ViSNeRF: Efficient Multidimensional Neural Radiance Field Representation for Visualization Synthesis of Dynamic Volumetric Scenes

Siyuan Yao* Yunfei Lu† Chaoli Wang‡

University of Notre Dame

ABSTRACT

Domain scientists often face I/O and storage challenges when keeping raw data from large-scale simulations. Saving visualization images, albeit practical, is limited to preselected viewpoints, transfer functions, and simulation parameters. Recent advances in scientific visualization leverage deep learning techniques for visualization synthesis by offering effective ways to infer unseen visualizations when only image samples are given during training. However, due to the lack of 3D geometry awareness, existing methods typically require many training images and significant learning time to generate novel visualizations faithfully. To address these limitations, we propose ViSNeRF, a novel 3D-aware approach for visualization synthesis using neural radiance fields. Leveraging a multidimensional radiance field representation, ViSNeRF efficiently reconstructs visualizations of dynamic volumetric scenes from a sparse set of labeled image samples with flexible parameter exploration over transfer functions, isovalues, timesteps, or simulation parameters. Through qualitative and quantitative comparative evaluation, we demonstrate ViSNeRF's superior performance over several representative baseline methods, positioning it as the state-of-the-art solution. The code is available at `https://github.com/JCBreath/ViSNeRF`.

1 INTRODUCTION

In scientific research, large-scale simulations are essential for modeling complex phenomena across diverse science and engineering fields. However, huge raw data generated from these simulations, especially those involving time sequences and ensemble runs, presents significant challenges for domain scientists. Specifically, I/O operations and storage capacity limitations can lead to cumbersome and inefficient data retrieval and visualization processes, hindering timely analysis and slowing the overall scientific discovery.

A common practice to mitigate these challenges is preserving visualization images rendered from simulation data. While this approach offers a practical solution by reducing the data footprint, it has inherent limitations. Saving visualization images restricts scientists to preselected viewpoints, transfer functions (TFs), and simulation parameters. Such constraints limit the flexibility required for exploratory data analysis, where researchers often need to investigate multiple perspectives and vary visualization settings to uncover hidden patterns or anomalies.

Over the past few years, deep learning has become viable for addressing diverse generation tasks in scientific visualization [55]. Most studies are dedicated to *data generation* for scalar fields [17, 19, 49, 50, 66] and vector fields [12–16] across various tasks such as super-resolution generation, data translation, reconstruction, and completion. In contrast, only a few have focused on *visualization generation* [3, 18, 20, 22]. Examples such as InSituNet [20] and CoordNet [18] support post hoc analysis on scientific simulations without accessing raw simulation data. However, these existing methods do not develop *3D awareness* of *dynamic* volumetric scenes from 2D visualization images. As a result, they constrain the viewpoints to interpolated positions between fixed camera angles, inhibiting the ability to freely rotate and scale the scene or utilize the full six degrees of freedom for camera movement. This limitation is significant because freely navigating visualizations in 3D is essential for thoroughly exploring complex data, identifying intricate structures, and gaining deeper scientific insights. Moreover, although these methods eliminate the need to handle simulation data, they still require over a hundred views of each scene frame and numerous intermediate timesteps and ensemble runs to achieve smooth interpolation. Additionally, the training can take from hours to days, further complicating their usability. These dilemmas not only amplify the costs of preparing training data but also constrain the practicality of these techniques for real-world scientific applications.

To address the limitations of existing methods, we present ViSNeRF, a deep learning framework for **Vi**sualization **S**ynthesis using **Ne**ural **R**adiance **F**ields. ViSNeRF introduces an innovative factorization approach that allows a single NeRF model to represent dynamic scenes with multiple controllable parameters. By incorporating this multidimensional NeRF representation, ViSNeRF enables us to explore dynamic volumetric scenes interactively. This includes adjusting parameters such as TFs for *direct volume rendering* (DVR), isovalues for *isosurface rendering* (IR), timesteps, and simulation parameters, as well as examining the scene from any desired viewpoint with confidence. Moreover, combining an explicit feature grid and a pair of *multilayer perceptron* (MLP) decoders for color and density, the hybrid NeRF representation allows ViSNeRF to accelerate training while improving generation quality. The contributions of ViSNeRF are as follows:

- ViSNeRF efficiently synthesizes high-resolution and high-quality visualizations from novel viewpoints via a hybrid radiance field representation.
- Leveraging a generalized factorization strategy, ViSNeRF accomplishes dynamic scene generation with plausible interpolation over the parameter space, including TFs, isovalues, timesteps, or simulation parameters.
- Our comprehensive study validates the effectiveness of ViSNeRF and demonstrates its superior performance compared to state-of-the-art methods.

2 RELATED WORK

Deep learning for visualization generation. Deep learning has been applied in scientific visualization to solve data generation, visualization generation, prediction, objection detection and segmentation, and feature learning and extraction tasks [55]. We restrict our attention to works that have succeeded in visualization generation.

Berger et al. [3] designed GAN-VR, a *generative adversarial network* (GAN) framework to synthesize DVR and allow users to explore the space of viewpoints and TFs. Hong et al. [22] introduced DNN-VolVis, which applies the rendering effects of the user-defined goal image to the original DVR image without knowing the explicit TFs. He et al. [20] proposed InSituNet that generates visualization

*e-mail: syao2@nd.edu

†e-mail: ylu25@nd.edu

‡e-mail: chaoli.wang@nd.edu

at simulation time and enables post hoc exploration of ensemble simulations. Shi et al. [47] leveraged a view-dependent surrogate model named VDL-Surrogate to infer volume data and generate visualizations with user-defined visual mappings for parameter-space exploration. Han and Wang [18] built CoordNet, which leverages *implicit neural representation* (INR) to solve various scientific visualization tasks, including view synthesis.

To achieve super-resolution of IR images while maintaining consistent geometric properties, Weiss et al. [59] proposed a fully *convolutional neural network* (CNN) trained with depth and normal maps. Weiss et al. [61] presented a neural rendering approach that employs low-resolution rendering images to predict the density distribution of the volume data and adaptively samples it to produce high-resolution images. Weiss and Navab [58] developed DeepDVR to model the DVR process by learning the implicit semantics for feature extraction, classification, and visual mapping. To save rendering time, Bauer et al. [2] designed the FoVolNet, which reconstructs full-frame renderings from foveated renderings.

Scene representation networks (SRNs) represent volumetric data for neural volume rendering without requiring direct access to the original volume data. Weiss et al. [60] introduced a technique using a volumetric grid of latent features that allows for effective rendering by encoding essential volumetric information. Further advancing this direction, Wurster et al. [64] developed an adaptive multi-grid SRN that enhances rendering efficiency by representing large-scale data across various resolutions. Wu et al. [63] proposed a compressive neural representation that employs multi-resolution hash encoding, achieving rapid training speeds and optimized memory usage for large-scale volumetric data. Recently, Tang and Wang [51] introduced StyleRF-VolVis, a method leveraging NeRF to perform geometry-aware stylization with only 2D images.

ViSNeRF is designed for visualization synthesis of dynamic volumetric scenes and is benchmarked exclusively against methods that can natively accomplish this task. GAN-VR [3] requires 200,000 rendering images for training dynamic scenes. DNN-VolVis [22] and InSituNet [20] only work with low-resolution (256×256) images. VDL-Surrogate [47] needs access to the raw data during training and incurs a long training time (50 hours). While CoordNet [18] supports high-resolution (1024×1024) image synthesis, it requires up to five days to train with 200 images. In contrast, our ViSNeRF requires only 42 rendering images for training a static scene and several hundred to a few thousand images for training a dynamic scene. It works with high-resolution (1024×1024) images, achieving fast training (up to 35 minutes for a static scene and up to 2.5 hours for a dynamic scene) and superior quality for synthesized images.

3D-aware image synthesis. Recent works of novel-view image synthesis have moved on to incorporate camera information to enhance the 3D consistency of generated views. Early approaches, such as PrGAN [10], VON [73], PlatonicGAN [21], HoloGAN [37], and BlockGAN [38], use voxels to represent the scene and generate images based on the voxel shape. However, due to the limited voxel resolution, these methods fail to reconstruct fine details of the original scenes. Liao et al. [31] suggested using 3D primitives to represent the scene for 3D-aware view synthesis. While this scheme allows for 3D control, it may be inadequate when reconstructing complex scenes, as primitives provide only limited information. Consequently, the resulting image synthesis may be suboptimal or of low quality. Unlike the above approaches, neural field representations are more popular and effective for 3D-aware image synthesis. NeRF [35] is the seminal work demonstrating the great potential of neural scene representations. Numerous variants of NeRF have successfully produced remarkable synthesis results [6, 11, 26, 34, 45, 71], improved the capability of NeRF in many scenarios, and extended its applications from image synthesis to 3D reconstruction [40, 56, 68], 3D content generation [5, 25, 39, 54], and dynamic scene representations [4, 41, 42, 48].

Efficient NeRFs. Although NeRF can generate realistic results with a compact MLP, slow convergence and long training and inference are common issues among most pure implicit methods. Later efforts of efficiency improvement [9, 32, 70] focus on space-time tradeoff, which sacrifice memory space to accelerate the rendering process of radiance field methods. By factorizing the complex voxel-based feature grid of radiance fields, emerging decomposed hybrid NeRF architectures express exceptionally high efficiency in both computation and memory usage. *Generative scene network* (GSN) [8] is the first plane-based work that uses 2D representations of the radiance fields. Efficient geometry-aware 3D GAN (EG3D) [5] enables style-mixing and latent-space interpolation by leveraging StyleGAN2 [29] to generate features of the triplane representation. Instant-NGP [36] integrates a multiresolution hash table of trainable feature vectors with a compact network, illustrating a hybrid representation approach known for its efficiency in training and inference. *3D Gaussian splatting* (3DGS) [30], which represents 3D scenes with 3D Gaussians, offers an efficient explicit approach that eliminates the need for neural networks, significantly speeding up training and inference. iVR-GS [52] designs editable 3DGS to achieve inverse volume rendering for explorable visualization of color, opacity, and lighting parameters. *Tensorial radiance field* (TensoRF) [7] introduces a tensor-based architecture to obtain photorealistic quality with high computational and memory efficiency.

Factorized NeRFs. Recent advancements like K-Planes [43] and HexPlane [4], inspired by TensoRF, extend to factorized 4D NeRF representations to reconstruct dynamic scenes. We also acknowledge works, such as Tensor4D [46], which factorize signed distance fields to represent 3D geometry. Yet, those works do not perform factorization on radiance fields, so they do not fit our scope. ViSNeRF adopts a generalized factorization strategy for multidimensional NeRF representations to address the demands of visualization synthesis of dynamic volumetric scenes.

3 ViSNeRF

ViSNeRF leverages a NeRF model to learn a volumetric representation of a scene from a set of DVR or IR images, along with camera pose information, which can be provided directly by users or estimated using tools like COLMAP [44]. This NeRF representation allows us to produce 3D-consistent visualizations from any viewpoint. ViSNeRF uses a hybrid NeRF model combining *explicit feature grids* and *implicit feature decoders* to improve speed and quality. To handle the dynamic scenes efficiently, ViSNeRF incorporates a *generalized factorization strategy* on the NeRF to achieve a multidimensional representation with minimal increase in model size and GPU memory cost. This strategy allows us to efficiently represent dynamic scenes with an arbitrary number of adjustable parameters, such as TFs, isovalues, timesteps, and simulation parameters.

As shown in Figure 1, to synthesize a visualization of a dynamic volumetric scene, we first cast rays from screen pixels toward the scene, originating from the camera locations. For every point sampled along a ray, we provide ViSNeRF with the sampling point location (x,y,z) and the parameters $(p_0, p_1, ...)$ defining a scene frame. ViSNeRF then retrieves features from spatial matrices and vectors as well as parameter vectors, decoding them into color and density values. It finally composes a visualization of the scene frame from the camera view by collecting sampled points on the rays.

3.1 Volume Rendering with Radiance Fields

Following the classical volume rendering [27] as NeRF [35] suggested, we formulate the rendering function as

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

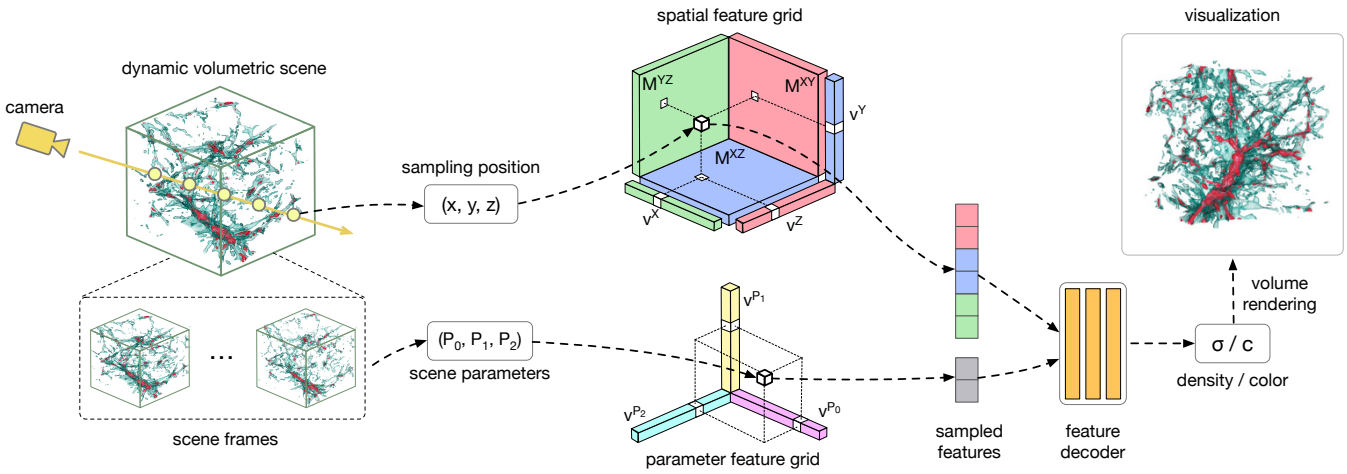


Figure 1: Overview of ViSNeRF using the Nyx dataset as an example of a dynamic volumetric scene. Features are sampled from spatial and parameter feature grids based on the camera ray's sampling position and scene parameters. These features are processed by the decoder to generate density and color values, which are then used in volume rendering to visualize the scene frame from the chosen camera view.

In Equation 1, $\mathcal{C}(\mathbf{r})$ denotes the color sampled through the ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ from the camera, where $\mathbf{o}$ is the origin, $\mathbf{d}$ is the viewing direction, and t_n and t_f are near and far bounds. The accumulated transmittance function $\mathcal{T}(t)$ determines the probability that no particle is present along the ray between t_n and t. The volume density function $\sigma(\mathbf{x})$ is the probability that the ray is stopped at location $\mathbf{x}$ by a particle. The view-dependent color term $c(\mathbf{x}, \mathbf{d})$ belongs to the particle that terminates the ray at location $\mathbf{x}$ with $\mathbf{d}$.

We use stratified sampling along the ray to estimate the integral. Following the quadrature rule [33], the resulting color of the ray, $\mathcal{C}(\mathbf{r})$, is estimated using n samples along the ray by

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

In Equation 2, the ray marching step size δ_i is defined by the distance between the locations of contiguous samples, i.e., $\delta_i = t_{i+1} - t_i$.

3.2 Factorization of Static Radiance Fields

In a dynamic scene, individual frames can be treated as static scenes represented by 3D radiance fields. Like TensoRF [7], we use the block-term tensor decomposition [69] to factorize the 3D volume $\mathbf{V} \in \mathbb{R}^{X \times Y \times Z}$ of the static radiance fields as the sum of vector-matrix outer products

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

where $\mathbf{M}_r^{XY}$, $\mathbf{M}_r^{XZ}$, and $\mathbf{M}_r^{YZ}$ are the spatial matrices (refer to Figure 1) representing the XY, XZ, and YZ planes. $\mathbf{v}_r^X$, $\mathbf{v}_r^Y$, and $\mathbf{v}_r^Z$ are the spatial vectors along the X, Y, and Z axes. $\mathbf{M}_r^{XY} \circ \mathbf{v}_r^Z$ denotes the outer product of $\mathbf{M}_r^{XY}$ and $\mathbf{v}_r^Z$. R_1, R_2, and R_3 denote 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 $\mathbf{M}_r^{YZ} \circ \mathbf{v}_r^X$, and the numbers can be updated according to the complexity and dimensions of the effective volume. By employing factorization, we can reduce the memory complexity for representing the radiance field from $O(N^3)$ to $O(N^2)$ when $N \gg R_1 + R_2 + R_3$, where N is the spatial resolution. For simplicity, we assume the same complexity across all three dimensions. Thus, we use the same number of components for each dimension in 3D static radiance fields (i.e., $R = R_1 = R_2 = R_3$). The factorization is then expressed as

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

3.3 Factorization of Parameter Space

Allowing NeRF to represent dynamic volumetric scenes is necessary to synthesize visualizations for time-varying or ensemble simulation data with different TFs or isovalues. Training multiple NeRF models for visualizing individual volumes is impractical, as isolating these models makes effective interpolation between them impossible.

Existing works such as HexPlane [4] and K-Plane [43] have made strides in factorizing dynamic 4D scenes and achieved commendable results in dynamic scene reconstruction, they still fall short in our (3+K)-D scenarios, where K represents the number of changeable parameters defining different scenes. A key limitation of these methods is their lack of efficient scalability to multidimensional spaces, primarily due to excessive reliance on planes. For instance, both HexPlane and K-Plane necessitate the addition of three extra planes when transitioning from 3D to 4D. Moreover, while HexPlane is limited to 4D, K-Plane suggests adding another four planes to move from 4D to 5D. This requirement increases the complexity exponentially, making these methods impractical for direct application in our more complex, higher-dimensional scenarios.

To address the scalability challenges in our (3+K)-D scenarios, shown in Figure 1, we propose to divide the scene into two separate tensors: a 3D tensor $\mathcal{T}^3$ and a K-D tensor $\mathcal{T}^K$, each subjected to a distinct factorization strategy. While $\mathcal{T}^3$ undergoes vector-matrix decomposition for enhanced expressiveness, $\mathcal{T}^K$ is factorized using the CANDECOMP/PARAFAC (CP) decomposition for higher compactness and scalability. This technique allows us to factorize the K-D tensor into a series of K vectors, effectively managing the complexity inherent in multidimensional parameter spaces. As a result, the factorization of our K-D parameter space can be represented as

$$\mathcal{T}^K = \sum_{r=1}^{R} \mathbf{v}_r^1 \circ \mathbf{v}_r^2 \circ \cdots \circ \mathbf{v}_r^K, \tag{5}$$

where R represents the number of one-rank components in the form of $\mathbf{v}_r^1 \circ \mathbf{v}_r^2 \circ \cdots \circ \mathbf{v}_r^K$. This tensor factorization significantly reduces the complexity of the parameter space in arbitrary dimensions from $O(M^K)$ to $O(M)$, where M is the parameter-space grid resolution. Consequently, we can represent the entire (3+K)-D scene in a more manageable and efficient manner. Written in equation

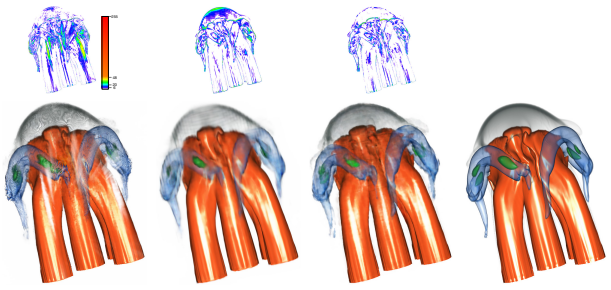
$$\begin{aligned}\mathcal{T}^{3+K} &= \mathcal{T}^3 \circ \mathcal{T}^K \\ &= \sum_{r=1}^{R_s} \mathbf{M}_r^{XY} \circ \mathbf{v}_r^Z + \mathbf{M}_r^{XZ} \circ \mathbf{v}_r^Y + \mathbf{M}_r^{YZ} \circ \mathbf{v}_r^X \\ &\circ \sum_{r=1}^{R_p} \mathbf{v}_r^1 \circ \mathbf{v}_r^2 \circ \cdots \circ \mathbf{v}_r^K,\end{aligned} \tag{6}$$

where $\mathcal{T}^{3+K}$ represents the (3+K)-D tensor encapsulating the entire dynamic scene, R_s is the number of low-rank components in $\mathcal{T}^3$, and R_p is the number of one-rank components in $\mathcal{T}^K$. This approach significantly reduces the overall complexity from $O(N^3M^K)$ to $O(N^2M)$. Provided that $R_s \ll N$ and $R_p \ll M^K$, ViSNeRF achieves ultra-compactness in its representation while retaining the capability to handle dynamic scenes with an arbitrary number of parameters.

In Table 1 and Figure 2, we demonstrate that ViSNeRF, despite having the smallest model size among the three methods, surpasses both K-Planes and HexPlane in visualization generation quality for 4D scenarios, using the five jets dataset with an additional time dimension. In Figure 2, to help with the comparison, we produce difference images with respect to the ground-truth (GT) result in the CIELUV color space. Noticeable pixel differences are mapped according to the colormap legend as shown in (a). For this comparison, K-Plane is configured according to its recommended training settings, and HexPlane is set up with its suggested configurations but matched to the same grid resolution as ViSNeRF. All three methods undergo training for 90,000 iterations. We exclude K-Plane and HexPlane from further discussion as these methods are not designed to handle (3+K)-D scenarios.

Table 1: Factorization methods for dynamic scenes using five jets (timestep) DVR images with 1024×1024 resolution: average PSNR (dB), SSIM, LPIPS, model size (MS, in MB), training time (TT, in hours), and inference time (IT, in minutes) across all 181 synthesized views. The best ones are highlighted in bold.

method	PSNR↑	SSIM↑	LPIPS↓	MS↓	TT↓	IT↓
K-Planes	26.67	0.939	0.057	417.40	3.02	22.95
HexPlane	26.54	0.941	0.063	22.23	**1.06**	**18.93**
ViSNeRF	**27.51**	**0.945**	**0.050**	**15.43**	1.43	21.20



(a) K-Planes (b) HexPlane (c) ViSNeRF (d) GT

Figure 2: Inferred five jets (timestep) DVR images generated by K-Planes, HexPlane, and ViSNeRF using 462 views to train the dynamic scene with a full 360-degree view.

In our implementation, the combination of features sampled from spatial matrices and vectors is achieved through element-wise multiplications between each vector-matrix pair. This element-wise multiplication is also applied for combining features sampled from the parameter vectors. We concatenate the outcomes of these element-wise multiplications to construct the final feature. This involves integrating the products derived from the three pairs of spatial matrix and vector as well as the product of the parameter vectors.

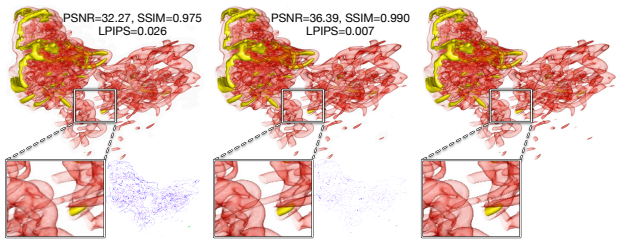
3.4 Feature Decoder

Unlike fully explicit methods, such as PlenOctrees [70] and Plenoxels [9], using *spherical harmonics* (SH) as the decoder, our hybrid ViSNeRF model leverages two small MLPs to decode the density and color features separately. TensoRF [7] and K-Planes [43] compare a fully explicit model and a hybrid model. The results suggest that, although using SH reduces rendering time, the quality of synthesized images gets worse due to the limited expressiveness of SH and the difficulty of SH coefficient optimization. Instead of directly employing SH as the decoder, we utilize SH to encode the view directions, which are then fed as input into the color MLP decoder, denoted as g_c. We observe that encoding view directions with SH slightly enhances network performance compared to *positional encoding* (PE) [53].

In TensoRF [7], without an implicit MLP decoder for the density features, the explicit density features are not sufficiently expressive to discern the edges of overlapping translucent regions in DVR images, especially when the training images are scarce (e.g., less than 50 images for the full 360-degree view). Thus, in addition to the MLP g_c for color features f_c, we use an extra small MLP g_σ to decode the density features f_σ at a trivial cost of efficiency, which also helps improve the accuracy of overlapping translucent regions for DVR images. Both g_c and g_σ have only one hidden layer. They map the feature of a coordinate $\mathbf{x}$ to density and color values as

$$\begin{aligned}\sigma(\mathbf{x}) &= g_\sigma(f_\sigma(\mathbf{x}), E_{\mathrm{P}}(\mathbf{x})), \\ c(\mathbf{x}) &= g_c(f_c(\mathbf{x}), E_{\mathrm{SH}}(\mathbf{d})),\end{aligned} \tag{7}$$

where E_{P} and E_{SH} denote PE and SH encoding. As shown in Figure 3, the DVR images generated by ViSNeRF exhibit improved clarity in the translucent regions compared with TensoRF. Both methods apply the same L1 and TV regularization (refer to Section 3.5) to ensure fairness.



(a) TensoRF (b) ViSNeRF (c) GT

Figure 3: Inferred DVR images of the Tangaroa dataset using 42 views to train the full 360-degree view. Zoom-in and difference images are provided for better comparison.

3.5 Loss Functions

To optimize ViSNeRF, a straightforward approach utilizes a loss function that measures the difference between the generated visualization and the GT, known as reconstruction loss. However, as shown in Figure 4, without any regularization, excessive noises and film grains are apparent in the visualized spatial matrices of ViSNeRF and the generated images. Therefore, in addition to the reconstruction loss, we apply an L1 norm loss and a total variation (TV) loss as regularization terms in the loss function

$$\mathcal{L} = \mathcal{L}_{\mathrm{REC}} + \lambda_1 \mathcal{L}_{\mathrm{L1}} + \lambda_2 \mathcal{L}_{\mathrm{TV}}, \tag{8}$$

where λ_1 and λ_2 are weights of the regularization terms.

Reconstruction loss. During the training process, in each batch, a set of rays $\mathcal{R}$ is randomly selected from the pool of all pixels from the images. As characterized in Equation 2, we query n samples along each ray $\mathbf{r}$ and predict the color of the corresponding pixel $\hat{C}(\mathbf{r})$. The reconstruction loss is computed using the MSE between

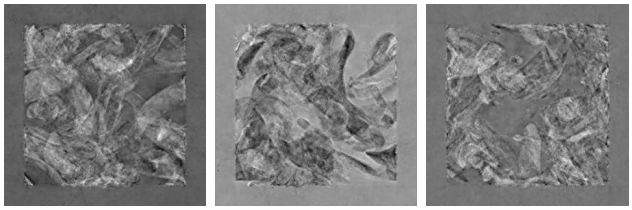
the predicted colors $\hat{\mathcal{C}}(\mathbf{r})$ and GT colors $\mathcal{C}(\mathbf{r})$ of the set of pixels in each batch, and is defined as

$$\mathcal{L}_{\text{REC}} = \frac{1}{|\mathcal{R}|} \sum_{\mathbf{r}\in\mathcal{R}} \left|\left|\mathcal{C}(\mathbf{r}) - \hat{\mathcal{C}}(\mathbf{r})\right|\right|_2^2. \tag{9}$$

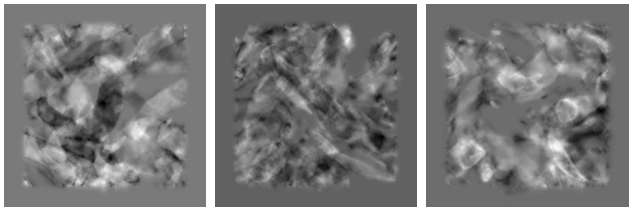
L1 norm loss. To reduce overfitting by extracting more relevant features, we employ L1 norm loss as a regularization term

$$\mathcal{L}_{\text{L1}} = \frac{1}{|W|} \sum_{w\in W} \begin{cases} ||1-w||_1 & \text{if } W \in \mathbf{v}_p \\ ||w||_1 & \text{otherwise} \end{cases} \tag{10}$$

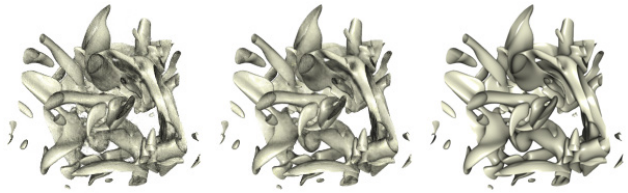
where W are the weights in the vectors and matrices in the radiance field representation, and $\mathbf{v}_p$ is the set of vectors representing input parameter features. For the vectors containing the parameter features, the weights are initialized and regularized to 1 for no alteration in the spatial content. Note that L1 regularization is only applied to the vectors and matrices, not the implicit MLP decoder.



(a) no regularization



(b) L1+TV regularization



(c) no regularization (d) L1+TV GT

Figure 4: Spatial matrices and IR images of novel views generated by ViSNeRF using 42 views of the vortex dataset for training. (a) and (b) shows the three spatial matrices of density feature.

Total variation loss. Although L1 norm loss reduces the artifacts by preventing overfitting, if we have only a few views of the scene available for training, the vectors and matrices would still be noisy since some regions are not seen in these views. TV loss is a regularization term that aims to denoise the vectors and matrices by penalizing high variation and encouraging smoothness. That is,

$$\begin{aligned} \mathcal{L}_{\text{TV}} &= \mathcal{L}_{\text{TV}_1} + \mathcal{L}_{\text{TV}_2}, \\ \mathcal{L}_{\text{TV}_1} &= \frac{1}{|\mathcal{V}||\mathcal{Q}|l} \sum_{\mathbf{v},q,i} \left(\left|\left|\mathbf{v}_q^i - \mathbf{v}_q^{i-1}\right|\right|_2^2\right), \\ \mathcal{L}_{\text{TV}_2} &= \frac{1}{|\mathcal{M}||\mathcal{Q}|hw} \sum_{\mathbf{M},q,i,j} \left(\left|\left|\mathbf{M}_q^{i,j} - \mathbf{M}_q^{i-1,j}\right|\right|_2^2 \right. \\ &\left. + \left|\left|\mathbf{M}_q^{i,j} - \mathbf{M}_q^{i,j-1}\right|\right|_2^2\right), \end{aligned} \tag{11}$$

where $\mathbf{v} \in \mathcal{V}$, $\mathbf{M} \in \mathcal{M}$, $q \in \mathcal{Q}$, $\mathcal{V}$ is the set of vectors, $\mathcal{M}$ is the set of matrices, $\mathcal{Q}$ is the channels of the features, l is the length of the vector, and h and w are the height and width of the plane.

4 Results and Discussion

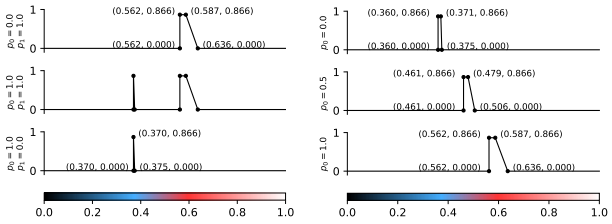
For dynamic volumetric scenes, we present the results of ViSNeRF and compare them with four baseline methods (i.e., InSituNet, CoordNet, StyleGAN2, and EG3D) in parameter-space exploration tasks. For static volumetric scenes, we also compare ViSNeRF with eight baseline and representative methods (i.e., InSituNet, CoordNet, StyleGAN2, EG3D, NeRF, 3DGS, Instant-NGP, and TensoRF). The experimental results are furnished in the appendix. The appendix also includes optimization schemes, baseline training details, hyperparameter study, and additional results and discussion.

4.1 Datasets

As shown in Table 2, we evaluate ViSNeRF in five dynamic volumetric scenes from four simulation datasets with different controllable parameters. The image resolutions are determined based on the content present in the data. The vortex and Nyx are relatively simple and can use low-resolution renderings. The Tangaroa and five jets need high-resolution renderings to capture the details.

To evenly distribute the viewpoints around the volume data, the camera positions are determined by the vertices of an icosphere, which approximates a sphere composed of equilateral triangles. The number of vertices starts with 12 at the subdivision level 0 (initial icosahedron), followed by 42, 92, 162, and 252 for successive subdivision levels. Experiments on static scenes show that 42 vertices at subdivision level 1 are sufficient to create a training set that delivers satisfactory generation quality for ViSNeRF. Hence, we also use 42 views per scene frame of dynamic volumetric scenes for training ViSNeRF and the baseline models. For each viewpoint, we record the camera poses for ViSNeRF and other 3D-aware methods and convert them to the corresponding azimuth ([-180,180]) and elevation ([-90,90]) for 2D-based approaches.

The parameter settings that define each scene frame are evenly sampled within the chosen parameter ranges. The exact number of sampled parameter settings is provided in Table 2. For inference on the full 360-degree view, we synthesize 181 views along a path on a spherical surface from azimuth -180 and elevation -90, through azimuth 0 and elevation 0, to azimuth 180 and elevation 90. During the inference sequence, as the camera moves, the scene dynamically changes by varying parameters within their defined ranges.



(a) TF-1 (b) TF-2

Figure 5: Illustration of two ways of interpolation over TFs of the vortex dataset. The tuple of each control point indicates its scalar value (x-axis) and opacity (y-axis).

4.2 Network Training

We use PyTorch to implement ViSNeRF and train it using a single NVIDIA Tesla V100 GPU with 32 GB of video memory. We set the number of low-rank components in the factorization of the 3D tensor $\mathcal{T}^3$ as $R_s = 64$, where it is split to $R_\sigma = 16$ for density and $R_c = 48$ for color to represent the radiance field. Hence, the depths of density and appearance features are 16 and 48, respectively. For the factorization of the K-D parameter space, $\mathcal{T}^K$, we set $R_p = 4$. The length of each parameter vector corresponds to the number of distinct values available for each parameter in the training set. The resolution of the spatial feature grid is initialized as $N \times N \times N$, where $N = 128$. For more complex datasets like Tangaroa, the grid

Table 2: Parameter-space exploration of dynamic scenes: numbers of training images sampled and their resolutions.

dataset (scenario)	volume resolution	# images per scene	# parameters (K)	# parameter samples (M)	total # images	image resolution
five jets (timestep)	128×128×128	42	1	11	462	1024×1024
Tangaroa (isovalue)	300×180×120	42	1	13	546	1024×1024
vortex (TF-1)	128×128×128	42	2	36	1512	256×256
vortex (TF-2)	128×128×128	42	1	11	462	256×256
Nyx-DVR/IR (simulation parameters)	256×256×256	42	3	45	1890	256×256

Table 3: Parameter-space exploration of dynamic scenes: average PSNR (dB), SSIM, LPIPS, MS (in MB), TT (in hours), and IT (in minutes) across all 181 synthesized views.

dataset	method	PSNR↑	SSIM↑	LPIPS↓	MS↓	TT↓	IT↓
five jets timestep (1024×1024)	InSituNet	16.37	0.850	0.176	351.74	39.22	**0.70**
	CoordNet	17.71	0.864	0.169	**5.71**	112.23	13.38
	StyleGAN2	16.41	0.854	0.180	158.68	53.25	1.07
	EG3D	22.74	0.908	0.067	152.14	75.75	1.03
	ViSNeRF	**27.51**	**0.945**	**0.050**	15.43	**1.43**	21.20
Tangaroa isovalue (1024×1024)	InSituNet	17.08	0.841	0.230	351.73	51.97	**0.42**
	CoordNet	19.83	0.876	0.178	**5.72**	120.90	11.84
	StyleGAN2	18.53	0.860	0.178	158.68	60.64	1.19
	EG3D	22.92	0.915	0.081	152.14	89.52	1.17
	ViSNeRF	**29.36**	**0.967**	**0.042**	69.91	**2.49**	31.42
vortex TF-1 (256×256)	InSituNet	17.03	0.726	0.211	199.66	17.52	0.18
	CoordNet	18.06	0.728	0.222	**5.72**	60.57	1.35
	StyleGAN2	18.38	0.823	0.201	103.62	9.07	**0.10**
	EG3D	26.20	0.929	0.054	150.64	82.78	0.22
	ViSNeRF	**37.26**	**0.994**	**0.005**	12.79	**1.68**	1.55
vortex TF-2 (256×256)	InSituNet	16.75	0.779	0.199	199.66	6.13	**0.08**
	CoordNet	17.10	0.793	0.212	**5.72**	35.71	0.64
	StyleGAN2	17.08	0.790	0.197	103.62	4.15	0.10
	EG3D	27.53	0.969	0.024	150.63	37.97	0.35
	ViSNeRF	**36.58**	**0.995**	**0.006**	12.79	**1.42**	1.45
Nyx-DVR simulation parameters (256×256)	InSituNet	14.06	0.495	0.294	199.67	21.41	**0.12**
	CoordNet	15.09	0.514	0.370	**5.71**	87.80	0.85
	StyleGAN2	14.48	0.508	0.302	103.62	11.79	0.15
	EG3D	18.76	0.759	0.157	150.64	108.34	0.37
	ViSNeRF	**30.02**	**0.978**	**0.018**	12.79	**1.66**	1.62
Nyx-IR simulation parameters (256×256)	InSituNet	16.52	0.550	0.234	199.67	21.89	**0.12**
	CoordNet	17.41	0.468	0.299	**5.71**	78.90	0.85
	StyleGAN2	16.88	0.562	0.250	103.62	11.85	0.15
	EG3D	20.38	0.759	0.149	150.64	101.65	0.35
	ViSNeRF	**28.73**	**0.950**	**0.042**	12.79	**1.66**	1.60

Table 4: Simulation parameter values for the Nyx dataset shown in Figure 7.

subfigure	*OmM*	*OmB*	*h*
(a)	0.128448	0.0215	0.55
(b)	0.155	0.022052	0.55
(c)	0.155	0.0235	0.591379
(d)	0.150227	0.023227	0.809091
(e)	0.146552	0.0215	0.55
(f)	0.155	0.023017	0.55
(g)	0.155	0.0235	0.808621
(h)	0.142273	0.022773	0.740909

resolution would increase at iterations 2,000, 4,000, 6,000, and 8,000, and the final grid has 300^3 voxels. Note that the grid does not necessarily stay cubic, meaning the length, width, and height can differ depending on the shape of the bounding box.

The number of cells in the parameter feature grid M^K corresponds to the number of distinct values available for each parameter in the training set. Refer to Table 2, for five jets (timestep) and vortex (TF-2), $K = 1$ and $M^1 = 11$. For Tangaroa (isovalue), $K = 1$ and $M^1 = 13$. For vortex (TF-1), $K = 2$ and $M^2 = 36$ (6×6). For Nyx-DVR/IR (simulation parameters), $K = 3$ and $M^3 = 45$ ($3 \times 3 \times 5$). The model size and training time should grow as the feature depth (R_σ, R_c, and R_p) or the feature grid size (N^3 and M^K) increases.

For all static scenes, we train ViSNeRF for 30,000 iterations, while the number of iterations is tripled for dynamic scenes in all parameter-space exploration experiments. At iteration 2,000, we create the mask volume for storing empty voxel information and shrink the bounding box according to the actual scene content. At iteration 4,000, we update the mask volume and enable voxel skipping.

The MLP decoders for density and appearance have one hidden layer, and each fully connected layer has 128 channels. We set the batch size as 4096, considering the training efficiency and video memory consumption. The number of sample points per ray starts at 192 and increases to 512 if the grid resolution increases to 300^3 voxels. The matrices and vectors and the two MLPs are optimized by an Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.99$. We set the initial learning rate of the matrices and vectors as 0.02 and that of the MLPs as 0.001. The weight of L1 loss (λ_1) is set as 10^{-4} initially and changed to 10^{-5} after 2,000 iterations. The weights of TV loss (λ_2) for density and appearance features are set as 1.0.

4.3 Baselines

For dynamic volumetric scenes, we compare ViSNeRF with methods that natively support the synthesis of visualizations, including InSituNet, CoordNet, StyleGAN2, and EG3D:

- InSituNet [20] is a GAN-based surrogate model supporting parameter-space exploration. We only use azimuth and elevation as the input and add extra upscaling residual blocks in the regressor to support output resolutions larger than 256×256.
- CoordNet [18] uses an INR architecture for visualization generation. We use the original implementation as it can handle any image resolution.
- StyleGAN2 [29] uses a style-based architecture with a discriminator to provide adversarial supervision. We use azimuth and elevation instead of a random latent vector to define the targets explicitly.
- EG3D [5] can be seen as a style-based NeRF where a StyleGAN2 generator produces the triplane representation.

We use StyleGAN2 instead of StyleGAN3 [28] due to two reasons: (1) In the context of visualization generation, where the emphasis is faithful reconstruction, the new features of StyleGAN3, such as translation and rotation equivariance, hardly improve the quality but significantly increase the training difficulty. (2) As we also incorporate EG3D with the StyleGAN2 backbone in our comparison, we aim to closely evaluate and compare the two approaches and illustrate the transition from a 2D approach to a 3D method.

4.4 Results

The quantitative results are displayed in Table 3. To evaluate the quality of generated visualizations, we employ three metrics: *peak signal-to-noise ratio* (PSNR), *structural similarity index* (SSIM) [57], and *learned perceptual image patch similarity* (LPIPS) [72]. Overall, ViSNeRF significantly outperforms the four baseline methods across all three metrics, EG3D comes a distant second, and the rest of the three methods are the worst.

InSituNet, CoordNet, and StyleGAN2 perform poorly because they depend heavily on large training sets. With a small number of training images, as in our experiments, these methods struggle to produce desirable results. Moreover, they require far more scene frames to capture the dynamic scenes adequately. For example, under identical Nyx simulation settings, He et al. [20] used 100 images per scene frame and 400 frames, resulting in 40,000 training images for InSituNet. In contrast, ViSNeRF produces accurate results with 1,890 images, reducing the training data by 95%. This highlights

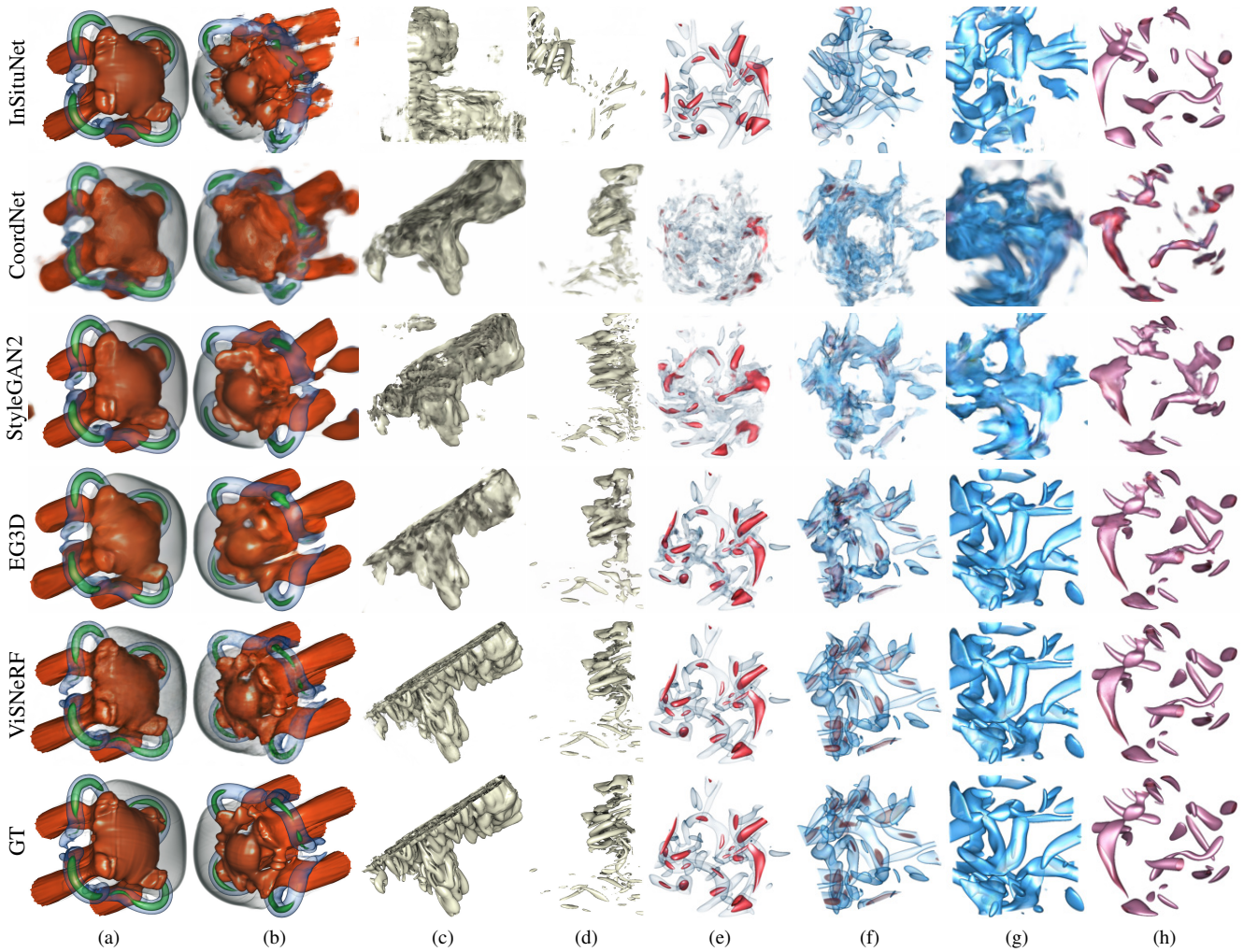


Figure 6: Inferred images under different views for parameter-space exploration of dynamic scenes (timesteps, isovalues, and TFs). (a) and (b) DVR images with interpolation over timesteps using the five jets dataset. (c) and (d) IR images with interpolation over isovalues using the Tangaroa dataset. (e) to (h) DVR images with interpolation over TFs using the vortex dataset (TF-1: (e) and (f); TF-2: (g) and (h)).

ViSNeRF's robustness and efficiency, offering a significant advantage in dynamic scene reconstruction, especially as the demand for training data increases with growing dynamic ranges.

EG3D, despite being 3D-aware, produces less accurate results due to its reliance on convolutional layers to upscale low-resolution synthesized images. While this approach helps address GPU memory consumption issues, it fails to leverage the advantages of 3D-aware reconstruction fully and, like 2D-based methods, depends on a large number of training images to achieve satisfactory results.

For model size, CoordNet wins as the fully implicit representation, ViSNeRF comes second as the hybrid neural representation, and the rest of the three methods incur much larger model sizes. ViSNeRF is the fastest for training time, followed by StyleGAN2 and InSituNet, and CoordNet and EG3D are the slowest. InSituNet and StyleGAN2 are the fastest for inference time, followed by EG3D and CoordNet, and ViSNeRF is the slowest. Nevertheless, for 256×256 images, the frame rate is still at least 1.86 FPS for ViSNeRF. For 1024×1024 images, the frame rate drops substantially.

For qualitative results, the synthesized visualization images are presented in Figure 6 and Figure 7. Figure 8 and Figure 9 show the corresponding difference images. In the following, we examine the qualitative results on a case-by-case basis.

Interpolation over timesteps. We use 1,000 timesteps of the five jets dataset for temporal interpolation. We pick every 100th timestep for training, resulting in 11 samples (including timesteps 1 and 1,000). To test the performance of ViSNeRF, we infer 181 images evenly from timestep 1 to timestep 1,000 in a full 360-degree view. The inferred images are used as the test set for evaluation. Figure 6 (a) and (b) show that CoordNet, StyleGAN2, and EG3D generate blurry results. InSituNet produces results with better clarity, but the reconstructed viewpoints deviate from GT, leading to the worst quantitative performance (refer to Table 3). The results of ViSNeRF are the best.

Interpolation over isovalues. We select an isovalue range [0.01, 0.3] for the Tangaroa dataset, producing meaningful IR occupying the resulting images well. Within the isovalue range, we evenly pick 13 samples (including the two ends) for training and 181 samples in a full 360-degree view for inference. Figure 6 (c) and (d) show that the reconstructed viewpoints from InSituNet are completely wrong, and ViSNeRF produces results with the best detail and clarity compared with CoordNet, StyleGAN2, and EG3D.

Interpolation over TFs. In this experiment, we provide two ways to interpolate TFs, denoted as TF-1 and TF-2, using the vortex dataset. For TF-1, we interpolate the opacity for each visible value range while keeping the ranges intact. As shown in Figure 5 (a), we control the opacities of the red and blue value ranges. We sample 36

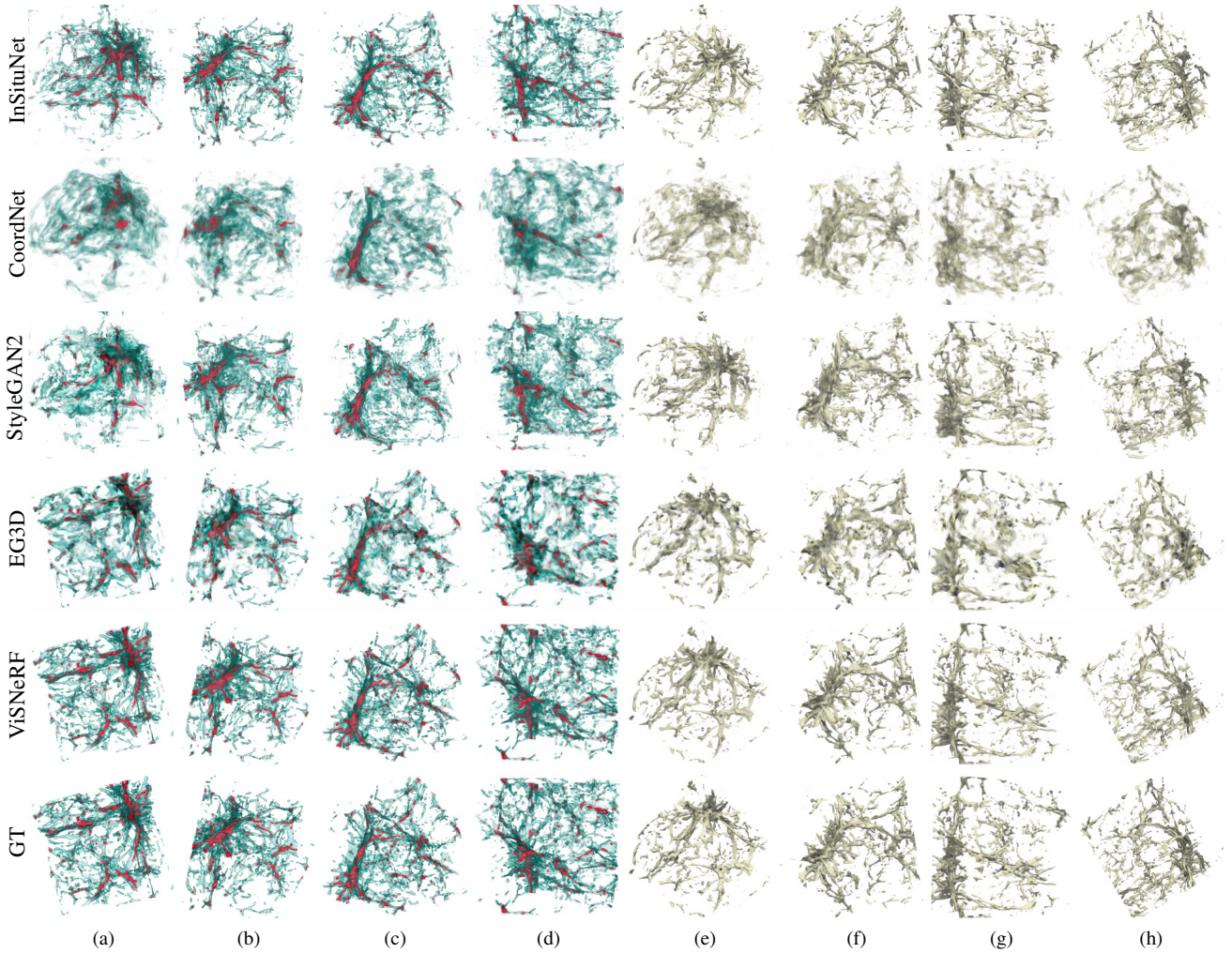


Figure 7: Inferred images under different views for parameter-space exploration of dynamic scenes (simulation parameters). Inferred DVR (a to d) and IR (e to h) images with interpolation over simulation parameters using the Nyx dataset. (a), (b), (c), (e), (f), and (g) are single-parameter variation results. (d) and (h) are multiple-parameter variation results. Table 4 gives each case's simulation parameter values.

steps, including the cases where both red and blue are opaque, one is opaque and the other is transparent, and both are transparent. For TF-2, we interpolate from a visible value range (source) to another (target) using 11 steps. This is achieved by translating the source while updating the bound and opacity to match those of the target. As shown in Figure 5 (b), the blue range is shifted from low values to match the red range at high values. The interpolation will show intermediate ranges with interpolated colors (such as purple). In either case (TF-1 or TF-2), we infer 181 images evenly from the corresponding interpolation scheme. For TF-1, Figure 6 (e) and (f) show that ViSNeRF is the best, followed by EG3D. CoordNet and StyleGAN2 produce rather blurry results, and InSituNet yields significant viewpoint deviations. Similar conclusions can be drawn for TF-2 from Figure 6 (g) and (h).

Interpolation over simulation parameters. For this study, we use an additional dataset, Nyx, derived from a cosmological simulation software developed by Almgren et al. [1]. Our investigation focuses on three critical parameters selected based on expert recommendations. These parameters and their respective ranges, inspired by the methodology used in InSituNet, include the total matter density ($OmM \in [0.12, 0.155]$), the total density of baryons ($OmB \in [0.0215, 0.235]$), and the Hubble constant ($h \in [0.55, 0.85]$). For the training phase, we selectively use 45 parameter configurations derived from a combination of three values of *OmM* (0.12, 0.1375, 0.155), three values of *OmB* (0.0215, 0.0225, 0.0235), and five values of h (0.55, 0.625, 0.70, 0.775, 0.85). The number of intervals chosen for each parameter is determined by its relative impact on the simulation outcome.

During inference, we analyze 181 rendered images, categorized as follows: (1) single-parameter variation: 90 images for isolated changes in each parameter (*OmM*, *OmB*, or h), along with 30 different views per parameter. This kind of variation results in 90 simulation parameter configurations. (2) multiple-parameter variation: 91 images with concurrent changes in all three parameters (*OmM*, *OmB*, and h) back and forth, each with a different view, testing complex interpolation conditions. This kind of variation results in 46 simulation parameter configurations. As five simulation parameter configurations are the same across these two kinds of variation, we end up with $90+46-5=131$ simulation parameter configurations for inference.

Figure 7 shows the results obtained using different variations on the Nyx dataset, encompassing both DVR and IR. The comparison highlights that ViSNeRF produces the highest quality and most accurate results. In contrast, InSituNet fails to generate correct views, whereas CoordNet, StyleGAN2, and EG3D tend to produce blurry outputs. The superior performance of ViSNeRF in simulation

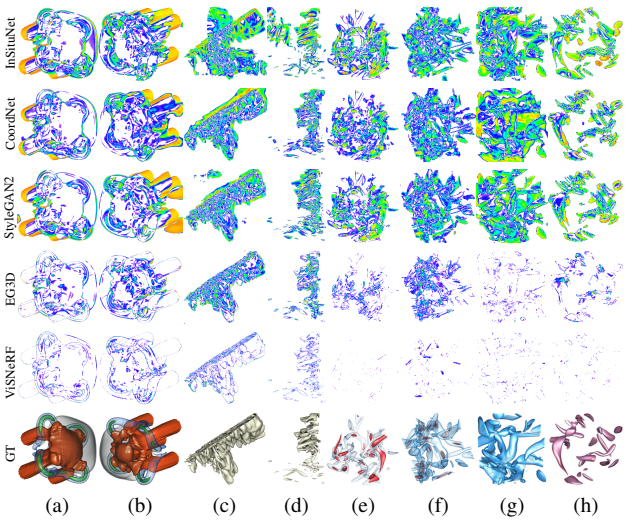


Figure 8: Difference images for Figure 6 of inferred images under different views. (a) and (b) DVR images with interpolation over timesteps using the five jets dataset. (c) and (d) IR images with interpolation over isovalues using the Tangaroa dataset. (e) to (h) DVR images with interpolation over TFs using the vortex dataset (TF-1: (e) and (f); TF-2: (g) and (h)).

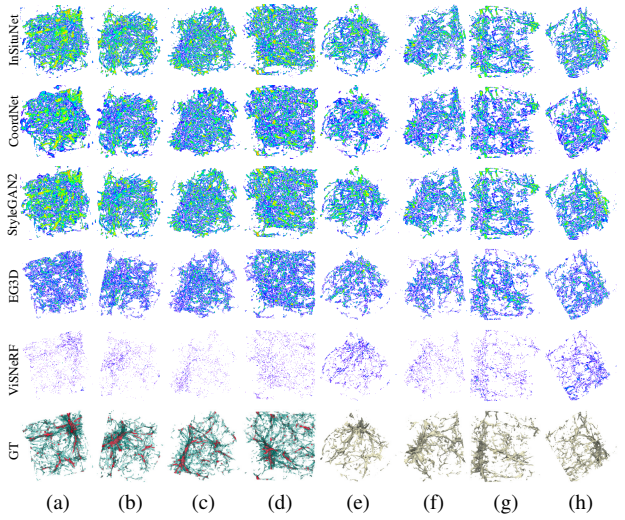


Figure 9: Difference images for Figure 7 of inferred DVR (a to d) and IR (e to h) images with interpolation over simulation parameters using the Nyx dataset. (a), (b), (c), (e), (f), and (g) are single-parameter variation results. (d) and (h) are multiple-parameter variation results.

parameter interpolation is further confirmed by quantitative results given in Table 3.

4.5 Limitations and Future Work

Our ViSNeRF framework has the following limitations, which we will improve in future work. First, we use an icosphere to evenly select sampled viewpoints and choose 42 samples for novel view synthesis. This is not a tight lower bound, as the subdivision level determines the number of samples, which increases significantly with each subdivision level. An alternative solution, such as Poisson disk sampling, provides a good balance between flexibility and control over the density of camera viewpoints on a sphere's surface, even though the random sampling process may not guarantee complete uniformity.

Second, ViSNeRF incorporates both DVR and IR from volume visualization. For IR, there are better ways to synthesize rendering images for isosurfaces than the underlying neural volumetric representation. A more accurate representation should consider implicit surfaces such as signed or unsigned distance functions [67].

Third, as indicated in Table 3, ViSNeRF renders images much slower than InSituNet, which poses a challenge for real-time parameter-space exploration. A potential solution is to adopt 3DGS in the NeRF representation to improve rendering speed. Unfortunately, state-of-the-art Gaussian splatting methods for dynamic scenes [23, 24, 62, 65] are based on deforming a fixed set of 3D Gaussians. This works well for real-world scenarios where motion, like human movement, is the main source of dynamics. However, it creates major problems in cases where different parts of a scene could move, merge, split, appear, or disappear simultaneously, which is common in scientific visualization. When parts appear or disappear, these methods move Gaussians from nearby components with similar colors instead of adding or removing them locally. This leads to visible artifacts and reduces the reconstruction quality of interpolated scene frames. Adding more Gaussians can help by making them smaller and less noticeable when moving, but such a practice requires much more GPU memory. Thus, in the context of scientific visualization, there is a clear need for more efficient and effective methods to handle Gaussians in dynamic scenes to achieve real-time parameter-space exploration.

Finally, fully implementing ViSNeRF within the CUDA framework could speed up rendering. However, it may present a barrier for researchers aiming to extend ViSNeRF to other scientific visualization tasks, such as segmentation or language embedding.

5 Conclusions

We have presented ViSNeRF, a novel solution for dynamic visualization synthesis using NeRF. Compared with representative NeRF methods, ViSNeRF achieves better quality across PSNR, SSIM, and LPIPS using the same number of training views. Furthermore, ViSNeRF excels in static scene reconstruction and showcases its scalability and efficiency in handling dynamic scenes, thanks to its novel factorization strategy. For parameter-space exploration tasks, ViSNeRF also outperforms other deep learning-based methods, including InSituNet, CoordNet, StyleGAN2, and EG3D, in terms of visualization generation quality. Its superior performance suggests that ViSNeRF is a well-designed framework for the visualization synthesis of dynamic volumetric scenes.

Acknowledgments

This research was supported in part by the U.S. National Science Foundation through grants IIS-1955395, IIS-2101696, OAC-2104158, and IIS-2401144, and the U.S. Department of Energy through grant DESC0023145. The authors would like to thank the anonymous reviewers for their insightful comments.

Appendix

1 Optimization Schemes

Coarse-to-fine feature grid. To reduce the training cost at the initial stage and regularize the matrices and vectors, like TensoRF and HexPlane, we apply a coarse-to-fine scheme to the 3D feature grid. The grid is initialized with a coarse resolution. After the initial stage, the matrices and vectors are bi-linearly and linearly upscaled to higher resolution multiple times in subsequent iterations. The final fine resolution depends on the complexity of the visual content in the scene (i.e., input DVR or IR images).

Two-phase progressive sampling and importance sampling. The training is divided into two phases to optimize efficiency: *warm-up* and *finetuning*. During the warm-up phase, the sample points per ray begin at a small number, such as 64, and increase steadily until

Table 1: Novel view synthesis of static scenes: numbers of training images sampled and their resolutions.

dataset	volume resolution	# DVR images	# IR images	image resolution
vortex	128×128×128	42	42	256×256
five jets	128×128×128	42	42	256×256
Tangaroa	300×180×120	42	42	1024×1024
supernova	432×432×432	42	42	1024×1024

they reach the predetermined target number at the phase's conclusion. For the finetuning phase, the number of samples per ray matches the final count of the warm-up phase, plus an additional small number of sample points, which is 64 in our case. These points are sampled using an importance sampling technique akin to NeRF. This two-phase method not only accelerates convergence at the outset but also minimizes early-stage overfitting artifacts. Additionally, it aids the network in learning more intricate details by employing denser sampling in higher complexity areas.

Ray skipping. If a ray does not pass through the scene with any actual content, it is invalid, and we can skip it completely for efficiency. To achieve this, we define a bounding box, which is initialized to the full extent and shrunk to only capture the scene's actual content when the initial training stage completes. Such a tight bounding box contains fewer empty voxels, making the ray-skipping operation more effective.

Voxel skipping. Although a compact bounding box eliminates invalid rays, we could still waste time sampling at empty voxels, forming substantial gaps along a valid ray. After the initial training stage, we create a mask volume (i.e., a binary volume) recording whether each voxel is empty so that we can skip empty voxels to improve subsequent training efficiency. When ViSNeRF is trained for a dynamic scene, we update the mask volume based on the empty voxels determined from multiple static scenes.

2 Novel View Synthesis of Static Scenes

2.1 Datasets

We evaluate ViSNeRF using four simulation datasets, each containing both a DVR scene and an IR scene. As listed in Table 1, in addition to the vortex, five jets, and Tangaroa datasets, we include the more complex supernova dataset to test ViSNeRF's performance on more challenging scenarios. Image resolutions are adjusted according to the content of each dataset. For every scene, we use 42 views for training and 181 views for inference, while the parameter settings remain constant throughout.

2.2 Baselines

In addition to the methods InSituNet, CoordNet, StyleGAN2, and EG3D we use for dynamic scenes, for static scenes, we also include four representative NeRF methods (i.e., NeRF, 3DGS, Instant-NGP, and TensoRF):

- NeRF [35] is a 3D-aware novel view synthesis method that uses neural radiance fields implemented with an MLP.
- 3DGS [30] uses Gaussian ellipsoids to represent radiance fields, enabling efficient optimization and real-time rendering.
- Instant-NGP [36] utilizes multiresolution hash table encoding to achieve rapid NeRF training and fast rendering while maintaining a compact model size even for large and complex scenes.
- TensoRF [7] employs tensor factorization in the radiance field to speed up NeRF training and rendering while improving the quality of reconstructed scenes.

2.3 Quantitative Analysis

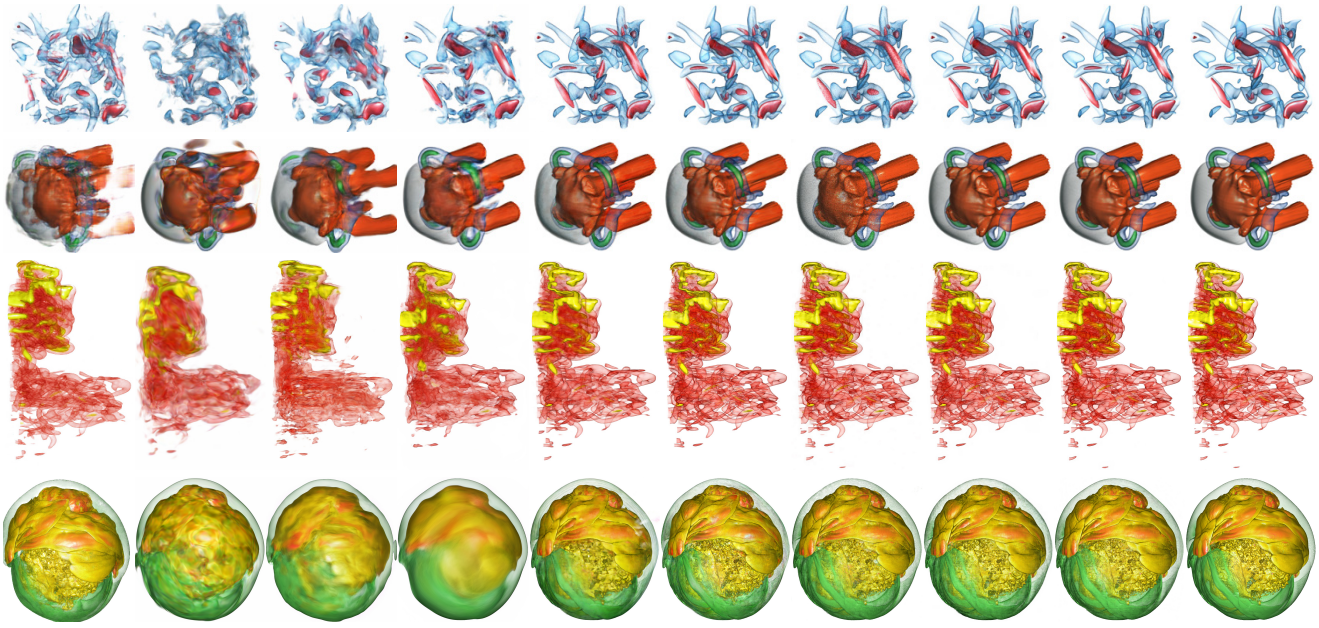
As shown in Table 2, ViSNeRF outperforms all baselines in terms of PSNR, SSIM, and LPIPS for novel view synthesis in static scenes, using the same training datasets. This performance advantage is consistent across resolutions, from lower resolution (256×256) to higher resolution (1024×1024), and applies to both DVR and IR scenes. These results demonstrate ViSNeRF's ability to deliver consistently higher-quality image generation than the baselines in

Table 2: Novel view synthesis of static scenes: average PSNR (dB), SSIM, and LPIPS values across all 181 synthesized views.

		DVR			IR		
dataset	method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
vortex (256×256)	InSituNet	17.15	0.714	0.189	16.09	0.658	0.222
	CoordNet	17.42	0.722	0.210	16.26	0.656	0.259
	StyleGAN2	17.35	0.727	0.199	16.42	0.669	0.232
	EG3D	21.47	0.857	0.109	17.00	0.717	0.221
	NeRF	28.78	0.963	0.023	24.51	0.926	0.063
	3DGS	33.30	0.990	0.009	28.60	0.966	0.028
	Instant-NGP	32.00	0.986	0.014	26.03	0.933	0.092
	TensoRF	35.43	0.993	0.005	29.30	0.968	0.027
	ViSNeRF	**37.32**	**0.996**	**0.003**	**29.46**	**0.970**	**0.025**
five jets (256×256)	InSituNet	17.19	0.794	0.152	19.93	0.822	0.140
	CoordNet	18.30	0.820	0.119	20.70	0.849	0.119
	StyleGAN2	17.26	0.806	0.163	20.50	0.836	0.137
	EG3D	24.13	0.911	0.044	26.79	0.936	0.046
	NeRF	28.48	0.957	0.023	26.32	0.952	0.051
	3DGS	31.57	0.979	0.014	35.96	0.992	0.008
	Instant-NGP	31.02	0.974	0.025	32.68	0.977	0.036
	TensoRF	34.82	0.990	0.007	38.63	**0.995**	**0.005**
	ViSNeRF	**35.80**	**0.992**	**0.005**	**38.66**	**0.995**	**0.005**
Tangaroa (1024×1024)	InSituNet	17.54	0.826	0.197	17.65	0.845	0.184
	CoordNet	18.79	0.835	0.212	18.34	0.851	0.192
	StyleGAN2	17.87	0.823	0.211	17.75	0.838	0.197
	EG3D	22.82	0.873	0.089	20.61	0.884	0.102
	NeRF	29.64	0.954	0.045	23.95	0.928	0.089
	3DGS	32.36	0.979	0.026	30.47	0.980	0.026
	Instant-NGP	35.19	0.986	0.013	30.77	0.976	0.030
	TensoRF	33.24	0.980	0.022	31.68	0.982	0.019
	ViSNeRF	**37.25**	**0.991**	**0.007**	**32.23**	**0.984**	**0.016**
supernova (1024×1024)	InSituNet	15.99	0.626	0.352	16.74	0.726	0.269
	CoordNet	18.05	0.674	0.381	17.76	0.744	0.277
	StyleGAN2	17.38	0.669	0.352	17.16	0.736	0.249
	EG3D	20.00	0.697	0.346	20.22	0.774	0.226
	NeRF	23.82	0.771	0.221	25.15	0.873	0.129
	3DGS	24.21	0.815	0.129	27.49	0.931	0.051
	Instant-NGP	25.99	0.830	0.148	27.58	0.914	0.082
	TensoRF	25.86	0.828	0.181	29.32	0.938	0.071
	ViSNeRF	**27.01**	**0.859**	**0.120**	**29.71**	**0.946**	**0.049**

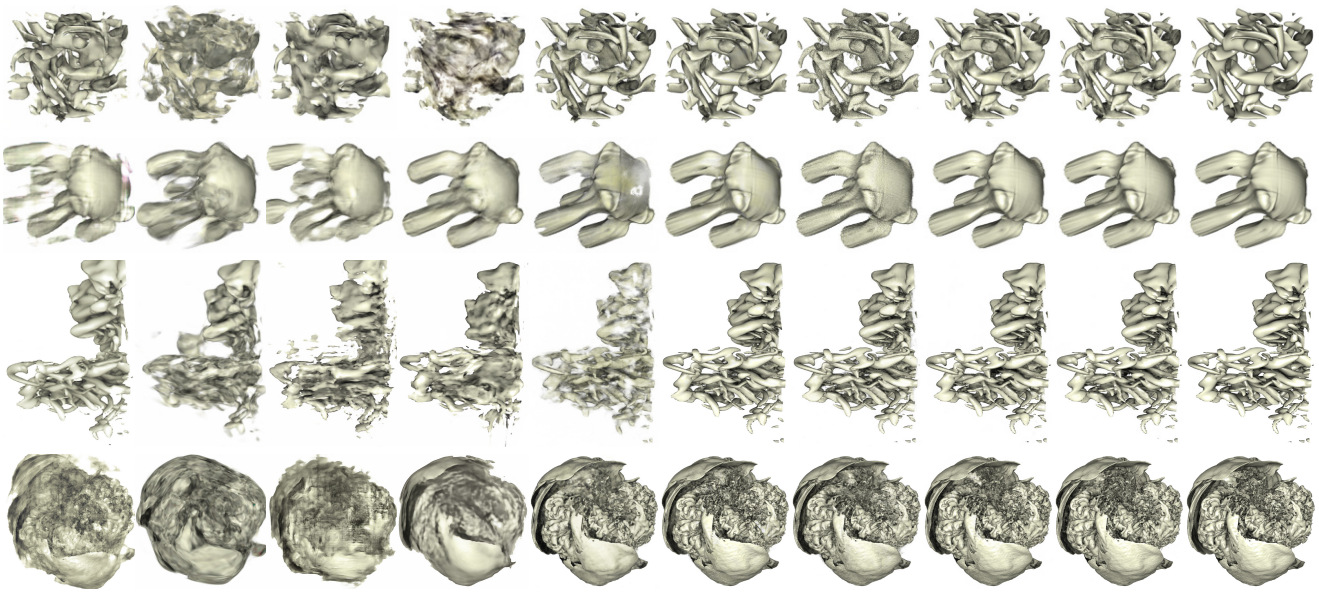
Table 3: Novel view synthesis of static scenes: MS (in MB), TT (in hours), and IT (in minutes) across all 181 synthesized views.

		DVR			IR		
dataset	method	MS↓	TT↓	IT↓	MS↓	TT↓	IT↓
vortex (256×256)	InSituNet	166.78	1.92	0.11	166.78	1.92	**0.08**
	CoordNet	5.71	9.36	0.64	5.71	9.22	0.78
	StyleGAN2	102.62	1.52	0.08	102.62	1.50	**0.08**
	EG3D	145.24	14.73	0.37	145.24	14.70	0.37
	NeRF	**2.27**	27.26	9.40	**2.27**	26.07	9.26
	3DGS	21.64	0.06	**0.07**	37.49	0.07	**0.08**
	Instant-NGP	14.31	**0.05**	0.23	14.49	**0.05**	0.23
	TensoRF	12.41	0.41	0.97	12.51	0.42	1.47
	ViSNeRF	12.57	0.22	0.83	12.60	0.25	0.88
five jets (256×256)	InSituNet	166.78	2.02	0.08	166.78	1.91	0.07
	CoordNet	5.71	10.08	0.66	5.71	9.48	0.71
	StyleGAN2	102.62	1.49	0.10	102.62	1.50	0.10
	EG3D	145.24	14.64	0.37	145.24	14.59	0.32
	NeRF	**2.27**	25.80	9.06	**2.27**	25.66	9.26
	3DGS	18.39	**0.05**	**0.07**	14.22	**0.05**	**0.05**
	Instant-NGP	14.26	**0.05**	0.23	14.29	0.06	0.20
	TensoRF	12.48	0.42	0.73	12.59	0.40	0.77
	ViSNeRF	12.62	0.26	0.83	12.68	0.26	0.77
Tangaroa (1024×1024)	InSituNet	318.73	17.82	**0.37**	318.73	17.80	**0.35**
	CoordNet	**5.71**	44.60	12.43	**5.71**	43.20	11.16
	StyleGAN2	157.68	18.19	1.07	157.68	18.13	1.12
	EG3D	149.76	26.15	1.49	149.75	26.25	1.49
	NeRF	16.36	27.89	148.99	16.36	27.23	148.06
	3DGS	97.88	0.17	0.53	105.33	0.19	0.53
	Instant-NGP	55.87	**0.07**	2.57	57.35	**0.08**	2.03
	TensoRF	68.25	0.94	11.57	68.50	0.73	14.00
	ViSNeRF	69.22	0.35	14.24	68.63	0.41	13.61
supernova (1024×1024)	InSituNet	318.73	17.92	**0.53**	318.73	17.75	**0.41**
	CoordNet	**5.71**	43.17	10.32	**5.71**	44.00	10.47
	StyleGAN2	157.68	18.22	1.91	157.68	18.12	1.59
	EG3D	149.75	26.38	1.35	149.75	26.27	1.32
	NeRF	16.36	27.24	146.05	16.36	27.23	146.46
	3DGS	149.41	0.32	1.22	120.55	0.26	1.07
	Instant-NGP	57.06	**0.08**	6.90	57.17	**0.08**	4.57
	TensoRF	67.16	0.90	23.38	67.22	0.61	17.45
	ViSNeRF	67.29	0.57	35.50	67.22	0.49	24.85



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 1: Novel view synthesis of DVR images for static scenes. (a) to (h): InSituNet, CoordNet, StyleGAN2, EG3D, NeRF, 3DGS, Instant-NGP, TensoRF, ViSNeRF, and GT. Top to bottom: vortex, five jets, Tangaroa, and supernova.



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 2: Novel view synthesis of IR images for static scenes. (a) to (h): InSituNet, CoordNet, StyleGAN2, EG3D, NeRF, 3DGS, Instant-NGP, TensoRF, ViSNeRF, and GT. Top to bottom: vortex, five jets, Tangaroa, and supernova.

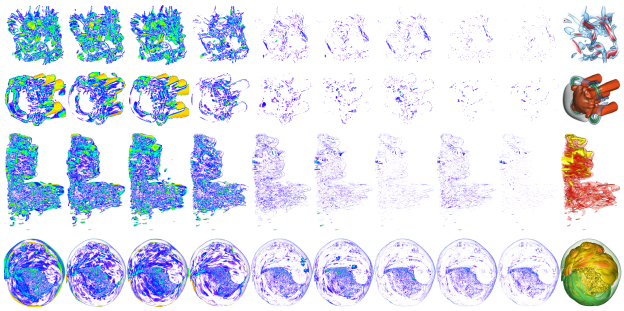
all tested scenarios.

According to Table 3, while ViSNeRF and TensoRF are significantly faster than NeRF, their training and rendering speeds remain slower than 3DGS and Instant-NGP. This is largely due to their implementation in PyTorch rather than CUDA and differences in NeRF representations. While not suitable for real-time rendering, ViSNeRF is still practical for interactive applications. Transitioning to the CUDA framework could be a future improvement, but the current efficiency of ViSNeRF is acceptable. ViSNeRF has relatively close model sizes to 3DGS, Instant-NGP, and TensoRF, as their training configurations are carefully adjusted to ensure fair comparisons with equivalent model capabilities.

From Tables 2 and 3, we can draw similar conclusions as Table 3 in the paper for dynamic scenes when comparing ViSNeRF to InSituNet, CoordNet, StyleGAN2, and EG3D.

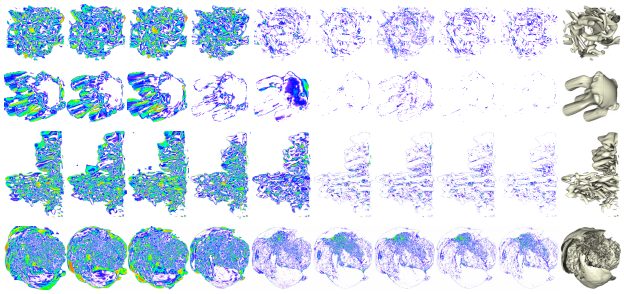
2.4 Qualitative Analysis

Figures 1 and 2 show DVR and IR images synthesized by ViSNeRF and baseline methods for comparison, where the difference images are shown in Figures 3 and 4. The visual quality of images synthesized by ViSNeRF is the best among all these methods. Similar to the results observed in experiments with dynamic scenes, traditional



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 3: Difference images of novel view synthesis of DVR images for static scenes. (a) to (h): InSituNet, CoordNet, StyleGAN2, EG3D, NeRF, 3DGS, Instant-NGP, TensoRF, ViSNeRF, and GT. Top to bottom: vortex, five jets, Tangaroa, and supernova.



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 4: Difference images of novel view synthesis of IR images for static scenes. (a) to (h): InSituNet, CoordNet, StyleGAN2, EG3D, NeRF, 3DGS, Instant-NGP, TensoRF, ViSNeRF, and GT. Top to bottom: vortex, five jets, Tangaroa, and supernova.

2D-based methods such as InSituNet, CoordNet, and StyleGAN2 struggle to reconstruct visualizations accurately. In contrast, NeRF-based methods, including NeRF, 3DGS, Instant-NGP, TensoRF, and ViSNeRF, synthesize visualization images with substantially better visual quality. Among these, ViSNeRF demonstrates finer details in the synthesized images as well as the lowest errors in difference images for both DVR and IR cases due to its ability to handle transparency and lighting with better accuracy and consistency in novel views. This capability also ensures higher fidelity in synthesizing visualizations for dynamic scenes.

3 Baseline Training Details

Like ViSNeRF, all the baselines are trained using a single NVIDIA Tesla V100 graphic card with 32 GB of video memory. While 3DGS and Instant-NGP leverage the CUDA framework to accelerate training and rendering, other baseline methods, including InSituNet, CoordNet, StyleGAN2, 3DGS, NeRF, and TensoRF, are implemented in PyTorch. In Table 4, we provide hyperparameters of the baselines for network training. InSituNet, StyleGAN2, and EG3D are trained with a batch of images per iteration, whereas CoordNet, NeRF, 3DGS, Instant-NGP, TensoRF, and ViSNeRF are trained with a batch of pixels in an image per iteration.

As shown in Tables 5 and 6, in experiments of dynamic scenes, for a single parameter input (e.g., timestep), we found that tripling the number of training iterations compared to the static scene was enough for the models to converge adequately. When two or three input parameters are used (e.g., TF-1 or simulation parameters), the number of training iterations should be 6× or 7.5× for the static scene to train the baselines fully. In contrast, ViSNeRF can still be sufficiently trained for the tripled number of iterations. Other hyperparameters and model configurations of baseline methods, such as the number of channels in each layer and positional encoding computation, follow the original papers and default values provided in the code.

Table 4: Hyperparameters of baseline training. For 1024×1024 resolution, the batch size is reduced due to GPU memory constraint (shown in parentheses).

method	batch size	initial learning rate	β_1	β_2
InSituNet	4 (2)	5×10^{-5}	0	0.999
CoordNet	32,000	10^{-5}	0.9	0.999
StyleGAN2	16 (2)	10^{-3}	0	0.99
EG3D	4 (1)	2.5×10^{-3}	0	0.99
NeRF	4096	5×10^{-4}	0.9	0.999
3DGS	256	1.6×10^{-4}	0.9	0.999
Instant-NGP	262144	10^{-2}	0.9	0.99
TensoRF	4096	0.02	0.9	0.99

Table 5: Training iterations for different numbers of training images with 256×256 resolution. The number of parameters is shown in parentheses (refer to Table 2 in the paper).

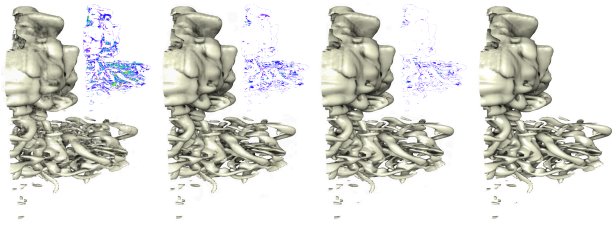
method	42 (1)	462 (1)	1512 (2)	1890 (3)
InSituNet	150,000	450,000	900,000	1,125,000
CoordNet	150,000	450,000	900,000	1,125,000
StyleGAN2	150,000	450,000	900,000	1,125,000
EG3D	150,000	450,000	900,000	1,125,000
NeRF	100,000	—	—	—
3DGS	30,000	—	—	—
Instant-NGP	35,000	—	—	—
TensoRF	30,000	—	—	—
ViSNeRF	30,000	90,000	90,000	90,000

Table 6: Training iterations for different numbers of training images with 1024×1024 resolution. The number of parameters is shown in parentheses (refer to Table 2 in the paper).

method	42 (1)	546 (1)
InSituNet	150,000	450,000
CoordNet	600,000	1,800,000
StyleGAN2	150,000	450,000
EG3D	150,000	450,000
NeRF	100,000	—
3DGS	30,000	—
Instant-NGP	35,000	—
TensoRF	30,000	—
ViSNeRF	30,000	90,000

Table 7: Hyperparameter study: number of training images using Tangaroa IR images.

# training images	12	42	92
PSNR↑	24.16	32.23	**33.72**
SSIM↑	0.937	0.984	**0.988**
LPIPS↓	0.053	0.016	**0.013**

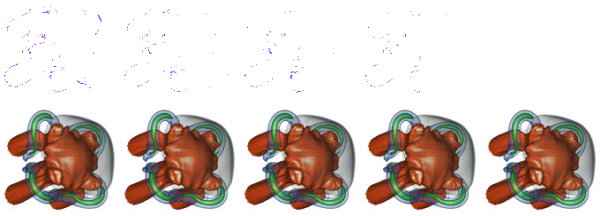


(a) 12 (b) 42 (c) 92 (d) GT

Figure 5: Inferred Tangaroa IR images using different numbers of training images.

Table 8: Hyperparameter study: number of training iterations using five jets DVR images.

# iterations	10,000	20,000	30,000	40,000
PSNR↑	33.02	34.83	35.80	**35.98**
SSIM↑	0.985	0.990	**0.992**	**0.992**
LPIPS↓	0.010	0.007	**0.005**	**0.005**

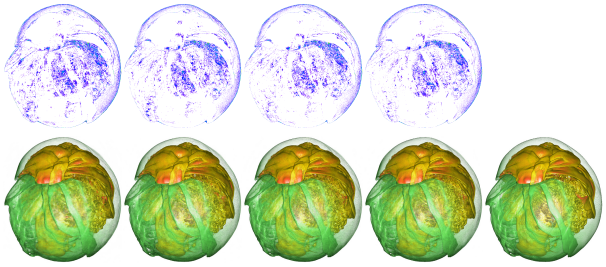


(a) 10,000 (b) 20,000 (c) 30,000 (d) 40,000 (e) GT

Figure 6: Inferred five jets DVR images using different numbers of training iterations.

Table 9: Hyperparameter study: number of feature grid voxels using supernova DVR images.

# voxels	200^3	300^3	400^3	500^3
PSNR↑	26.01	27.01	27.17	**27.22**
SSIM↑	0.821	0.859	0.864	**0.865**
LPIPS↓	0.183	0.120	0.113	**0.108**

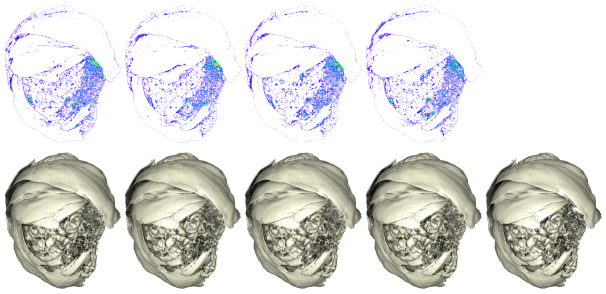


(a) 200^3 (b) 300^3 (c) 400^3 (d) 500^3 (e) GT

Figure 7: Inferred supernova DVR images using different numbers of feature grid voxels.

Table 10: Hyperparameter study: depth of density feature using supernova IR images.

depth	8	16	24	32
PSNR↑	29.08	29.71	**29.79**	29.77
SSIM↑	0.937	0.946	**0.947**	**0.947**
LPIPS↓	0.072	**0.049**	0.051	0.050



(a) 8 (b) 16 (c) 24 (d) 32 (e) GT

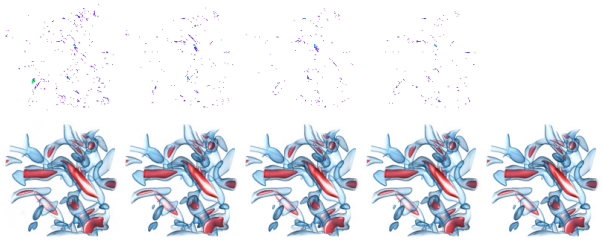
Figure 8: Inferred supernova IR images using different depths of density feature.

Table 11: Hyperparameter study: depth of appearance feature using vortex DVR images.

depth	32	40	48	56
PSNR↑	35.25	36.83	37.32	**38.04**
SSIM↑	0.993	0.995	0.996	**0.997**
LPIPS↓	0.005	0.004	**0.003**	**0.003**

4 Hyperparameter Study

We investigate several key hyperparameters for ViSNeRF, including the number of training images, training iterations, feature grid voxels, and depths of density feature and appearance feature.



(a) 32 (b) 40 (c) 48 (d) 56 (e) GT

Figure 9: Inferred vortex DVR images using different depths of appearance feature.

Table 12: Unified vs. split feature grids: average PSNR (dB), SSIM, LPIPS, MS (in MB), TT (in hours), and IT (in minutes) across all 181 synthesized supernova IR views.

feature grid	PSNR↑	SSIM↑	LPIPS↓	MS↓	TT↓	IT↓
unified	29.64	0.943	0.056	**67.22**	0.98	33.22
split	**29.71**	**0.946**	**0.049**	**67.22**	**0.62**	**24.13**

Table 13: Tensor decomposition: average PSNR (dB), SSIM, and LPIPS across all 181 synthesized supernova DVR views.

decomposition	PSNR↑	SSIM↑	LPIPS↓
vectors	22.53	0.717	0.313
matrices	25.83	0.827	0.155
vectors+matrices	**27.01**	**0.859**	**0.120**

Number of training images. Reconstruction accuracy largely depends on the number of training images available to ViSNeRF. As shown in Table 7, from 12 to 42 images, the additional 30 images boost PSNR from 24.16 dB to 32.23 dB. Adding another 50 extra images improves PSNR only by 1.49 dB with slightly fewer errors shown in Figure 5. We conclude that 42 images are sufficient for ViSNeRF to build knowledge of the entire visualization for most cases. However, more images will help refine the reconstructed visualization's details, such as highlights, shadows, and ambient occlusion.

Number of training iterations. As Table 8 and Figure 6 suggest, ViSNeRF reaches satisfactory convergence after 30,000 iterations for a single static visualization. While increasing the number of iterations may slightly improve the performance, we set the training iterations to 30,000 for all static visualizations to standardize the training process.

Number of feature grid voxels. As indicated by Table 9 and Figure 7, increasing the number of feature grid voxels may help restore fine details in the visualization. In contrast, a sparse feature grid could result in visualizations with reduced clarity. However, increasing the size of the feature grid will significantly increase the model size and training time. For example, if we increase the feature grid from 300^3 to 400^3 for the supernova dataset, the mode size increases from 67 MB to 119 MB, and the training time increases from 34 to 59 minutes. To strike a balance, ViSNeRF has 300^3 voxels in the finest feature grid for all the cases.

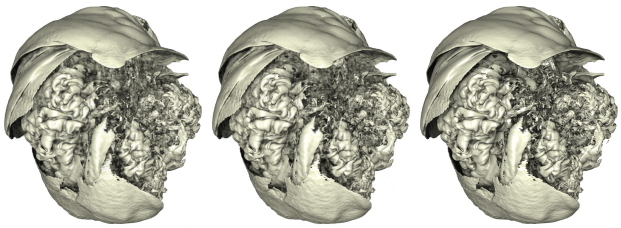
Depths of density feature and appearance feature. From Tables 10 and 11 and Figures 8 and 9, we find that the depths of density feature and appearance feature are minor factors to the reconstruction quality. The results suggest that the optimal depth for the density feature is 16, while that for the appearance feature is 48. Increasing the depth beyond these values yields little quality improvement.

5 Additional Results and Discussion

Split feature grid. By splitting the feature grid, as shown in Table 12, ViSNeRF achieves reduced training and inference times while maintaining the quality of the synthesized images. Note that, to maintain the model size, we set the depth of features in the unified

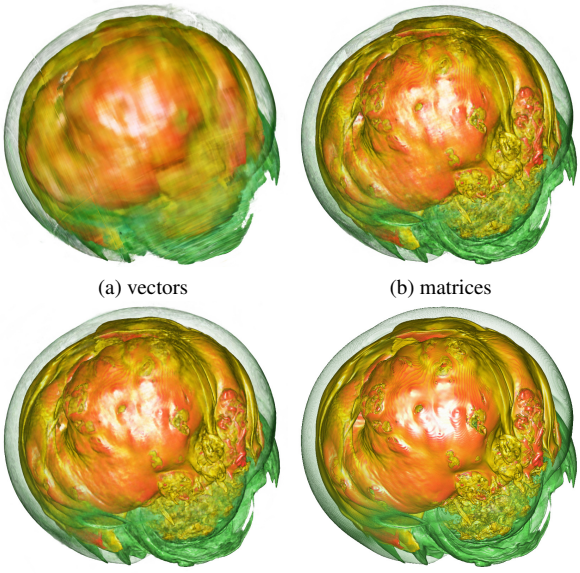
Table 14: DVR or IR vs. ViSNeRF for dynamic scenes. All timing numbers reported are in minutes, and data/image sizes or memory consumptions reported are in GB.

	DVR or IR			ViSNeRF			
dataset (scenario)	data size	processing time	RAM/VRAM consumption	image size	training time	inference time	RAM/VRAM consumption
five jets (timestep)	1.414	14.06	2.10/1.10	0.137	85.80	21.20	27.94/5.83
Tangaroa (isovalue)	0.877	1.51	1.00/0.27	0.117	149.68	31.42	27.77/9.91
vortex (TF-2)	0.008	0.23	1.01/1.26	0.010	84.92	1.45	3.87/5.24
Nyx-DVR (simulation parameters)	8.188	14.90	2.90/1.15	0.119	99.53	1.62	7.97/5.58



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

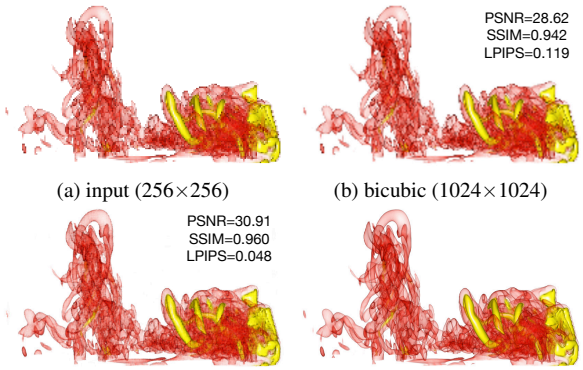
Figure 10: Unified vs. split feature grids: ViSNeRF-synthesized supernova IR images.



(a) vectors (b) matrices

(c) vectors+matrices (d) GT

Figure 11: Tensor decomposition: ViSNeRF-synthesized supernova DVR images.



(a) input (256×256) (b) bicubic (1024×1024)

(c) ViSNeRF (1024×1024) (d) GT (1024×1024)

Figure 12: Super-resolution synthesis: low-resolution input and high-resolution synthesized Tangaroa DVR images.

grid to 64, the sum of the depths of split feature grids. As shown in Figure 10, with split feature grids, ViSNeRF generates synthesized images of similar quality compared to those produced using a unified approach.

Tensor decomposition. We investigate the effectiveness of vector-matrix decomposition by ablating ViSNeRF regarding the factorization of the 3D volume representing the radiance fields. As shown in Figure 11, with vector decomposition, reconstructed visualization is blurry. Although matrix decomposition is sufficient to restore the visualization details, thinner areas, especially the shell colored in green, are not correctly synthesized. By employing vector-matrix decomposition, as shown in Table 13, the performance of ViSNeRF is further improved.

Super-resolution synthesis. A by-product of VisReRF is the ability to synthesize super-resolution rendering images from low-resolution training images. With an optimized NeRF, visualizations of any resolutions can be rendered by the volume renderer of ViSNeRF. As shown in Figure 12, leveraging the knowledge acquired from training 256×256 resolution images, ViSNeRF demonstrates its capability to restore images at a higher resolution of 1024×1024. Most of the details in the high-resolution images can be recovered by ViSNeRF despite noticeable artifacts due to a limited number of 42 training images.

Traditional rendering vs. ViSNeRF. In Table 14, we compare traditional DVR and IR with ViSNeRF regarding memory consumption and rendering time using the same GPU. Our results indicate that traditional DVR and IR necessitate access to the original volume data or isosurfaces, which can be sizeable, often reaching multiple gigabytes in the case of time-varying or ensemble datasets. An exception is observed in the vortex dataset, where only a small volume of 8 MB is necessary due to the interpolation across transfer functions. In contrast, ViSNeRF operates primarily on rendered images (PNG format), which typically occupy less storage than volume data or isosurfaces. To illustrate, around 100 MB is sufficient for processing 546 images of Tangaroa and 1890 images of Nyx, and notably, only 10 MB is needed for 462 images of vortex. This storage efficiency is significant, especially considering ViSNeRF's ability to produce high-quality visualization synthesis from interpolated parameters without the actual data.

Regarding generation time to produce all rendering or inference images, although traditional DVR and IR include the file reading time, they are still more efficient than ViSNeRF, which requires training. However, ViSNeRF exhibits a faster generation time for Nyx datasets of low image resolution (256×256). Regarding memory usage, ViSNeRF inherently demands more RAM and VRAM, a common trait of machine learning methods. Before training commences, ViSNeRF must load training and inference images, converting them into a data structure of rays and associated attributes. Each ray consists of ray origin and direction (six floats), RGB values (three floats), and parameters (one float per parameter), totaling approximately ten floats per pixel. Given that each 256×256 image comprises 65,536 pixels, the memory requirement for one image can be calculated as $65,536 \times 10 \times 4 = 2,621,440$ bytes, equating to roughly 2.5 MB per image. For a dataset like vortex, which includes 462 training images

and 181 inference images of 256×256 resolution, the total memory footprint for storing these images as rays would be approximately 1.57 GB.

Considering the Tangaroa dataset, which utilizes images of resolution 1024×1024, the initial memory requirement for storing image data in ViSNeRF is substantial. The calculation for this dataset would be $1024 \times 1024 \times 10 \times 4 \times (546 + 181) = 30,492,590,080$ bytes, amounting to approximately 29 GB. However, ViSNeRF employs an optimization by filtering out rays that do not intersect with the bounding box, effectively reducing the long-term runtime RAM usage below 29 GB for the Tangaroa dataset. Most of the remaining memory consumption, particularly VRAM, is attributed to the model training process. This includes the storage of the density feature grid, color feature grid, parameter feature grid, two small MLPs, and their gradient graphs, which are essential for model optimization. Given these requirements, we recommend a minimum of 32 GB of RAM and 10 GB of VRAM for running ViSNeRF efficiently. A configuration with at least 64 GB of RAM and 16 GB of VRAM is ideal for optimal performance, especially with high-resolution datasets.

REFERENCES

[1] A. S. Almgren, J. B. Bell, M. J. Lijewski, Z. Lukic, and E. V. Andel. Nyx: A massively parallel AMR code for computational cosmology. *The Astrophysical Journal*, 765(1):39:1–39:14, 2013.

[2] D. Bauer, Q. Wu, and K.-L. Ma. FoVolNet: Fast volume rendering using foveated deep neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):515–525, 2023.

[3] M. Berger, J. Li, and J. A. Levine. A generative model for volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 25(4):1636–1650, 2019.

[4] A. Cao and J. Johnson. HexPlane: A fast representation for dynamic scenes. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 130–141, 2023.

[5] E. R. Chan, C. Z. Lin, M. A. Chan, K. Nagano, B. Pan, S. De Mello, O. Gallo, L. J. Guibas, J. Tremblay, S. Khamis, T. Karras, and G. Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 16123–16133, 2022.

[6] E. R. Chan, M. Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein. Pi-GAN: Periodic implicit generative adversarial networks for 3D-aware image synthesis. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5799–5809, 2021.

[7] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su. TensoRF: Tensorial radiance fields. In *Proceedings of European Conference on Computer Vision*, pp. 333–350, 2022.

[8] T. Devries, M. Á. Bautista, N. Srivastava, G. W. Taylor, and J. M. Susskind. Unconstrained scene generation with locally conditioned radiance fields. In *Proceedings of IEEE International Conference on Computer Vision*, pp. 14284–14293, 2021.

[9] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5501–5510, 2022.

[10] M. Gadelha, S. Maji, and R. Wang. 3D shape induction from 2D views of multiple objects. In *Proceedings of International Conference on 3D Vision*, pp. 402–411, 2017.

[11] J. Gu, L. Liu, P. Wang, and C. Theobalt. StyleNeRF: A style-based 3D aware generator for high-resolution image synthesis. In *Proceedings of International Conference on Learning Representations*, 2022.

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

[13] P. Gu, J. Han, D. Z. Chen, and C. Wang. Scalar2Vec: Translating scalar fields to vector fields via deep learning. In *Proceedings of IEEE Pacific Visualization Symposium*, pp. 31–40, 2022.

[14] L. Guo, S. Ye, J. Han, H. Zheng, H. Gao, D. Z. Chen, J.-X. Wang, and C. Wang. SSR-VFD: Spatial super-resolution for vector field data analysis and visualization. In *Proceedings of IEEE Pacific Visualization Symposium*, pp. 71–80, 2020.

[15] J. Han, J. Tao, H. Zheng, H. Guo, D. Z. Chen, and C. Wang. Flow field reduction via reconstructing vector data from 3D streamlines using deep learning. *IEEE Computer Graphics and Applications*, 39(4):54–67, 2019.

[16] J. Han and C. Wang. TSR-VFD: Generating temporal super-resolution for unsteady vector field data. *Computers & Graphics*, 103:168–179, 2022.

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

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

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

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

[21] P. Henzler, N. Mitra, and T. Ritschel. Escaping Plato's Cave: 3D shape from adversarial rendering. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9983–9992, 2019.

[22] F. Hong, C. Liu, and X. Yuan. DNN-VolVis: Interactive volume visualization supported by deep neural network. In *Proceedings of IEEE Pacific Visualization Symposium*, pp. 282–291, 2019.

[23] Y. Huang, B. Cui, L. Bai, Z. Guo, M. Xu, M. Islam, and H. Ren. Endo-4DGS: Endoscopic monocular scene reconstruction with 4D Gaussian splatting. In *Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention*, vol. 15006, pp. 197–207, 2024.

[24] Y. Huang, Y. Sun, Z. Yang, X. Lyu, Y. Cao, and X. Qi. SC-GS: Sparse-controlled Gaussian splatting for editable dynamic scenes. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4220–4230, 2024.

[25] A. Jain, B. Mildenhall, J. T. Barron, P. Abbeel, and B. Poole. Zero-shot text-guided object generation with dream fields. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 867–876, 2022.

[26] A. Jain, M. Tancik, and P. Abbeel. Putting NeRF on a diet: Semantically consistent few-shot view synthesis. In *Proceedings of IEEE International Conference on Computer Vision*, pp. 5885–5894, 2021.

[27] J. T. Kajiya and B. P. Von Herzen. Ray tracing volume densities. In *Proceedings of ACM SIGGRAPH Conference*, pp. 165–174, 1984.

[28] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila. Alias-free generative adversarial networks. In *Proceedings of Advances in Neural Information Processing Systems*, pp. 852–863, 2021.

[29] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and improving the image quality of StyleGAN. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8107–8116, 2020.

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

[31] Y. Liao, K. Schwarz, L. Mescheder, and A. Geiger. Towards unsupervised learning of generative models for 3D controllable image synthesis. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5871–5880, 2020.

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

[33] N. Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995.

[34] B. Mildenhall, P. Hedman, R. Martin-Brualla, P. P. Srinivasan, and J. T. Barron. NeRF in the dark: High dynamic range view synthesis from noisy raw images. In *Proceedings of IEEE Conference on Computer*

Vision and Pattern Recognition, pp. 16190–16199, 2022.

[35] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proceedings of European Conference on Computer Vision*, pp. 405–421, 2020.

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

[37] T. Nguyen-Phuoc, C. Li, L. Theis, C. Richardt, and Y. Yang. HoloGAN: Unsupervised learning of 3D representations from natural images. In *Proceedings of IEEE International Conference on Computer Vision*, pp. 7587–7596, 2019.

[38] T. Nguyen-Phuoc, C. Richardt, L. Mai, Y.-L. Yang, and N. Mitra. BlockGAN: Learning 3D object-aware scene representations from unlabelled images. In *Proceedings of Advances in Neural Information Processing Systems*, 2020.

[39] M. Niemeyer and A. Geiger. GIRAFFE: Representing scenes as compositional generative neural feature fields. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11453–11464, 2021.

[40] M. Oechsle, S. Peng, and A. Geiger. UNISURF: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *Proceedings of IEEE International Conference on Computer Vision*, pp. 5589–5599, 2021.

[41] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of IEEE International Conference on Computer Vision*, pp. 5865–5874, 2021.

[42] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer. D-NeRF: Neural radiance fields for dynamic scenes. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10318–10327, 2021.

[43] Sara Fridovich-Keil and Giacomo Meanti, F. R. Warburg, B. Recht, and A. Kanazawa. K-Planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 12479–12488, 2023.

[44] J. L. Schönberger, E. Zheng, J.-M. Frahm, and M. Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *Proceedings of European Conference on Computer Vision*, pp. 501–518, 2016.

[45] K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger. GRAF: Generative radiance fields for 3D-aware image synthesis. In *Proceedings of Advances in Neural Information Processing Systems*, 2020.

[46] R. Shao, Z. Zheng, H. Tu, B. Liu, H. Zhang, and 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*, pp. 16632–16642, 2023.

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

[48] U. Singer, S. Sheynin, A. Polyak, O. Ashual, I. Makarov, F. Kokkinos, N. Goyal, A. Vedaldi, D. Parikh, J. Johnson, and Y. Taigman. Text-to-4D dynamic scene generation. *arXiv:2301.11280*, 2023.

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

[50] K. Tang and C. Wang. STSR-INR: Spatiotemporal super-resolution for time-varying multivariate volumetric data via implicit neural representation. *Computers & Graphics*, 119:103874, 2024.

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

[52] K. Tang, S. Yao, and 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. Accepted.

[53] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

[54] C. Wang, M. Chai, M. He, D. Chen, and J. Liao. CLIP-NeRF: Text-and-image driven manipulation of neural radiance fields. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3835–3844, 2022.

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

[56] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang. NeuS: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *Proceedings of Advances in Neural Information Processing Systems*, pp. 27171–27183, 2021.

[57] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[58] J. Weiss and N. Navab. Deep direct volume rendering: Learning visual feature mappings from exemplary images. *arXiv:2106.05429*, 2021.

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

[60] S. Weiss, P. Hermüller, and R. Westermann. Fast neural representations for direct volume rendering. *Computer Graphics Forum*, 41(6):196–211, 2022.

[61] S. Weiss, M. Işlk, J. Thies, and R. Westermann. Learning adaptive sampling and reconstruction for volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 28(7):2654–2667, 2022.

[62] G. Wu, T. Yi, J. Fang, L. Xie, X. Zhang, W. Wei, W. Liu, Q. Tian, and X. Wang. 4D Gaussian splatting for real-time dynamic scene rendering. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 20310–20320, 2024.

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

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

[65] Z. Yang, X. Gao, W. Zhou, S. Jiao, Y. Zhang, and X. Jin. Deformable 3D Gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 20331–20341, 2024.

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

[67] S. Yao, W. Song, and C. Wang. A comparative study of neural surface reconstruction for scientific visualization. In *Proceedings of IEEE VIS Conference (Short Papers)*, pp. 186–190, 2024.

[68] L. Yariv, J. Gu, Y. Kasten, and Y. Lipman. Volume rendering of neural implicit surfaces. In *Proceedings of Advances in Neural Information Processing Systems*, pp. 4805–4815, 2021.

[69] J. Ye, L. Wang, G. Li, D. Chen, S. Zhe, X. Chu, and Z. Xu. Learning compact recurrent neural networks with block-term tensor decomposition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9378–9387, 2018.

[70] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *Proceedings of IEEE International Conference on Computer Vision*, pp. 5752–5761, 2021.

[71] A. Yu, V. Ye, M. Tancik, and A. Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4578–4587, 2021.

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

[73] J.-Y. Zhu, Z. Zhang, C. Zhang, J. Wu, A. Torralba, J. Tenenbaum, and B. Freeman. Visual object networks: Image generation with disentangled 3D representations. In *Proceedings of Advances in Neural Information Processing Systems*, pp. 118–129, 2018.