV regularization term.

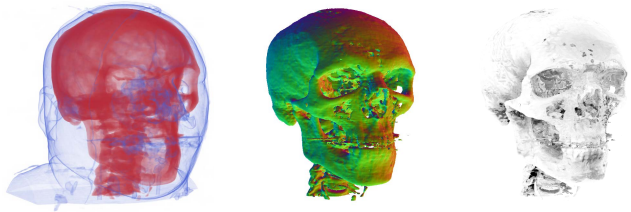
3.5. Neural Visualization Enhancement

ReVolVE includes visualization enhancements within the PyTorch framework, enabling direct rendering with neural volumes and eliminating the efforts of exporting and handling reconstructed volumes for enhanced rendering. The workflow is streamlined and simplified by adapting traditional enhancement techniques for neural representations.

Isosurface enhancement. Isosurface enhancement renders the volumetric scene to appear as a fully opaque isosurface. However, since the original volume data is unavailable, the isovalue corresponds to the density value in the NeRF space rather than the original data value. Consequently, the rendered isosurface should be regarded as approximating an isosurface from the original volume corresponding to a peak in the TF.

To streamline the rendering process, ReVolVE renders the isosurface implicitly using the neural volume instead of rasterizing extracted surface triangles [66]. Specifically, rays are cast towards the neural volume and terminated at the first instance where the density value of the current sample exceeds a predefined threshold while the previous sample's density value is below it. The midpoint between these two sample points is an approximate location on the isosurface.

Figure 5 (d) shows the isosurface-enhanced rendering of a segmented volume. Figures 5 (b) and (c) further illustrate the



(a) DVR (b) derived normal (c) derived AO

(d) upscaling+smoothing (e) w/o upscaling (f) w/o smoothing

Figure 5: Isosurface enhancement applied to the red region of the head dataset. (a) DVR input. (b) Surface normals where XYZ directions are colored as RGB. (c) AO volume. (d) Isosurface enhancement with Blinn-Phong lighting, AO, and feature grid upscaling and smoothing. (e) With smoothing but no upscaling. (f) With upscaling but no smoothing.

estimated surface normals and *ambient occlusion* (AO) derived from the neural volume. As illustrated in Figure 5 (d), (e), and (f), ReVolVE leverages neural representation to minimize artifacts and enhance surface smoothness through feature grid upscaling and smoothing. Detailed explanations of these techniques are provided in Appendix A.

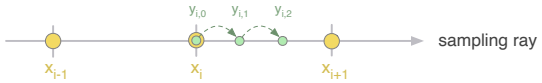


Figure 6: Local front-to-back averaging for depth enhancement. Yellow points are consecutive primary sample points along the ray, and green points denote intermediate sample points for averaging. Green arrows indicate front-to-back accumulation.

Depth enhancement. As explained in [6], we can see the inner structure when rendering the volume semi-transparently with DVR, but the depth-ordering perception is ambiguous. Following the shader implementation in [67, 68], during the DVR process, as illustrated in Figure 6, we perform a local front-to-back averaging within each ray segment. Such a segment is the interval between two consecutive sample points along the ray. As more steps are taken in the ray segment averaging process, the density at each sample point increases, making the visible volume appear less transparent. Low-density regions appear darker, creating an AO-like effect, and high-density regions become more distinguishable. This technique is formulated as

$$\begin{aligned} &\sigma(\mathbf{x}_i) = \sum_{j=0}^{D} w(\mathbf{y}_{i,j}) \text{ and } c(\mathbf{x}_i) = \sum_{j=0}^{D} c(\mathbf{y}_{i,j}) w(\mathbf{y}_{i,j}), \text{ where} \\ &w(\mathbf{y}_{i,j}) = \sigma(\mathbf{y}_{i,j}) \Big(1 - \sum_{k=0}^{j-1} w(\mathbf{y}_{i,k})\Big) \text{ and } w(\mathbf{y}_{i,0}) = \sigma(\mathbf{y}_{i,0}), \end{aligned} \tag{8}$$

where $\mathbf{x}_i$ is the i-th primary sample point along the ray, $\mathbf{y}_{i,j}$ is the j-th secondary sample point starting from $\mathbf{x}_i$. D denotes the number of sampling steps, i.e., the number of secondary sample points taken for each primary sample point. Functions σ and c denote the density and color at the sample point location. The weighted density function w determines the contribution of the density at each secondary point to the accumulated density at the corresponding primary point. It also controls how the color at each secondary point affects the color at the primary point. Additionally, we incorporate the same lighting method into the rendering, as higher density causes the volume to resemble semi-transparent surfaces with perceptible thickness. The normals and lighting at primary sample points are computed after averaging.

(a) Instant-NGP (b) ReVolVE (c) GT

Figure 7: The surface extracted from a volumetric region of the chameleon dataset using Instant-NGP and ReVolVE. The GT is an isosurface obtained at the isovalue representing the region.

3.6. Volume and Surface Extraction

For volume extraction, we first sample a feature grid at the desired resolution. The sampled grid is then decoded to obtain the density and color volumes, following the procedure described in Figure 2. The density volume is segmented into multiple volumes based on the representative colors (refer to Section 3.3) and exported to individual files. This enables users to render segmented density volumes with preferred tools for greater visualization flexibility.

ReVolVE can also extract a surface from each segmented density volume using the marching cubes algorithm [69]. During this process, feature grid upscaling and smoothing are applied. As shown in Figure 7, the surface generated by ReVolVE demonstrates greater detail and clarity than that by Instant-NGP, both reconstructed from a 512^3 resolution volume. ReVolVE excels at capturing finer details, particularly in regions such as the eye socket and head crest. This makes it reliable for reconstructing volumes and surfaces from DVR images.

4. Results and Discussion

In this section, we provide details of the datasets and network training, and then discuss the baselines and metrics employed in the comparison. We present ReVolVE's novel view synthesis results and compare them with baseline methods. Next, we showcase visualization enhancement results enabled by ReVolVE. Finally, we point out the limitations of ReVolVE. The ablation and hyperparameter study results are furnished in Appendix B and Appendix C.

4.1. Datasets and Network Training

Table 1 lists the datasets for evaluating ReVolVE and comparing it with baseline methods. These datasets are also used to showcase visualization results enhanced by ReVolVE. The sizes of these datasets range from small (8 MB) to large (4.2 GB). To achieve an even distribution of training viewpoints around the volume data, the camera positions are determined using the spherical Fibonacci point set [70], which employs a Fibonacci spiral to ensure a near-uniform arrangement without clustering. Doing this allows for selecting an arbitrary number of viewpoints, enabling us to choose an optimal number to learn a neural volume representing the scene effectively.

Table 1: Dataset details: data and training image information.

dataset	volume resolution	volume size (MB)	# training images	image size (MB)
vortex	128×128×128	8	6	1.53
five jets	128×128×128	8	12	3.50
head	256×256×256	64	12	5.26
supernova	432×432×432	308	18	8.48
combustion	480×720×120	158	18	6.02
chameleon	1024×1024×1080	4320	18	7.99

In all cases, we use a resolution of 1024×1024 for DVR images in training and inference. The training images are produced from generic color TFs (e.g., blue-red for vortex, head, supernova, and combustion, blue-green-cyan-red for five jets, and blue-green-red for chameleon) without any lighting applied. Alternative color schemes in TFs for the five jets and combustion datasets are used to assess the generalizability of color-based segmentation. As reported in Table 1, the number of training images ranges from six for a simple scene to 18 for a complex one. With the same number, the image size varies as they are saved in PNG format. To perform 360-degree inference, we create 181 new views following a spiral path on a spherical surface, beginning at an azimuth of -180 and elevation of -90, progressing through azimuth 0 and elevation 0, and concluding at an azimuth of 180 and elevation 90.

Table 2: Model configuration: the maximum number of voxels in the vector-matrix feature grid and the number of low-rank components R.

dataset	vortex	five jets	head	supernova	combustion	chameleon
max # voxels	128^3	128^3	256^3	320^3	384^3	384^3
R	8	16	16	16	24	16

ReVolVE is implemented using PyTorch without relying on custom CUDA kernels. All training is conducted on a single NVIDIA A40 GPU with 48 GB of video memory. We train ReVolVE for 30,000 iterations for each dataset. The process begins with a feature grid of 128^3 voxels, which expands at 2,000 iterations and reaches the maximum voxel count by the 7,000-th iteration. The number of sample points along the ray is adjusted according to the voxel count, allowing for efficient and detailed rendering. At 2,000 iterations, the bounding box is refined, and by 4,000 iterations, an alpha mask is generated to skip empty space, further improving training efficiency.

The MLP decoder in ReVolVE consists of a single hidden layer with 64 channels, remaining consistent across all experiments. We use the Adam optimizer, with an initial learning rate of 0.02 for the learnable features (vectors and matrices) and 0.001 for the MLP decoder. Both training and inference use the same batch size (4096) to balance performance and memory usage. Additionally, TV regularization with $\lambda = 1.0$ is applied to smooth the features. Table 2 reports the maximum number of voxels in the feature grid and the number of low-rank components R in volume factorization (i.e., the depth of features sampled from the vectors and matrices). As discussed in Appendix B, these values are empirically determined by finding the point where further parameter increases no longer improve quality significantly. We observed that a higher-resolution volumetric scene does not always require additional voxels or a larger R, as factors like scene complexity and variation could also play a role.

4.2. Baselines and Metrics

We choose DiffDVR along with four representative radiance field rendering methods as baselines for comparison.

- DiffDVR [28] makes the volume rendering process differentiable, allowing for optimization of parameters like viewpoint, transfer function, and voxel value.
- Plenoxels [29] uses a sparse voxel grid to represent radiance fields without using neural networks, enabling fast training and rendering while maintaining the same level of fidelity as NeRF.
- 3DGS [30] utilizes Gaussian ellipsoids to represent radiance fields efficiently and achieves faster optimization and real-time rendering with high visual quality.
- TensoRF [27] employs tensor factorization to accelerate NeRF and improve the quality of image details.
- Instant-NGP [31] leverages MHT for fast training and real-time performance for neural scene representations.

We evaluate the quality of synthesized images by *peak signal-to-noise ratio* (PSNR), *structural similarity index measure* (SSIM), and *learned perceptual image patch similarity* (LPIPS) [71]. To ensure fairness, we increase the model size of all baseline methods to enhance their overall performance when the scene complexity increases. As the complexity grows, the configurations of baseline models are adjusted in the following ways: Plenoxels uses a higher resolution for its sparse voxel grid; 3DGS increases the density of Gaussians; Instant-NGP boosts the feature grid resolution and the hash table size; and TensoRF, while matching ReVolVE in feature grid resolution and depth, maintains an approximately doubled model size compared to ReVolVE, due to the use of separate grids for density and color. The model configurations and training settings are reported in Table 3. For 3DGS, the maximum number of Gaussians ranges from 100,000 to 800,000, while for Instant-NGP, the size of the hash table is set between 2^{19} to 2^{22}, both adjusted based on dataset complexity. All other hyperparameters remain the same as the default settings provided in the official code repository. For example, Instant-NGP employs multiresolution hash grids with resolutions ranging from 16^3 to 2048^3 across eight levels.

We are aware of the modified 3DGS variant, *cinematic Gaussian splatting* (CGS) [53], designed for photorealistic path-traced images. In our experiments with unlit DVR datasets, including supernova and chameleon, CGS does not outperform the standard 3DGS, so we only show comparison results based on the vanilla 3DGS. Please refer to Appendix D for a detailed comparison among 3DGS, CGS, and ReVolVE.

Table 3: Novel view synthesis: model configurations and training settings for baselines.

method	max # features	feature depth	training length	batch size
DiffDVR	256^3 to 512^3	4	50 epochs	1 view
Plenoxels	256^3 to 512^3	10	128k steps	5,000 rays
3DGS	100k to 800k	59	30k steps	1 view
TensoRF	128^3 to 384^3	8 to 24	30k steps	4,096 rays
Instant-NGP	2^{19} to 2^{22}	4	35k steps	256k samples

Table 4: Novel view synthesis: average PSNR (dB), SSIM, and LPIPS values across all 181 synthesized views, and training time (TT, in minutes), average inference time (IT, in seconds) of a view, and model size (MS, in MB). The best ones are highlighted in bold.

dataset	method	PSNR↑	SSIM↑	LPIPS↓	TT↓	IT↓	MS↓
vortex	DiffDVR	31.81	0.938	0.088	22.38	1.91	256.09
	Plenoxels	27.55	0.908	0.185	11.08	0.44	91.90
	3DGS	21.20	0.860	0.275	5.70	**0.28**	18.61
	TensoRF	33.14	0.970	0.038	16.42	2.92	3.43
	Instant-NGP	32.31	0.955	0.099	**2.92**	0.71	23.54
	ReVolVE	**45.56**	**0.994**	**0.007**	7.60	2.47	**1.81**
five jets	DiffDVR	25.31	0.935	0.110	26.57	2.98	256.09
	Plenoxels	22.22	0.918	0.107	13.17	0.42	93.38
	3DGS	26.52	0.951	0.072	7.88	**0.22**	26.03
	TensoRF	29.23	0.964	0.050	16.42	2.68	6.49
	Instant-NGP	31.86	0.973	0.046	**2.02**	1.01	23.12
	ReVolVE	**39.08**	**0.991**	**0.009**	11.43	2.55	**3.37**
head	DiffDVR	29.55	0.928	0.138	25.30	1.58	256.09
	Plenoxels	23.22	0.860	0.164	12.98	0.56	559.74
	3DGS	26.04	0.895	0.148	9.23	**0.40**	46.87
	TensoRF	30.60	0.929	0.090	38.22	6.41	24.94
	Instant-NGP	32.14	0.936	0.116	**3.42**	1.64	24.23
	ReVolVE	**38.38**	**0.979**	**0.032**	16.45	3.71	**12.68**
supernova	DiffDVR	31.54	0.931	0.118	97.33	9.47	2048.09
	Plenoxels	29.17	0.903	0.120	33.83	1.09	839.27
	3DGS	29.39	0.907	0.093	12.83	**0.36**	89.62
	TensoRF	31.78	0.923	0.096	55.18	6.41	38.79
	Instant-NGP	34.36	0.946	0.083	**3.58**	1.44	41.15
	ReVolVE	**36.57**	**0.957**	**0.046**	25.15	4.54	**19.73**
combustion	DiffDVR	25.85	0.912	0.102	97.28	8.10	2048.09
	Plenoxels	26.12	0.923	0.062	30.65	0.90	611.75
	3DGS	29.38	0.956	0.033	10.80	**0.30**	99.94
	TensoRF	32.71	0.968	0.023	40.60	3.87	100.40
	Instant-NGP	34.54	0.982	0.019	**4.98**	1.23	110.96
	ReVolVE	**39.48**	**0.993**	**0.004**	28.05	3.69	**42.66**
chameleon	DiffDVR	28.06	0.912	0.082	96.88	9.25	2048.09
	Plenoxels	29.66	0.925	0.051	31.83	0.98	2057.75
	3DGS	29.90	0.932	0.037	17.57	**0.30**	195.73
	TensoRF	32.21	0.952	0.026	32.38	3.87	61.45
	Instant-NGP	32.92	0.962	0.020	**6.15**	1.26	207.01
	ReVolVE	**36.04**	**0.979**	**0.010**	27.75	3.76	**30.09**

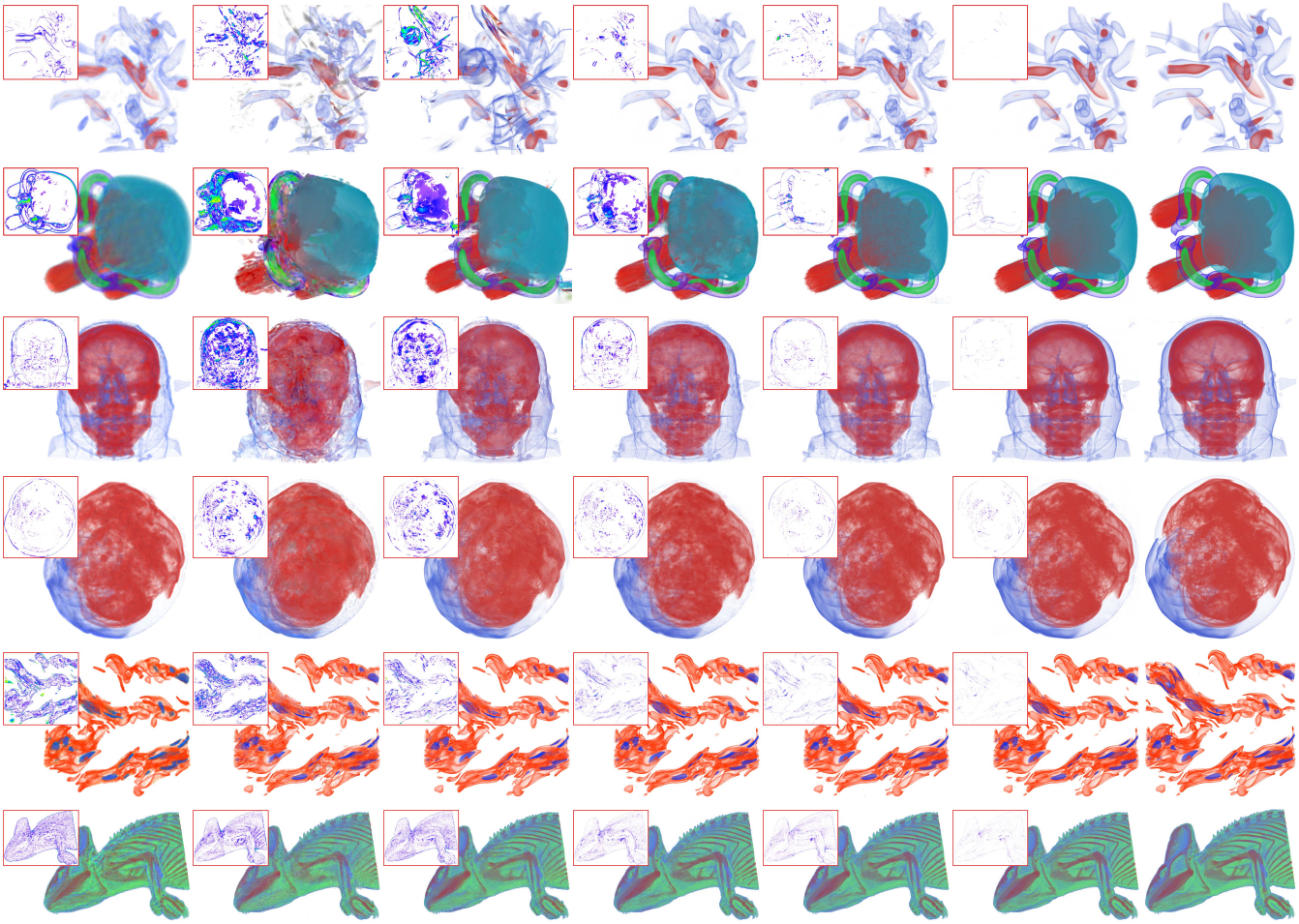
4.3. Novel View Synthesis

We begin our evaluation with DiffDVR, as it offers a relatively straightforward approach for learning density and color volumes from unlit DVR images. As shown in Figure 8 (a), while the rendered scenes are structurally accurate, they appear noticeably blurry across all datasets. This is partly because, unlike the original DiffDVR setup that uses 64 training views, our experiments rely on fewer views. Additionally, the limited expressiveness of the explicit voxel grid representation contributes to the lack of fine detail. Beyond reconstruction quality, DiffDVR also presents less desirable training speed, inference speed, and most importantly, model size. For example, when optimizing volumes at higher resolutions such as 512^3, the GPU memory usage exceeds 28 GB. This high memory consumption stems from its explicit volumetric representation, which poses a significant limitation to scalability when reconstructing high-resolution volumes.

Similar pattern could be observed in Plenoxels, which also uses a sparse voxel grid to represent 3D volumes. The reliance on an explicit voxel grid limits its capacity to capture fine-grained details, particularly in complex scenes, and leads to significantly larger model sizes compared to ReVolVE. In contrast to DiffDVR, Plenoxels incorporates several strategies aimed at mitigating its inherent limitations. Plenoxels tries to improve image quality with fewer training views using TV regularization, but as suggested by the results in Table 4, if training views are insufficient, it struggles to reconstruct the volumes accurately and produces inferior rendering images. Another drawback of Plenoxels lies in model size. Even skipping empty parts of the voxel grid, the size is still quite large compared to ReVolVE and other methods. Additionally, as highlighted in Figure 8 (b), it is difficult for Plenoxels to handle dense semi-transparent regions, such as the cyan dome in the five jets dataset and the red skull in the head dataset.

3DGS has recently gained significant attention, inspiring numerous follow-up works. Initially, we anticipated leveraging its potential for real-time rendering. However, our experiments reveal that representing volumes with 3D Gaussians is challenging. Unlike surfaces, where Gaussians can efficiently and accurately capture surface details, continuous and densely semi-transparent volumes require a complete filling of space with color and density values to be properly represented. Filling the entire volume with millions of Gaussians, however, is inefficient and will drastically reduce the efficiency of 3DGS. As a workaround, 3DGS tends to concentrate Gaussians on the surface of visible regions and relies on view-dependent coloring to simulate the internal structure of semi-transparent regions. The cyan dome in the five jets dataset, shown in Figure 8 (c), exemplifies this limitation. As another example, the supernova dataset illustrates how 3DGS attempts to fill the volume with Gaussians but struggles to capture the correct structure. Thus, while 3DGS can effectively depict surfaces, it is not ideal for accurate volume reconstruction or visualization improvement with plain, unlit DVR training images.

Our next candidate, TensoRF, has been selected for its compact and efficient representation of the radiance field through vector-matrix decomposition. Our experiments show its strong stability with just six training views on the vortex dataset, and TensoRF handles complex scenes with good overall quality. However, one issue we encountered was the blurriness of rendered images. While TensoRF supports TV regularization, its hybrid design brings some complications. Since the neural network also uses the sample point's location as input, the impact of TV regularization is reduced, as it only applies to features



(a) DiffDVR (b) Plenoxels (c) 3DGS (d) TensoRF (e) Instant-NGP (f) ReVolVE (g) GT

Figure 8: Novel view synthesis from unlit DVR images produced from generic TFs. Top to bottom: vortex, five jets, head, supernova, combustion, and chameleon. The corner images indicate the perceivable pixel-wise differences (ranging from purple to green for low to high differences).

in the vectors and matrices. The location input tends to dominate, making it harder for the network to utilize the regularized features in the vectors and matrices fully. This mismatch can result in conflicts, especially when capturing fine details in the volume, as illustrated in Figure 8 (d). To overcome this, we use only the features sampled from the vectors and matrices as inputs to the neural network. This adjustment enhances the model's stability and reduces computational cost.

For Instant-NGP, its MHT encoding significantly boosts training and rendering speeds, as shown in Table 4. Despite its highly efficient architecture, it struggles with stability when training views are insufficient. We must adjust training configurations and run multiple attempts to ensure proper convergence and achieve high-quality results. When it does converge correctly, the performance is impressive, especially considering the short training time. However, a huge limitation of the hash grid is that it cannot apply TV regularization to reduce the required training views. TV regularization requires a tensor-based representation, such as a volume or a plane, whereas Instant-NGP stores its features in the MHT. Additionally, as seen in Figure 8 (e), the thin blue layer in the supernova dataset appears incomplete on the right side. This issue arises in Instant-NGP because voxels corresponding to small or distant objects are visited infrequently by training rays, resulting in fewer gradient updates. Due to the limited capacity of the hash table, these infrequently accessed entries are also more likely to be overwritten by those corresponding to dominant, high-frequency regions. As a result, small objects located far from the main scene structures tend to be undertrained or ignored during optimization. As the training speed improves with an increased learning rate, such an issue occurs more frequently.

Figure 8 (f) shows that the visual quality of DVR images synthesized by ReVolVE is the best. While most methods struggle to accurately render the scene with only six images of the vortex dataset, ReVolVE produces visualizations nearly identical to the GT. As shown in Table 4, ReVolVE achieves noticeably higher PSNR values for the synthesized images than other methods across all datasets. This suggests that ReVolVE requires only a small set of images to reconstruct accurate density and color volumes. In contrast, other methods typically need several times more images to ensure results with similar fidelity. For example, our experiment shows that Instant-NGP needs about 54 training views of the chameleon dataset to deliver a similar rendering quality to ReVolVE, which only needs

18 views. We also want to point out that ReVolVE offers excellent model compactness, with sizes as small as under 2 MB when representing a 128^3 volumetric scene and around 30 MB for a $1024 \times 1024 \times 1080$ scene, all while maintaining a rendering quality above 35 dB in PSNR. Therefore, ReVolVE is excellent for accurately reconstructing volumes and synthesizing high-quality DVR images, making it well-positioned for applying subsequent enhancement options.

4.4. SRNs vs. ReVolVE

Since ReVolVE is closely related to SRNs, we include a comparison between ReVolVE and three SRNs for volume visualization, including fV-SRN [22], APMGSRN [24], and Instant-VNR [25]. To best highlight the differences, we use supernova and chameleon, the two most complex datasets in our experiments. In addition to metrics that evaluate the quality of the rendered images, we also compare training time, single-frame inference time, and model size. ReVolVE uses the same training configuration described in Section 4.3, so the reported results remain consistent. For fV-SRN and APMGSRN, we evaluate both the base and large versions to provide a more comprehensive assessment of their performance, as their feature grid resolutions are relatively small. The configurations for both the base and large models follow those specified in the original papers [22, 24]. For Instant-VNR, we use the default configuration provided by the original code, which has a resolution sufficient to handle both datasets. The details of model configurations and training settings are summarized in Table 5.

Table 5: SRNs: model configurations and training settings.

method	fV-SRN (base/large)	APMGSRN (base/large)	Instant-VNR
feature grid resolution	32^3/64^3	32^3/64^3	16^3 to 2048^3
feature grid depth	16	2	8
# of feature grids	1	16/32	8
# of hidden layers in MLP	4/2	2	4
# of channels in each layer	32/128	64	64
hash table size	–	–	2^{19}
training length	200 epochs	100k steps	1,000k steps
batch size	524,288	1,048,576	65,536

Table 6: SRNs vs. ReVolVE: average PSNR (dB), SSIM, and LPIPS values across all 181 synthesized views, and training time (TT, in minutes), average single-frame inference time (IT, in seconds) of a view, model size (MS, in MB), as well as CPU memory (RAM, in GB) and GPU memory usage (VRAM, in GB). The best ones are highlighted in bold.

dataset	method	PSNR↑	SSIM↑	LPIPS↓	TT↓	IT↓	MS↓	RAM↓	VRAM↓
supernova	fV-SRN (base)	31.45	0.923	0.051	2.67	6.08	**2.02**	5.54	3.37
	fV-SRN (large)	33.03	0.948	0.049	4.87	9.76	16.09	5.52	4.81
	APMGSRN (base)	32.18	0.934	0.056	**0.48**	0.80	4.03	3.13	**1.75**
	APMGSRN (large)	33.31	0.943	0.055	1.25	0.86	64.04	3.12	2.13
	Instant-VNR	31.35	0.939	0.049	2.02	**0.38**	46.72	**0.72**	4.73
	ReVolVE	**36.57**	**0.957**	**0.046**	25.15	4.54	19.73	3.62	5.13
chameleon	fV-SRN (base)	30.05	0.936	0.052	2.60	4.28	**2.02**	9.54	8.18
	fV-SRN (large)	31.48	0.948	0.075	4.92	9.77	16.09	9.59	9.55
	APMGSRN (base)	29.58	0.933	0.040	**0.50**	0.92	4.03	8.62	**5.76**
	APMGSRN (large)	30.88	0.948	0.029	1.38	1.12	64.04	8.67	6.15
	Instant-VNR	31.03	0.952	0.023	1.98	**0.37**	46.72	6.33	7.73
	ReVolVE	**36.04**	**0.979**	**0.010**	27.75	3.76	30.09	**3.49**	6.07

From the results shown in Table 6, we observe that ReVolVE outperforms all SRNs in terms of PSNR, SSIM, and LPIPS, indicating superior rendering quality. This improvement stems from ReVolVE's approach of learning the volumetric scene from DVR-generated images, with optimization based on pixel-wise image loss. In contrast, SRNs learn directly from the original volumetric data and are optimized based on voxel-wise errors. While voxel errors may be numerically small (all above 40 dB in PSNR of the reconstructed volume compared to the original volume), they often lead to significant discrepancies in the rendered images, especially for high-resolution datasets like supernova and chameleon. As Figure 9 suggests, SRN-generated results tend to appear blurry and less geometrically accurate. In contrast, ReVolVE demonstrates a stronger ability to reconstruct the visible volumetric scene with higher fidelity to the original visualization.

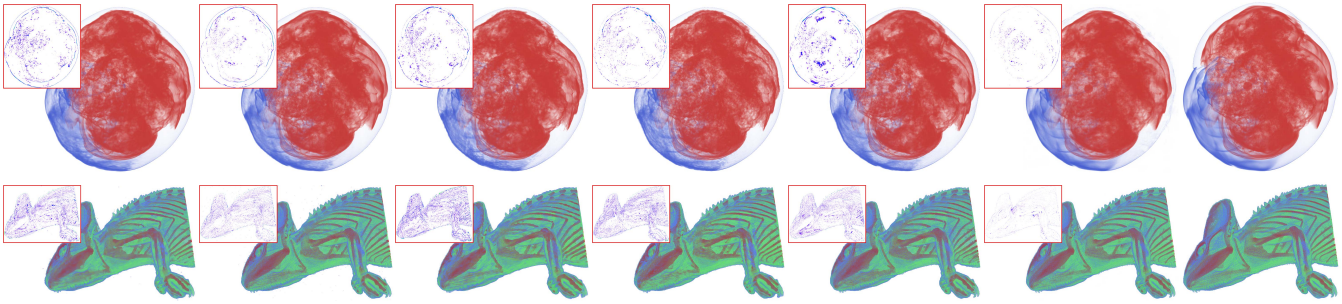
As indicated in Table 6, the CPU memory (RAM) and GPU memory (VRAM) usage of SRNs scales with the size of the volume data. This trend is evident in our tests on supernova and chameleon, where larger volumes result in significantly higher RAM and VRAM usage. Unlike SRNs, ReVolVE's RAM usage is determined solely by the size of the DVR images, which is only indirectly influenced by the size of the original volume data. Notably, increasing image resolution or the number of views typically results in a much smaller image size increase compared to the substantial growth in volume data size. Its VRAM usage is driven mainly by the model size, specifically the feature grid resolution and the depth of the feature grid (i.e., the number of feature channels). Notably, for the chameleon dataset, even though fV-SRN maintains a lightweight design by limiting its feature grids to 32^3 in the base model, its VRAM usage can still surpass that of ReVolVE, which supports a 384^3 feature grid at the same feature depth. ReVolVE also demonstrates lower VRAM usage compared to Instant-VNR and the large model of APMGSRN on the chameleon dataset.

However, we must acknowledge the strengths of SRNs. They generally offer faster training speed, and more importantly, they retain the original data fidelity, allowing the use of TFs and other rendering techniques with full control and confidence. APMGSRN and Instant-VNR can achieve interactive rendering speeds for 1024×1024 images, which is significantly faster than ReVolVE. Still, ReVolVE provides a promising alternative that relies solely on rendered images, making it especially valuable in scenarios where the original volume data is unavailable or too large to store immediately after every simulation run.

4.5. Visualization Enhancement

With synthesized novel views, we further apply visualization enhancements to each dataset, demonstrating the practical applications of ReVolVE. This special feature of ReVolVE uses the neural representation of the volume learned from DVR images. It can work effectively because the learned volume is high-fidelity, allowing faithful application of visualization enhancements. We did not implement this feature for baseline methods, as they are not optimized for accurate volume reconstruction. In Section 4.3, we demonstrate that ReVolVE outperforms the baselines in synthesized novel views.

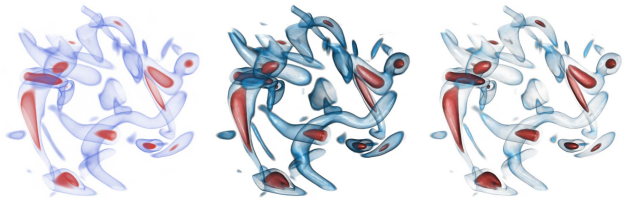
All results presented in this section are neural visualization enhancement outcomes, except for the chameleon dataset,



(a) fV-SRN (base) (b) fV-SRN (large) (c) APMGSRN (base) (d) APMGSRN (large) (e) Instant-VNR (f) ReVolVE (g) GT

Figure 9: Rendering results of representative SRNs and ReVolVE. Top and bottom: supernova and chameleon. The corner images indicate the perceivable pixel-wise differences (ranging from purple to green for low to high differences).

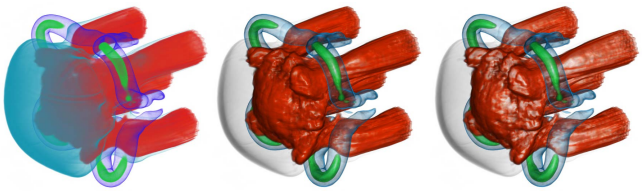
where we present a mixed visualization from extracted density volumes via conventional rendering techniques.



(a) before (b) after (c) after (lower opacity)

Figure 10: Visualizations of the vortex dataset enhanced by ReVolVE.

Vortex. As shown in Figure 10, we aim to demonstrate depth ordering in these DVR images by applying depth enhancement techniques. After enhancing the depth of the blue and red regions, we observed that the blue region appeared too thick, obscuring the inner red region. Therefore, we reduce the opacity of the blue region and sharpen its shape by squaring the opacity, which makes it appear thinner. Furthermore, we add Blinn-Phong lighting to both regions, shading them like surfaces to convey the volume's shape better. As a result, the overall perceptual quality of the visualization is improved.



(a) before (b) after (c) after (more highlights)

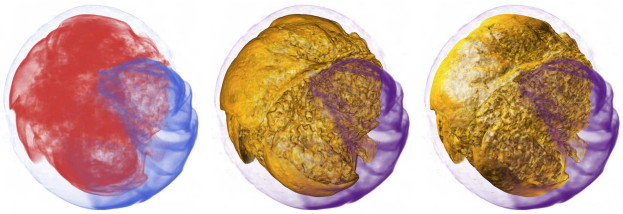
Figure 11: Visualizations of the five jets dataset enhanced by ReVolVE.

Five jets. As shown in Figure 11 (a), since the cyan dome is too dense and blocks other regions in the DVR result, we lower the opacity of this region. We recolor the dome to grey to further lower its interference with the red color region beneath. The blue region, which initially had a darker tone after applying depth enhancement, is brightened to cyan for improved contrast and clarity. Additionally, lighting is introduced to highlight the red region, which previously lacked details at the top. With the lighting applied, the surface-like features of the red region (five jets) become much more apparent, offering a better depth perception of the overall structure (Figure 11 (b)). We also demonstrate an example where the red region is assigned a smaller shininess value α, thus yielding more specular highlights (Figure 11 (c)).

(a) before (b) after (c) after (different view)

Figure 12: Visualizations of the head dataset enhanced by ReVolVE.

Head. For a CT scan like the head dataset shown in Figure 12, our focus is typically on the internal structures, such as the skull, colored in red in the original unlit DVR images. Since the skull is naturally perceived as a solid object, we use isosurface enhancement for display, adding AO and feature grid upscaling and smoothing. The surface shown in Figure 12 retains a smooth appearance yet still preserves clear details, such as the teeth and joints. Moreover, we change the blue color to a skin tone for clarity, providing a better sense of the head's contour.

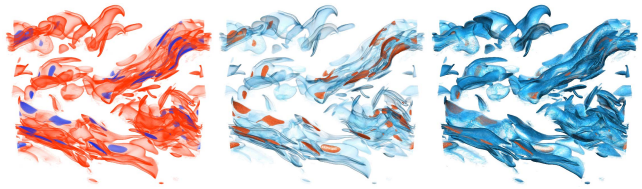


(a) before (b) after (c) after (custom light)

Figure 13: Visualizations of the supernova dataset enhanced by ReVolVE.

Supernova. The supernova dataset exhibits a high level of complexity. To better visualize the internal structure of the red region with depth ordering, we apply depth enhancement specifically to that region. As illustrated in Figure 13, illuminating a semi-transparent volume with Blinn-Phong lighting causes the internal structure to appear differently when the light source is moved (refer to (b) with a headlight and (c) with a custom light), providing additional visual cues. We also adjust the red region to a brighter shade of yellow to increase the contrast. Meanwhile, the blue region is changed to purple, as

purple and yellow are complementary colors that create a strong visual contrast and make each other stand out more.



(a) before (b) after (c) after (higher opacity)

Figure 14: Visualizations of the combustion dataset enhanced by ReVolVE.

Combustion. As shown in Figure 14, the combustion dataset contains numerous thin, semi-transparent layers that overlap, making it challenging to discern their spatial relationships, especially when the layers are in the same color. To address this, we swap the two colors and apply depth enhancement to introduce a subtle color gradient to the outer regions. This adjustment helps better distinguish individual layers. Additionally, depth enhancement makes the inner regions appear more solid and easier to see. We also present an example where increasing the overall opacity enhances the outer blue regions' lighting effect but reduces the inner red regions' visibility.

(a) DVR (b) flesh & skeleton (c) skin

Figure 15: Visualizations of the reconstructed volume (flesh) and isosurfaces (skeleton and skin) of the chameleon dataset using ReVolVE and rendered via ParaView.

Chameleon. We use the chameleon dataset to demonstrate the quality of both the reconstructed volumes and the surface extracted using the marching cubes algorithm [69]. Like the head dataset, we represent the skeleton as a surface by extracting it from the red region's density volume. Unlike the head dataset, which uses isosurface enhancement without extracting surface geometry, for this dataset, an isosurface representing the skeleton is extracted for subsequent rendering via conventional rendering techniques. The green region is exported as a volume, partially covering the skeleton to represent the flesh. As shown in Figure 15, we use ParaView to render the combined scene, applying its global illumination model to enhance the visual effects. The enhanced visualizations make its detailed texture visible, and the skeleton stands out with a smooth surface. We also present an isosurface extracted from the blue region of the volume, representing the skin, which reveals fine details. These results show that ReVolVE reliably reconstructs high-quality density volumes, giving users the flexibility to render them using preferred tools.

Summary. These scenarios show that ReVolVE can enhance the visualizations in various ways using synthesized, plain DVR images as the starting point. Its ability to enhance and fine-tune specific regions offers high flexibility, allowing for targeted adjustments. This makes ReVolVE a versatile solution for refining and enhancing DVR results, especially when the original volume data is unavailable or the rendering cannot be easily recreated. For large datasets (e.g., the chameleon dataset), the ratio between raw volume and training image sizes can be significant (540× for chameleon), making it particularly meaningful for efficient visualization enhancement via ReVolVE.

4.6. Limitations

ReVolVE outperforms the selected baselines in generation quality and model size across all datasets, showcasing its superior performance. It is fully implemented in the PyTorch framework, providing easy integration and strong adaptability for future deep learning advances, including semantic volume segmentation and text-guided visualization enhancement. However, this choice brings a limitation of slower rendering speed than CUDA-based methods like Plenoxels, 3DGS, and Instant-NGP. The underlying reason is the difference in strategies for parallel ray computation. In the CUDA framework, those models compute individual rays across different cores and can terminate early when the accumulated alpha reaches its upper bound, significantly enhancing inference speed. In contrast, the PyTorch-based implementation processes ray batches simultaneously, necessitating the computation of all sample points without the possibility of early termination. Future work may focus on implementing ReVolVE within a CUDA framework to enhance rendering speed and user experience.

Another limitation of ReVolVE is its reliance on emission-absorption DVR for accurate volume reconstruction. Any pre-applied enhancements in the training images will likely distort the reconstruction quality, preventing accurate reconstruction of the original volumetric scene before the enhancements. Moreover, we assume the DVR images are generated using a TF designed to depict meaningful regions clearly, even though the images are plain and unlit. ReVolVE would not know any data content where the corresponding value ranges are of zero opacity and, therefore, could not reconstruct it in the resulting neural volume. Given another set of DVR images for the same volumetric dataset but under a different TF highlighting previously unseen content, ReVolVE needs retraining.

5. Conclusions and Future Work

We have presented ReVolVE, a novel framework designed to enhance the visualization of plain DVR images without requiring access to the original volume data. Leveraging a NeRF with vector-matrix decomposition, ReVolVE efficiently reconstructs the volumetric scene from unlit DVR inputs. Compared to other mainstream NeRF-based methods, ReVolVE achieves significantly higher performance, even with limited training views. Once trained, ReVolVE can extract representative colors from the reconstructed volume and create segmentation masks for targeted volume editing and visual enhancements. ReVolVE implements various visualization enhancement options that fully leverage NeRF. ReVolVE can also employ optimization techniques like feature grid upscaling and smoothing

to enhance rendering quality while exporting high-quality density volumes and isosurfaces. Therefore, ReVolVE provides a reliable and versatile way for high-fidelity visualization enhancements of unlit DVR images in the NeRF space.

Despite the success of ReVolVE, we envision four promising future directions. First, we will reimplement ReVolVE within the CUDA framework to accelerate training and rendering speeds. Alternatively, to enable real-time rendering, we could also leverage 3DGS by optimizing it for learning from unlit DVR images and incorporating enhancement features into the 3D Gaussian rasterization process. With real-time rendering, we aim to create a toolkit with a graphical user interface for ReVolVE that supports interactive visualization exploration. It will include widgets for tuning the visualization enhancement settings as well as basic volume operations such as clipping, slicing, segmentation, and color mapping. Second, integrating advanced segmentation methods could allow for semantic segmentation and targeted visual enhancements, particularly for scientific and medical volume data. Third, our framework could be extended to dynamic scenes, offering the potential to apply consistent visualization enhancements to time-varying volumetric data. Finally, since ReVolVE's visualization enhancement options are fully implemented within the PyTorch framework, we aim to explore the potential of integrating declarative languages to streamline the process.

Acknowledgements

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] N. L. Max, Optical models for direct volume rendering, IEEE Transactions on Visualization and Computer Graphics 1 (2) (1995) 99–108. doi:10.1109/2945.468400.

[2] J. Kniss, S. Premoze, C. D. Hansen, P. Shirley, A. McPherson, A model for volume lighting and modeling, IEEE Transactions on Visualization and Computer Graphics 9 (2) (2003) 150–162. doi:10.1109/TVCG.2003.1196003.

[3] E. B. Lum, K.-L. Ma, Lighting transfer functions using gradient aligned sampling, in: Proceedings of IEEE Visualization Conference, 2004, pp. 289–296. doi:10.1109/VISUAL.2004.64.

[4] S. Bruckner, M. E. Gröller, Enhancing depth-perception with flexible volumetric halos, IEEE Transactions on Visualization and Computer Graphics 13 (6) (2007) 1344–1351. doi:10.1109/TVCG.2007.70555.

[5] M.-Y. Chan, Y. Wu, W.-H. Mak, W. Chen, H. Qu, Perception-based transparency optimization for direct volume rendering, IEEE Transactions on Visualization and Computer Graphics 15 (6) (2009) 1283–1290. doi:10.1109/TVCG.2009.172.

[6] L. Zheng, Y. Wu, K.-L. Ma, Perceptually-based depth-ordering enhancement for direct volume rendering, IEEE Transactions on Visualization and Computer Graphics 19 (3) (2013) 446–459. doi:10.1109/TVCG.2012.144.

[7] P. Gu, J. Han, D. Z. Chen, C. Wang, Reconstructing unsteady flow data from representative streamlines via diffusion and deep learning based denoising, IEEE Computer Graphics and Applications 41 (6) (2021) 111–121. doi:10.1109/MCG.2021.3089627.

[8] J. Han, C. Wang, TSR-VFD: Generating temporal super-resolution for unsteady vector field data, Computers & Graphics 103 (2022) 168–179. doi:10.1016/J.CAG.2022.02.001.

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

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

[11] P. Gu, J. Han, D. Z. Chen, C. Wang, Scalar2Vec: Translating scalar fields to vector fields via deep learning, in: Proceedings of IEEE Pacific Visualization Symposium, 2022, pp. 31–40. doi:10.1109/PACIFICVIS53943.2022.00012.

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

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

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

[15] Y. Lu, P. Gu, C. Wang, FCNR: Fast compressive neural representation of visualization images, in: Proceedings of IEEE VIS Conference (Short Papers), 2024, pp. 31–35. doi:10.1109/VIS55277.2024.00014.

[16] M. Yang, K. Tang, C. Wang, Meta-INR: Efficient encoding of volumetric data via meta-learning implicit neural representation, in: Proceedings of IEEE Pacific Visualization Conference (Visualization Notes), 2025, pp. 246–251. doi:10.1109/PacificVis64226.2025.00030.

[17] H. Son, J. Noh, S. Jeon, C. Wang, W.-K. Jeong, MC-INR: Efficient encoding of multivariate scientific simulation data using meta-learning and clustered implicit neural representations, arXiv preprint arXiv:2507.02494 (2024). doi:10.48550/arXiv.2507.02494.

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

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

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

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

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

[23] J. Han, 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) (2023) 4951–4963. doi:10.1109/TVCG.2022.3197203.

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

[25] Q. Wu, D. Bauer, M. J. Doyle, K.-L. Ma, Interactive volume visualization via multi-resolution hash encoding based neural representation, IEEE Transactions on Visualization and Computer Graphics 30 (8) (2024) 5404–5418. doi:10.1109/TVCG.2022.3197203.

[26] K. Tang, C. Wang, StyleRF-VolVis: Style transfer of neural radiance fields for expressive volume visualization, IEEE Transactions on Visualization and Computer Graphics 31 (1) (2025) 613–623. doi:10.1109/TVCG.2024.3456342.

[27] A. Chen, Z. Xu, A. Geiger, J. Yu, H. Su, TensoRF: Tensorial radiance fields, in: Proceedings of European Conference on Computer Vision, 2022, pp. 333–350. doi:10.1007/978-3-031-19824-3_20.

[28] S. Weiss, R. Westermann, Differentiable direct volume rendering, IEEE Transactions on Visualization and Computer Graphics 28 (1) (2022) 562–572. doi:10.1109/TVCG.2021.3114769.

[29] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, A. Kanazawa, Plenoxels: Radiance fields without neural networks, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2022, pp. 5501–5510. doi:10.1109/CVPR52688.2022.00542.

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

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

[32] M. M. Loper, M. J. Black, OpenDR: An approximate differentiable renderer, in: Proceedings of European Conference on Computer Vision, 2014, pp. 154–169. doi:10.1007/978-3-319-10584-0_11.

[33] V. Sitzmann, M. Zollhöfer, G. Wetzstein, Scene representation networks: Continuous 3D-structure-aware neural scene representations, in: Proceedings of Advances in Neural Information Processing Systems, 2019, pp. 1119–1130.

[34] S. Liu, T. Li, W. Chen, H. Li, Soft rasterizer: A differentiable renderer for image-based 3D reasoning, in: Proceedings of IEEE International Conference on Computer Vision, 2019, pp. 7707–7716. doi:10.1109/ICCV.2019.00780.

[35] W. Chen, J. Gao, H. Ling, E. J. Smith, J. Lehtinen, A. Jacobson, S. Fidler, Learning to predict 3D objects with an interpolation-based differentiable renderer, in: Proceedings of Advances in Neural Information Processing Systems, 2019, pp. 9605–9616.

[36] M. Niemeyer, L. Mescheder, M. Oechsle, A. Geiger, Differentiable volumetric rendering: Learning implicit 3D representations without 3D supervision, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2020, pp. 3501–3512. doi:10.1109/CVPR42600.2020.00356.

[37] M. Nimier-David, D. Vicini, T. Zeltner, W. Jakob, Mitsuba 2: A retargetable forward and inverse renderer, ACM Transactions on Graphics 38 (6) (2019) 203:1–203:17. doi:10.1145/3355089.3356498.

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

[39] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, P. P. Srinivasan, Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields, in: Proceedings of IEEE International Conference on Computer Vision, 2021, pp. 5855–5864. doi:10.1109/ICCV48922.2021.00580.

[40] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, P. Hedman, Mip-NeRF 360: Unbounded anti-aliased neural radiance fields, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2022, pp. 5470–5479. doi:10.1109/CVPR52688.2022.00539.

[41] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, P. Hedman, Zip-NeRF: Anti-aliased grid-based neural radiance fields, in: Proceedings of IEEE International Conference on Computer Vision, 2023, pp. 19697–19705. doi:10.1109/ICCV51070.2023.01804.

[42] A. Jain, M. Tancik, P. Abbeel, Putting NeRF on a diet: Semantically consistent few-shot view synthesis, in: Proceedings of IEEE International Conference on Computer Vision, 2021, pp. 5885–5894. doi:10.1109/ICCV48922.2021.00583.

[43] J. Yang, M. Pavone, Y. Wang, FreeNeRF: Improving few-shot neural rendering with free frequency regularization, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2023, pp. 8254–8263. doi:10.1109/CVPR52729.2023.00798.

[44] A. Pumarola, E. Corona, G. Pons-Moll, F. Moreno-Noguer, D-NeRF: Neural radiance fields for dynamic scenes, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2021, pp. 10318–10327. doi:10.1109/CVPR46437.2021.01018.

[45] R. Shao, Z. Zheng, H. Tu, B. Liu, H. Zhang, Y. Liu, Tensor4D: Efficient neural 4D decomposition for high-fidelity dynamic reconstruction and rendering, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2023, pp. 16632–16642. doi:10.1109/CVPR52729.2023.01596.

[46] P. P. Srinivasan, B. Deng, X. Zhang, M. Tancik, B. Mildenhall, J. T. Barron, NeRV: Neural reflectance and visibility fields for relighting and view synthesis, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2021, pp. 7495–7504. doi:10.1109/CVPR46437.2021.00741.

[47] X. Zhang, P. P. Srinivasan, B. Deng, P. Debevec, W. T. Freeman, J. T. Barron, NeRFactor: Neural factorization of shape and reflectance under an unknown illumination, ACM Transactions on Graphics 40 (6) (2021) 237:1–237:18. doi:10.1145/3478513.3480496.

[48] M. Boss, R. Braun, V. Jampani, J. T. Barron, C. Liu, H. Lensch, NeRD: Neural reflectance decomposition from image collections, in: Proceedings of IEEE International Conference on Computer Vision, 2021, pp. 12684–12694. doi:10.1109/ICCV48922.2021.01245.

[49] B. Yang, Y. Zhang, Y. Xu, Y. Li, H. Zhou, H. Bao, G. Zhang, Z. Cui, Learning object-compositional neural radiance field for editable scene rendering, in: Proceedings of IEEE International Conference on Computer Vision, 2021, pp. 13779–13788. doi:10.1109/ICCV48922.2021.01352.

[50] C. Bao, Y. Zhang, B. Yang, T. Fan, Z. Yang, H. Bao, G. Zhang, Z. Cui, SINE: Semantic-driven image-based nerf editing with prior-guided editing field, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2023, pp. 20919–20929. doi:10.1109/CVPR52729.2023.02004.

[51] L. Liu, J. Gu, K. Zaw Lin, T.-S. Chua, C. Theobalt, Neural sparse voxel fields, in: Proceedings of Advances in Neural Information Processing Systems, 2020, pp. 15651–15663.

[52] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, A. Kanazawa, PlenOctrees for real-time rendering of neural radiance fields, in: Proceedings of IEEE International Conference on Computer Vision, 2021, pp. 5752–5761. doi:10.1109/ICCV48922.2021.00570.

[53] S. Niedermayr, C. Neuhauser, K. Petkov, K. Engel, R. Westermann, Application of 3D Gaussian splatting for cinematic anatomy on consumer class devices, in: Proceedings of Vision, Modeling, and Visualization, 2024, pp. 1–13. doi:10.2312/vmv.20241195.

[54] K. Tang, S. Yao, C. Wang, iVR-GS: Inverse volume rendering for explorable visualization via editable 3D Gaussian splatting, IEEE Transactions on Visualization and Computer Graphics 31 (6) (2025) 3783–3795. doi:10.1109/TVCG.2025.3567121.

[55] S. Peng, M. Niemeyer, L. M. Mescheder, M. Pollefeys, A. Geiger, Convolutional occupancy networks, in: Proceedings of European Conference on Computer Vision, 2020, pp. 523–540. doi:10.1007/978-3-030-58580-8_31.

[56] T. DeVries, M. Á. Bautista, N. Srivastava, G. W. Taylor, J. M. Susskind, Unconstrained scene generation with locally conditioned radiance fields, in: Proceedings of IEEE International Conference on Computer Vision, 2021, pp. 14284–14293. doi:10.1109/ICCV48922.2021.01404.

[57] E. R. Chan, C. Z. Lin, M. A. Chan, K. Nagano, B. Pan, S. D. Mello, O. Gallo, L. J. Guibas, J. Tremblay, S. Khamis, T. Karras, G. Wetzstein, Efficient geometry-aware 3D generative adversarial networks, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2022, pp. 16102–16112. doi:10.1109/CVPR52688.2022.01565.

[58] A. Tewari, J. Thies, B. Mildenhall, P. Srinivasan, E. Tretschk, W. Yifan, C. Lassner, V. Sitzmann, R. Martin-Brualla, S. Lombardi, T. Simon, C. Theobalt, M. Nießner, J. T. Barron, G. Wetzstein, M. Zollhöfer, V. Golyanik, Advances in neural rendering, Computer Graphics Forum 41 (2) (2022) 703–735. doi:10.1111/cgf.14507.

[59] T. Nguyen-Phuoc, C. Li, S. Balaban, Y.-L. Yang:, RenderNet: A deep convolutional network for differentiable rendering from 3D shapes, in: Proceedings of Advances in Neural Information Processing Systems, 2018, pp. 7902–7912.

[60] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, G. Wetzstein, Implicit neural representations with periodic activation functions, in: Proceedings of Advances in Neural Information Processing Systems, 2020.

[61] G. Li, Y. Liu, G. Shan, S. Cheng, W. Cao, J. Wang, K.-C. Wang, Params-Drag: Interactive parameter space exploration via image-space dragging, IEEE Transactions on Visualization and Computer Graphics 31 (1) (2025) 624–634. doi:10.1109/TVCG.2024.3456338.

[62] X. Pan, A. Tewari, T. Leimkühler, L. Liu, A. Meka, C. Theobalt, Drag your GAN: Interactive point-based manipulation on the generative image

manifold, in: Proceedings of ACM SIGGRAPH Conference, 2023, pp. 78:1–78:11. doi:10.1145/3588432.3591500.
[63] S. Yao, Y. Lu, C. Wang, ViSNeRF: Efficient multidimensional neural radiance field representation for visualization synthesis of dynamic volumetric scenes, in: Proceedings of IEEE Pacific Visualization Conference, 2025, pp. 235–245. doi:10.1109/PacificVis64226.2025.00029.
[64] J. Ye, L. Wang, G. Li, D. Chen, S. Zhe, X. Chu, Z. Xu, Learning compact recurrent neural networks with block-term tensor decomposition, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 9378–9387. doi:10.1109/CVPR.2018.00977.
[65] Z. Kuang, F. Luan, S. Bi, Z. Shu, G. Wetzstein, K. Sunkavalli, PaletteNeRF: Palette-based appearance editing of neural radiance fields, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2023, pp. 20691–20700. doi:10.1109/CVPR52729.2023.01982.
[66] S. Yao, W. Song, C. Wang, A comparative study of neural surface reconstruction for scientific visualization, in: Proceedings of IEEE VIS Conference (Short Papers), 2024, pp. 181–185. doi:10.1109/VIS55277.2024.00045.
[67] Kitware ParaView, `https://github.com/Kitware/ParaView`, accessed: 2024-09-15.
[68] NVDIA IndeX Cloud, `https://github.com/NVIDIA/nvindex-cloud`, accessed: 2024-09-15.
[69] W. E. Lorensen, H. E. Cline, Marching cubes: A high resolution 3d surface construction algorithm, in: Proceedings of ACM SIGGRAPH Conference, 1987, pp. 163–169. doi:10.1145/37401.37422.
[70] R. Marques, C. Bouville, M. Ribardiere, L. P. Santos, K. Bouatouch, Spherical Fibonacci point sets for illumination integrals, in: Computer Graphics Forum, Vol. 32, 2013, pp. 134–143. doi:10.1111/CGF.12190.
[71] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, O. Wang, The unreasonable effectiveness of deep features as a perceptual metric, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 586–595. doi:10.1109/CVPR.2018.00068.
[72] S. Marschner, R. Lobb, An evaluation of reconstruction filters for volume rendering, in: Proceedings of IEEE Visualization Conference, 1994, pp. 100–107. doi:10.1109/VISUAL.1994.346331.
[73] C. Tomasi, R. Manduchi, Bilateral filtering for gray and color images, in: Proceedings of IEEE International Conference on Computer Vision, 1998, pp. 839–846. doi:10.1109/ICCV.1998.710815.
[74] U. Tiede, T. Schiemann, K. H. Hohne, High quality rendering of attributed volume data, in: Proceedings of IEEE Visualization Conference, 1998, pp. 255–262. doi:10.1109/VISUAL.1998.745311.
[75] J. J. Choi, B.-S. Shin, Y. G. Shin, K. Cleary, Efficient volumetric ray casting for isosurface rendering, Computers & Graphics 24 (5) (2000) 661–670. doi:10.1016/S0097-8493(00)00069-8.
[76] Z. Li, T. Müller, A. Evans, R. H. Taylor, M. Unberath, M.-Y. Liu, C.-H. Lin, Neuralangelo: High-fidelity neural surface reconstruction, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2023, pp. 8456–8465. doi:10.1109/CVPR52729.2023.00817.
[77] J. F. Blinn, Models of light reflection for computer synthesized pictures, in: Proceedings of ACM SIGGRAPH Conference, 1977, pp. 192–198. doi:10.1145/563858.563893.
[78] S. Zhukov, A. Iones, G. Kronin, An ambient light illumination model, in: Proceedings of Eurographics Workshop on Rendering Techniques, 1998, pp. 45–55. doi:10.1007/978-3-7091-6453-2_5.
[79] M. Mittring, Finding next gen: Cryengine 2, in: ACM SIGGRAPH Courses, 2007, pp. 97–121. doi:10.1145/1281500.1281671.

Appendix A. Implementation Details

Feature grid upscaling and smoothing. To improve rendering quality, ReVolVE performs upscaling on the feature grid. Since ReVolVE leverages a vector-matrix decomposition, we upscale the feature grid by expanding the size of vectors and matrices through linear interpolation. Using a higher-resolution feature grid reduces the likelihood of sampling at locations with abrupt feature transitions. As demonstrated in Figure 5 (d) and (e) in the paper, this method provides anti-aliasing benefits and mitigates ringing artifacts [72] while being more efficient than supersampling. However, as illustrated in Figure 5 (f) in the paper, with only feature grid upscaling, we still render an abrupt surface with a visible grid pattern. This happens because upscaling retains the noise from the lower-resolution grid. Therefore, we apply bilateral filtering [73] to the feature grid to smooth the surface while preserving edge details. Note that this method is incompatible with certain radiance field representations, such as Instant-NGP and 3DGS. This is a key reason why we opt for vector-matrix over other representations.

Normal computation. Next, we compute the gradient vector at the surface point to determine its normal for subsequent lighting computation. As illustrated in Figure A.1, we follow a numerical approach to estimate the gradient. Given a point $\mathbf{p}$ on the surface, we sample a pair of points on opposite sides of $\mathbf{p}$ along each axis, with a sampling interval of ϵ. The differences between these points are then calculated to approximate the partial derivatives, forming gradient vector components together. For example, the x-component of the gradient vector can be computed as

$$\nabla_x \sigma(\mathbf{p}) = -\frac{1}{2\epsilon}(\sigma(\mathbf{p} + \epsilon_x) - \sigma(\mathbf{p} - \epsilon_x)), \tag{A.1}$$

where ∇_x is the x-component of the gradient vector, $\sigma(\mathbf{p})$ is the density at $\mathbf{p}$, and ϵ_x is a vector parallel to the x-axis with length ϵ. Note that we flip the direction of the gradient vector to ensure the normal points toward the lower-density region. For example, Figure 5 (b) in the paper visualizes the estimated surface normals.

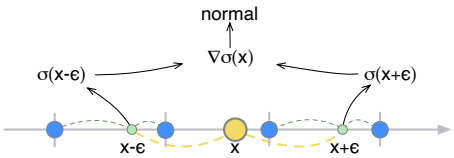


Figure A.1: Numerical normal estimation (x-component). The yellow point represents a point along the camera ray, the green points are sampling locations used to compute the gradient, and the blue points represent features in grid cells.

Since we estimate only at the surface point where the ray intersects the isosurface, this introduces minimal computational overhead to the rendering process. Although *analytical* gradient computation is available through the differentiable volume renderer, which automatically computes gradients, the *numerical* gradient approach offers more flexibility in normal estimation with arbitrary sampling intervals. Additionally, as observed in other surface rendering studies [74, 75, 76], the numerical approach results in smoother surfaces, as the estimated normals incorporate more voxels in the 3D grid when the sampling interval exceeds the voxel size.

Illumination. We implement surface lighting using the Blinn-Phong model [77], which locally approximates the illumination to improve the visual appearance of volume visualization. The Blinn-Phong model can be expressed as

$$\mathbf{I} = k_a + k_d \mathbf{I}_L(\mathbf{n} \cdot \mathbf{l}) + k_s \mathbf{I}_L(\mathbf{n} \cdot \mathbf{h})^\alpha, \text{where } \mathbf{h} = \frac{\mathbf{l} + \mathbf{v}}{|\mathbf{l} + \mathbf{v}|}. \tag{A.2}$$

In Equation A.2, k_a, k_d, and k_s are ambient, diffuse, and specular coefficients, $\mathbf{I}_L$ is the light intensity, $\mathbf{l}$, $\mathbf{n}$, and $\mathbf{v}$ are light, normal, and view directions, and α denotes the shininess term.

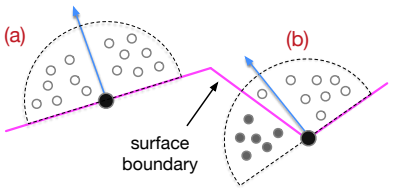


Figure A.2: AO approximation. At a ray-surface intersection point, ReVolVE samples N points within a hemisphere above the surface. AO occurs when certain sample points fall inside the surface, as shown in (b).

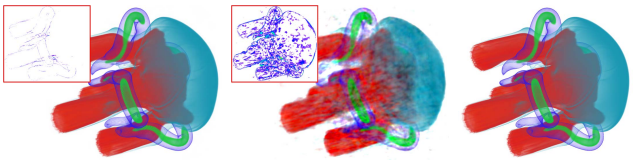
Ambient occlusion (AO). We further include AO to enhance the visualizations. The AO at a point $\mathbf{p}$ on the surface is computed by integrating the visibility over a hemisphere above the surface. The standard method [78] uses ray tracing to test whether a ray originating from the surface intersects with any other surface within the hemisphere. As sketched in Figure A.2, our approach approximates the visibility by sampling points inside the hemisphere and checks if the sample points fall inside the surface, determined by whether the sample's density exceeds the isosurface threshold. The approach for approximating AO is similar to screen-space AO [79], but instead of using depth, we rely on voxel density and compute it as

$$\begin{aligned} \text{AO}(\mathbf{p}, \mathbf{n}) &= \frac{1}{|X|} \sum_{\mathbf{x} \in X} S(\mathbf{x}), \\ \text{where } S(\mathbf{x}) &= \begin{cases} 0, & \text{if } \mathbf{x} \text{ is outside} \\ 1, & \text{otherwise} \end{cases} \end{aligned} \tag{A.3}$$

where $\text{AO}(\mathbf{p}, \mathbf{n})$ represents the AO at point $\mathbf{p}$ on the surface with normal $\mathbf{n}$, X is the set of sample points within the hemisphere, $\mathbf{x}$ is an individual sample point, and $S(\mathbf{x})$ is a binary function indicates whether the sample point is inside or outside the surface. Figure 5 (c) and (d) in the paper show an AO volume and rendering with an AO effect.

Table A.1: Quantitative comparison of with and without the feature decoder using the five jets dataset. Reported here and the rest of the tables in the Appendix are average PSNR (dB), SSIM, and LPIPS values across all 181 synthesized views, and training time (TT, in minutes), average single-frame inference time (IT, in seconds) of a view, and model size (MS, in MB).

decoder	PSNR↑	SSIM↑	LPIPS↓	TT↓	IT↓	MS↓
w/	**39.08**	**0.991**	**0.009**	11.43	**2.55**	**3.37**
w/o	26.40	0.918	0.165	**9.25**	2.88	3.47

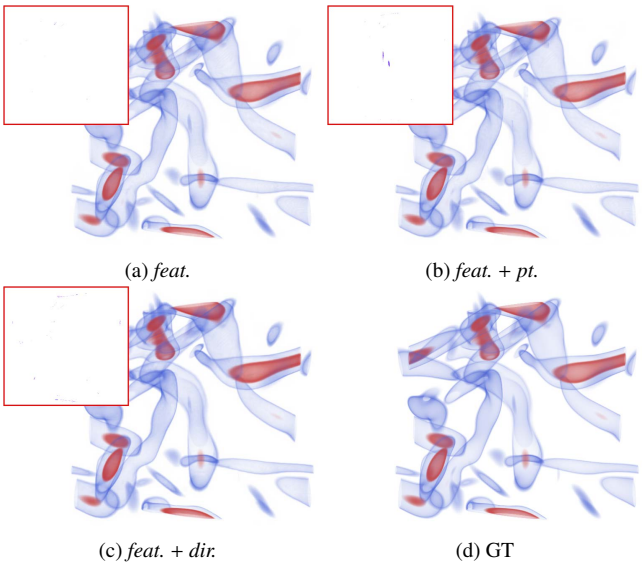


(a) w/ decoder (b) w/o decoder (c) GT

Figure A.3: Qualitative comparison of with and without the feature decoder using the five jets dataset.

Table A.2: Quantitative comparison of different feature decoder inputs using the vortex dataset. *feat.* denotes the grid feature, *pt.* represents the sample point position, and *dir.* means the camera view direction.

decoder input	PSNR↑	SSIM↑	LPIPS↓	TT↓	IT↓	MS↓
feat.	**45.56**	**0.994**	**0.007**	**7.60**	2.47	**1.81**
feat. + pt.	44.22	0.993	0.009	8.10	2.51	**1.81**
feat. + dir.	42.74	**0.994**	0.008	7.68	**2.44**	**1.81**



(a) *feat.* (b) *feat. + pt.* (c) *feat. + dir.* (d) GT

Figure A.4: Qualitative comparison of different feature decoder inputs using the vortex dataset.

Table A.3: Quantitative comparison of different feature grid designs using the head dataset.

type	PSNR↑	SSIM↑	LPIPS↓	TT↓	IT↓	MS↓
unified	**38.38**	**0.979**	**0.032**	**16.45**	**3.71**	**12.68**
split	33.83	0.943	0.077	22.37	4.50	12.75

Appendix B. Ablation Study

In this section, we ablate the ReVolVE framework to analyze the impact of each module. For each experiment, in the tables, we present average PSNR (dB), SSIM, and LPIPS values across all 181 synthesized views, and training time (TT, in minutes), inference time (IT, in minutes), and model size (MS, in MB). The best ones are highlighted in bold.

Feature decoder. To evaluate the effect of the feature decoder, we perform an ablation study that excludes it from the ReVolVE framework. We use the five jets dataset while keeping all training configurations consistent. The concatenated features sampled from vectors and matrices are then passed through a fully connected layer to produce output density and color. The results in Table A.1 and Figure A.3 suggest that the feature decoder is essential for improving ReVolVE's performance, compared to using only the grid features.

Feature decoder input. To assess the impact of sample point position or view direction as an additional input alongside the grid feature for the decoder, we compare the outcomes of three input combinations using the vortex dataset and the same hyperparameters. Note that we use positional encoding in NeRF [38] to add extra expressiveness to position and view direction inputs. According to Table A.2 and Figure A.4, incorporating position as an additional input to the decoder decreases model performance and slightly increases training and inference time. Adding view direction as an additional input to the decoder also leads to a decrease in performance, but it has a minimal impact on training and inference time. Therefore, using the grid feature as the sole input for ReVolVE is sufficient, avoiding interference from additional inputs. This could be one reason ReVolVE performs better than TensoRF and Instant-NGP in terms of rendering quality in our experiments.

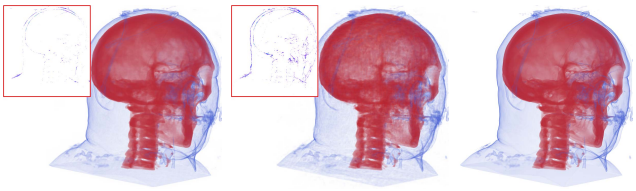
Unified feature grid. TensoRF employs a split design, with one feature grid and decoder handling densities and another set for colors. ReVolVE, on the other hand, uses a single unified feature grid and decoder for both densities and colors. To assess the impact of these designs, we compare the performance of the split vs. unified approaches using the head dataset. The results in Table A.3 and Figure A.5 indicate that using a single unified feature grid and decoder delivers significantly better performance and more efficient training and inference. This could also explain why ReVolVE outperforms TensoRF in reconstructing volumes from DVR images.

Tensor decomposition. ReVolVE applies a vector-matrix decomposition, but alternatively, volumes can be decomposed using either a full vector or a full matrix. To assess the impact of each, we ablate the components of vector-matrix decomposition with the supernova dataset. All other training configurations remain consistent except for the decomposition method. The results in Table A.4 and Figure A.6 indicate that the vector-matrix combination outperforms other decomposition methods. The vector-matrix decomposition noticeably improves the performance of ReVolVE compared to matrix-only decomposition, with only a minimal increase in training and inference time.

Appendix C. Hyperparameter Study

This section evaluates various hyperparameter configurations to determine the optimal setup for ReVolVE. For each experiment, we show, in the table, the average PSNR (dB), SSIM, and LPIPS metrics across 181 synthesized views, alongside the training time (TT, in minutes), inference time (IT, in minutes), and model size (MS, in MB). The best results in the tables are highlighted in bold.

Number of training images. Identifying the minimum number of training images needed for ReVolVE to produce solid results would be valuable. We assess this using the five jets dataset. Figure A.7 (a) and Table A.5 show that ReVolVE can reach satisfactory quality (PSNR over 30 dB) using just five DVR images for training. Fewer than five training images are insufficient for ReVolVE to produce reliable visual enhance-



(a) unified (b) split (c) GT

Figure A.5: Qualitative comparison of different feature grid designs using the head dataset.

Table A.4: Quantitative comparison of different decomposition methods using the supernova dataset. V represents vectors, and M denotes matrices.

scheme	PSNR↑	SSIM↑	LPIPS↓	TT↓	IT↓	MS↓
VM	**36.57**	**0.957**	**0.046**	25.15	4.54	19.73
M	34.74	0.943	0.065	25.65	**4.24**	19.71
V	28.18	0.885	0.175	**23.80**	4.29	**1.11**

ments. As shown in Figure A.8, we used 12 images for training on the five jets dataset, as the generation quality shows little improvement beyond that number.

Number of training iterations. We use the supernova dataset to assess the performance of ReVolVE across various training iterations. As shown in Figure A.7 (b), Table A.6, and Figure A.9, ReVolVE can generate solid rendering results even with just 10,000 iterations. This demonstrates the model's ability to reach high accuracy relatively quickly, offering efficiency in training time and resource use. We selected 30,000 iterations as ReVolVE tends to converge around this point.

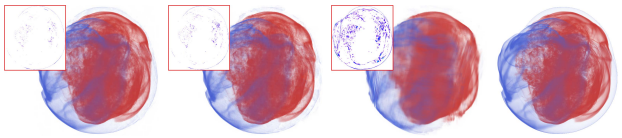
Number of channels in the hidden layer. We use the head dataset to test different numbers of channels for the hidden layer in the feature decoder. The results, as shown in Figure A.7 (c), Table A.7, and Figure A.10, suggest that increasing the number of channels could improve the performance of ReVolVE. We chose 64 channels to obtain a balance between performance and cost. Note that for ReVolVE to produce reasonable results after training, the number of channels must exceed the depth of the feature grid.

Resolution of feature grid. We use the chameleon dataset to investigate the effects of the maximum number of voxels for the feature grid. As shown in Figure A.7 (d), Table A.8, and Figure A.11, increasing the voxel count enhances ReVolVE's performance at the cost of greater training time, inference time, and model size. We settled on 384^3 voxels to ensure optimal performance with reasonable costs.

Depth of feature grid. We use the combustion dataset to compare the performance of ReVolVE using various feature grid depths. The results in Figure A.7 (e), Table A.9, and Figure A.12 show that increasing depth enhances performance at the price of increasing training time and model size. Therefore, we set the depth to 24 to achieve optimal results while maintaining reasonable training time and model size.

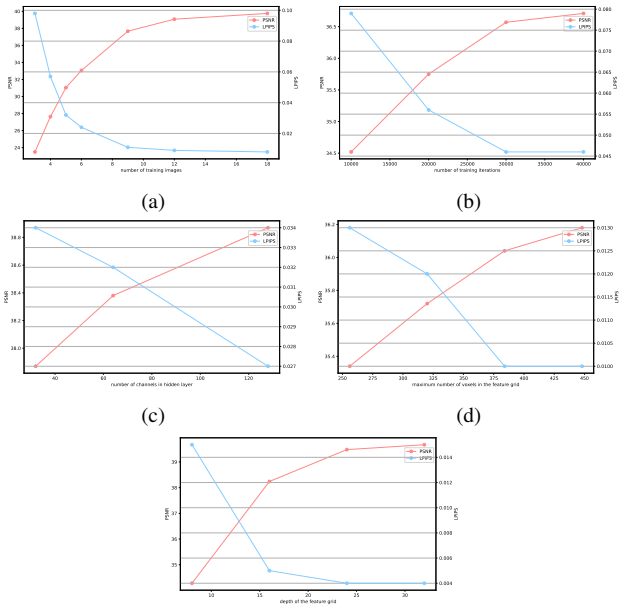
Appendix D. Comparison with 3DGS and CGS

We evaluate ReVolVE against 3DGS [30] and CGS [53] with different Gaussian counts to ensure a thorough comparison. Specifically, we increase the number of Gaussians from 400,000 (0.4 M) to 1,600,000 (1.6 M) for the supernova



(a) VM (b) M (c) V (d) GT

Figure A.6: Qualitative comparison of different decomposition methods using the supernova dataset.



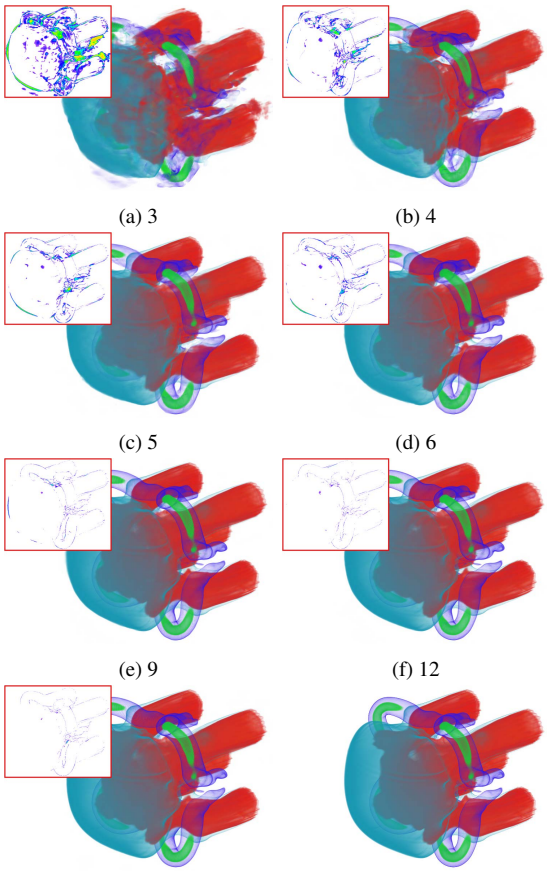
(a) (b) (c) (d) (e)

Figure A.7: Plots of PSNR and LPIPS metrics versus (a) the number of training DVR images using the five jets dataset, (b) the number of training iterations using the supernova dataset, (c) the number of channels in the hidden layer of the feature decoder using the head dataset, (d) the maximum number of voxels for the feature grid using the chameleon dataset, and (e) the depth of the feature grid using the combustion dataset.

dataset, and from 800,000 (0.8 M) to 3,200,000 (3.2 M) for the chameleon dataset. This comparison aims to demonstrate that the selected version of 3DGS used as a baseline in the paper represents a fair benchmark within the context of scientific visualization. All methods are trained using the same set of input views. For fairness, we exclude view selection and compression stages from CGS, focusing solely on core reconstruction capabilities. As shown in Table D.10 and Figure D.13, ReVolVE consistently outperforms both 3DGS and CGS across both datasets. The number of Gaussians used in 3DGS and CGS appears sufficient to represent the scenes, as further increases yield no apparent quality improvement. Since CGS does not outperform 3DGS in either dataset, we use the vanilla 3DGS as a baseline method for comparison in the paper.

Table A.5: Quantitative comparison of different numbers of training DVR images of the five jets dataset.

# images	PSNR↑	SSIM↑	LPIPS↓	TT↓	IT↓	MS↓
3	23.49	0.930	0.098	11.10	2.63	**3.32**
4	27.63	0.955	0.057	10.83	2.59	**3.32**
5	31.04	0.972	0.032	11.40	2.58	3.34
6	33.06	0.979	0.024	11.25	2.58	3.33
9	37.67	0.989	0.011	**10.55**	2.55	3.35
12	39.08	**0.991**	0.009	11.43	2.55	3.37
18	**39.76**	**0.991**	**0.008**	11.05	**2.54**	3.33



(a) 3 (b) 4 (c) 5 (d) 6 (e) 9 (f) 12 (g) 18 (h) GT

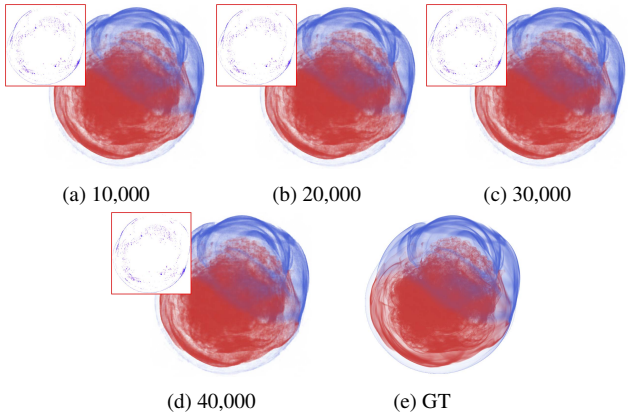
Figure A.8: Qualitative comparison of different numbers of training DVR images of the five jets dataset.

Table A.6: Quantitative comparison of different numbers of training iterations using the supernova dataset.

iterations	PSNR↑	SSIM↑	LPIPS↓	TT↓	IT↓	MS↓
10,000	34.52	0.941	0.079	**7.22**	4.62	19.75
20,000	35.75	0.951	0.056	17.42	4.62	19.75
30,000	36.57	**0.957**	**0.046**	25.15	**4.54**	**19.73**
40,000	**36.71**	**0.957**	**0.046**	35.27	4.61	**19.73**

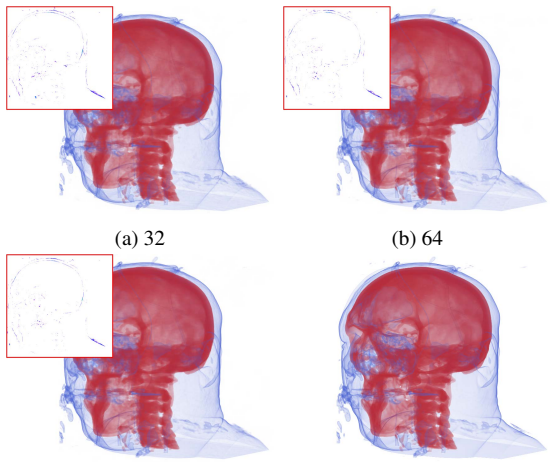
Table A.7: Quantitative comparison of different numbers of channels in the hidden layer of the feature decoder using the head dataset.

# channels	PSNR↑	SSIM↑	LPIPS↓	TT↓	IT↓	MS↓
32	37.87	0.977	0.034	**15.65**	**3.60**	12.75
64	38.38	0.979	0.032	16.45	3.71	**12.68**
128	**38.87**	**0.980**	**0.027**	19.38	3.87	12.76



(a) 10,000 (b) 20,000 (c) 30,000 (d) 40,000 (e) GT

Figure A.9: Qualitative comparison of different numbers of training iterations using the supernova dataset.

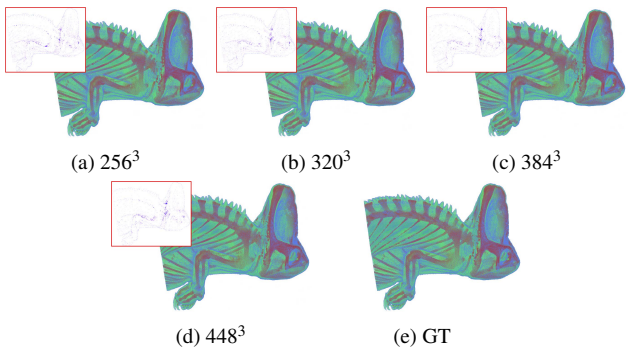


(a) 32 (b) 64 (c) 128 (d) GT

Figure A.10: Qualitative comparison of different numbers of channels in the hidden layer of the feature decoder using the head dataset.

Table A.8: Quantitative comparison of different maximum numbers of voxels for the feature grid using the chameleon dataset.

max # voxels	PSNR↑	SSIM↑	LPIPS↓	TT↓	IT↓	MS↓
256^3	35.34	0.975	0.013	**17.87**	**3.19**	**14.03**
320^3	35.72	0.977	0.012	22.37	3.43	21.01
384^3	36.04	**0.979**	**0.010**	27.75	3.76	30.09
448^3	**36.18**	**0.979**	**0.010**	32.50	4.04	40.72

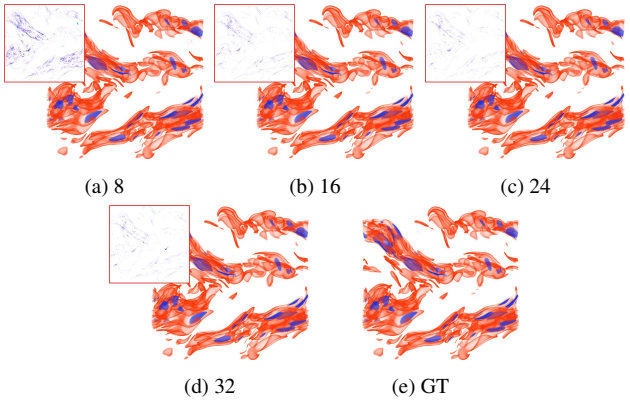


(a) 256^3 (b) 320^3 (c) 384^3 (d) 448^3 (e) GT

Figure A.11: Qualitative comparison of different maximum numbers of voxels for the feature grid using the chameleon dataset.

Table A.9: Quantitative comparison of different feature grid depths using the combustion dataset.

depth	PSNR↑	SSIM↑	LPIPS↓	TT↓	IT↓	MS↓
8	34.28	0.979	0.015	**19.32**	**3.61**	**15.25**
16	38.24	0.991	0.005	21.92	3.71	29.77
24	39.48	**0.993**	**0.004**	28.05	3.69	42.66
32	**39.67**	**0.993**	**0.004**	29.62	3.81	56.26



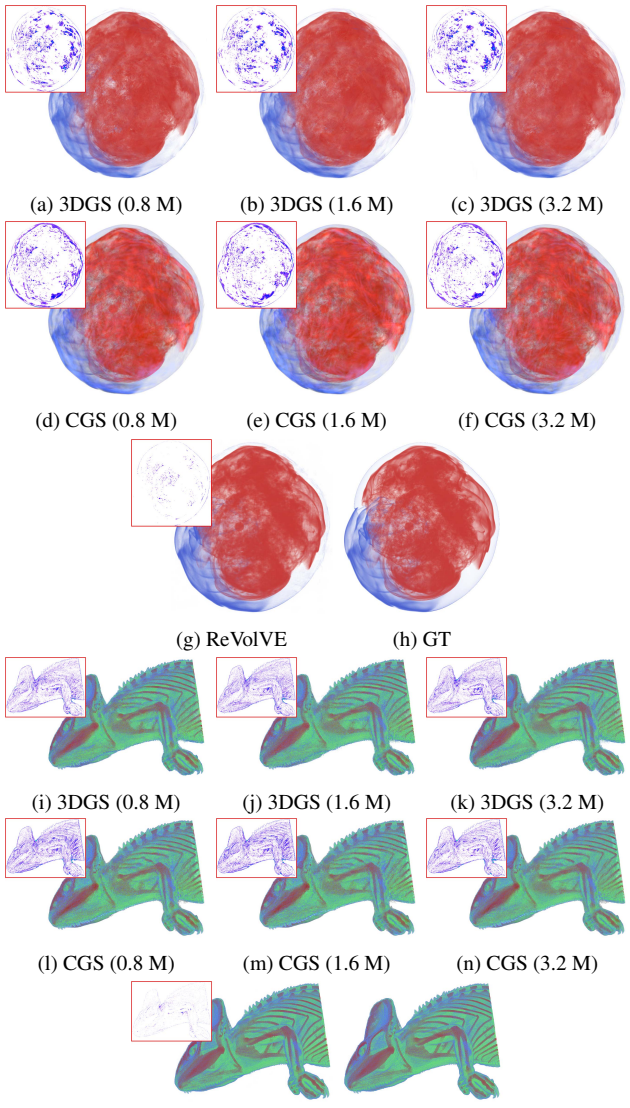
(a) 8 (b) 16 (c) 24

(d) 32 (e) GT

Figure A.12: Qualitative comparison of different feature grid depths using the combustion dataset.

Table D.10: Quantitative comparison of ReVolVE against 3DGS and CGS with different numbers of Gaussians using the supernova and chameleon datasets.

dataset	method	PSNR↑	SSIM↑	LPIPS↓	TT↓	IT↓	MS↓
supernova	3DGS (0.4 M)	29.39	0.907	0.093	**12.83**	0.36	89.62
	3DGS (0.8 M)	29.46	0.908	0.094	17.80	**0.33**	189.79
	3DGS (1.6 M)	29.18	0.907	0.104	26.32	**0.33**	382.54
	CGS (0.4M)	28.76	0.918	0.116	16.32	0.56	94.70
	CGS (0.8 M)	28.56	0.919	0.118	21.45	0.55	196.22
	CGS (1.6 M)	28.89	0.920	0.118	27.20	0.54	378.98
	ReVolVE	**36.57**	**0.957**	**0.046**	25.15	4.54	**19.73**
chameleon	3DGS (0.8 M)	29.90	0.932	0.037	**17.57**	0.30	195.73
	3DGS (1.6 M)	29.64	0.932	0.040	30.57	**0.29**	393.30
	3DGS (3.2 M)	29.56	0.932	0.041	53.87	**0.29**	771.10
	CGS (0.8M)	29.16	0.924	0.046	24.47	0.31	200.59
	CGS (1.6 M)	29.10	0.923	0.048	46.57	0.34	403.96
	CGS (3.2 M)	29.43	0.928	0.048	71.70	0.34	778.05
	ReVolVE	**36.04**	**0.979**	**0.010**	27.75	3.76	**30.09**



(a) 3DGS (0.8 M) (b) 3DGS (1.6 M) (c) 3DGS (3.2 M)

(d) CGS (0.8 M) (e) CGS (1.6 M) (f) CGS (3.2 M)

(g) ReVolVE (h) GT

(i) 3DGS (0.8 M) (j) 3DGS (1.6 M) (k) 3DGS (3.2 M)

(l) CGS (0.8 M) (m) CGS (1.6 M) (n) CGS (3.2 M)

(o) ReVolVE (p) GT

Figure D.13: Qualitative comparison of ReVolVE against 3DGS and CGS with different number of Gaussians using the supernova and chameleon datasets.