Computers & Graphics (2023)

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

NeRVI: Compressive Neural Representation of Visualization Images for Communicating Volume Visualization Results

Pengfei Gu[a], Danny Z. Chen[a], Chaoli Wang[a]

[a]*Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, United States*

ARTICLE INFO

Article history:

Keywords: Volume visualization, Implicit neural representation, Image compression, Deep learning

ABSTRACT

We present NeRVI, a new deep-learning approach that compresses a large collection of visualization images generated from time-varying data for communicating volume visualization results. Based on an image-based implicit neural representation, our approach represents tens of thousands of high-resolution rendering images parametrized by different parameters via a hybrid model of multilayer perceptrons and convolutional neural networks. Our model predicts images and corresponding masks, and the masks are utilized for loss computation and network training to capture fine structural details and small components. In conjunction with model quantization and weight encoding, NeRVI yields highly compact compressive neural representations while preserving the image fidelity well. We demonstrate the effectiveness of NeRVI with isosurface rendering and direct volume rendering images generated from multiple data sets and compare NeRVI with other state-of-the-art deep learning-based (InSituNet, SIREN, NeRF, and NeRV) methods. Quantitative and qualitative results show that NeRVI provides an alternative solution that augments domain scientists' ability to manage, represent, and communicate scientific visualization output.

1. Introduction

In many scientific applications, scientists generate time-varying data from large-scale simulations to study various phenomena and produce high-resolution, high-quality visualization images for post hoc analysis and communication. Sharing such visualization images instead of the original large-scale data to end users (such as peer scientists or the general public) is meaningful in different scenarios, for example, when scientists are hesitant to make their simulation data publicly available, when visualizations need to be carefully prepared by visualization professionals to maintain high quality, or when end users cannot easily produce such rendering results due to lack of knowledge skills or hardware support (e.g., memory or graphics).

In practice, given a scalar field data set at a particular time step, many volume visualization images, such as isosurface rendering (IR) or direct volume rendering (DVR) images, could be generated. These images sample data parameters (e.g., a set of predetermined isovalues for IR), visual mapping parameters (e.g., color and opacity transfer function parameters for DVR), and viewing parameters (e.g., viewpoints) to present a comprehensive overview of the underlying data. For a time-varying data set, the corresponding visualization images can easily reach tens of thousands and occupy tens of gigabytes of space, which could be even larger than the data itself. This poses severe constraints, including disk storage, network bandwidth, accessibility, and interactivity, for communicating the visualization output to end users. Hence, effectively compressing and sharing such visualization images is highly desirable.

We study the problem of compressing a large collection of visualization images generated from time-varying data to effectively communicate volume visualization results. Given a time-varying data set, we produce a collection of visualization images under data (e.g., isovalues), visual mapping (e.g., color and opacity transfer function), and viewing (i.e., spherical an-

gle coordinates θ and ϕ for sampling viewpoints) parameters. Instead of following a conventional image compression solution, we aim to represent the generated visualization images via a novel deep learning approach inspired by the recent *implicit neural representation* (INR) work for video compression [1]. With this technique, the trained neural network *itself* becomes the compressed representation of the images. The visualization images can be inferred by looping through the sampled parameters used during training.

Achieving high-quality compression of visualization images in the scenario mentioned above presents several unique challenges. First, compressing visualization images is more challenging than compressing video frames because more parameters beyond the time step need to be considered. A significant difference often exists between two "neighboring" rendering images sampled from a scalar field data set. However, video frames typically exhibit better frame-to-frame coherence within a given shot. Second, fitting visualization images generated from different time steps of a time-varying data set under the same neural network brings additional obstacles because these images could encompass dramatically different temporal patterns over time. Furthermore, the stored time steps are often sparse samples from the simulation iterations, exacerbating the concern of temporal coherence. Third, unlike video frames, rendering images always have foreground and background pixels. As the foreground and background regions keep changing when varying the parameter values, the ratio of foreground pixels to background pixels could vary a lot due to different time steps, isovalues, and viewing parameter settings. Therefore, this foreground-background issue must be considered for achieving effective, high-quality compression.

To address these challenges, we present NeRVI, a new solution for **Ne**ural **R**epresentation of **V**isualization **I**mages. NeRVI advocates coarse-grained image-based input to efficiently learn a large collection of visualization images parameterized by different parameters. The resulting hybrid *multilayer perceptrons* (MLP) + *convolutional neural network* (CNN) model yields a compressive neural representation that is highly compact while supporting explorable visualization of high-fidelity rendering results via an interactive visual interface.

The inputs of NeRVI are the given parameters, and the outputs are the rendering and corresponding mask images. We minimize the errors between the predicted and ground truth (GT) rendering images over only foreground pixels and the errors between the predicted and GT masks. Then, model quantization and weight encoding are leveraged during post-training to achieve further compression. Finally, the compressed images could be extracted by feeding the parameters. The quantized model can also be utilized to interpolate visualization images. We quantitatively and qualitatively evaluate NeRVI with IR and DVR images generated from multiple data sets. The results show that NeRVI outperforms state-of-the-art deep learning methods: InSituNet, SIREN, NeRF, and NeRV.

The contributions of our work are as follows. First, we redesign a more powerful image-wise INR for compressive representation of visualization images featuring SIREN-based residual block and mask loss, achieving a high compression rate (705 to 1,263). Second, we show NeRVI performs better than four deep learning-based methods with IR and DVR images on multiple data sets. Third, we conduct a parameter study to investigate the impact of critical parameters (initial channels, input image resolution, and viewpoint sampling degree) on the performance of NeRVI.

2. Related Work

Image-based approach for volume visualization. With a similar goal for exploratory visualization, previous work has explored an image-based approach for volume visualization. Tikhonova et al. [2] presented the concept of explorable image (EI), a compact representation that enables interactive exploration of volume data in the transfer function space. They also extended EI for visualizing time-varying volume data [3] and utilized proxy images for volume visualization, enabling interactive exploration [4]. Frey et al. [5] proposed volumetric depth images (VDIs) for view-dependent volume visualization. Fernandes et al. [6] introduced space-time VDIs for in situ visualization of time-varying volumetric data. Ahrens et al. [7, 8] developed Cinema, a new image-based approach for extreme-scale in situ visualization and analysis. Biedert and Garth [9] combined in situ topological contour tree analysis and compact image-based data representation for explorative visualization and analysis while preserving flexibility. He et al. [10] proposed InSituNet, which infers visualization results from simulation and visualization parameters via deep learning. Jiao et al. [11] introduced an enhanced super-resolution generative adversarial network (ESRGAN) and a reduction-restoration workflow to obtain visualization results from reduced data without losing too many features.

INR for computer vision and graphics. Since 2019, the computer vision and graphics community has studied the use of INR for solving various problems [12]. Early work, including DeepSDF [13], occupancy networks [14], and IM-NET [15], first shows that INR outperforms point-, grid-, and mesh-based representations in parameterizing geometry and enables learning priors over shapes seamlessly. Later, researchers demonstrated the learning of *signed distance field* (SDF) from raw data [16, 17] or *unsigned distance field* (UDF) from raw point clouds [18] and proposed hybrid solutions incorporating voxel grids and INR to handle large-scale 3D scenes [19, 20, 21].

Pixel-wise or coordinate-based INR aims to model an input signal as a continuous function that maps the domain of the input signal (i.e., coordinate) to the value at that coordinate (e.g., RGB color of an image). Sitzmann et al. [22] proposed *scene representation network* (SRN), a 3D-structure-aware network that trains with only posed 2D images to reconstruct objects, encoding both geometry and appearance. Sitzmann et al. [23] introduced *sinusoidal representation network* (SIREN) that employs periodic activation functions for continuous INR, fitting complicated signals and their derivatives. They demonstrated the application of SIREN in audio, image, and video representation, 3D shape reconstruction, and solving first- and second-order differential equations. Mildenhall et al. [24] designed *neural radiance field* (NeRF) that leverages a fully-connected deep neural network to infer color and density values of novel

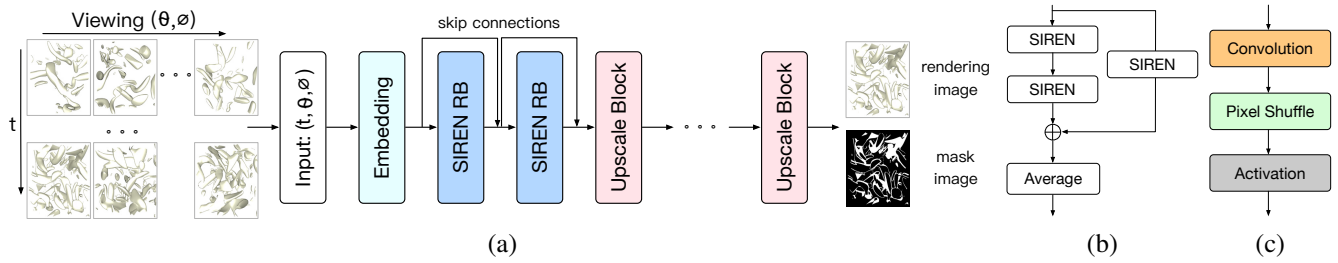


Fig. 1. (a) The image-wise INR of NeRVI takes parameters $(t;\theta;\phi)$ as input and outputs the whole rendering image and the corresponding mask image. (b) Our SIREN-based residual block (SIREN RB). (c) The upscale block.

views given 3D spatial locations and viewing directions. This seminal NeRF work has inspired a myriad of subsequent works on *neural rendering* [25], defined as "*deep image or video generation approaches that enable explicit or implicit control of scene properties*" [26]. Niemeyer et al. [27] presented *differentiable volumetric rendering*, a differentiable rendering pipeline that learns 3D representations from 2D RGB images. Sitzmann et al. [28] developed MetaSDF, which incorporates meta-learning with INR to learn the 3D shape space. Chan et al. [29] proposed Pi-GAN, combining neural representation and neural volume rendering to synthesize 3D-aware images under different views. Guo et al. [30] presented object-centric neural scattering functions to compose photorealistic scenes from captured 2D images.

INR for scientific visualization. In scientific visualization, despite the impressive growth of DL4SciVis research [31], only a few works have leveraged INRs. Lu et al. [32] designed neurcomp, a SIREN-like MLP for neural representations of scalar volumetric data. They achieved an impressive compression rate (over $1,000\times$) while preserving important volumetric features by quantizing network weights. Weiss et al. [33] presented fV-SRN, a fast version of SRN for DVR. fV-SRN leverages GPU tensor cores to integrate the reconstruction task into on-chip raytracing kernels and supports random access reconstruction at arbitrary granularity for temporal reconstruction tasks. Han and Wang [34] designed CoordNet, a single INR framework tackling different tasks for time-varying volumetric data, including spatial super-resolution, temporal super-resolution, view synthesis, and ambient occlusion prediction. Wu et al. [35] proposed an INR for volumetric data that can be trained instantaneously and rendered interactively, achieved by utilizing a CUDA machine learning network, GPU tensor cores, and optimized online training and rendering implementations. In flow visualization, a *flow map* is a mathematical object representing how a particle is transported according to an unsteady flow, starting at a given space-time location and for a given duration. Han et al. [36] developed an MLP-based model for learning particle end locations given their start locations and file cycles. The trained model can predict new particle trajectories with a small memory cost and fast inference. Instead of directly predicting flow maps, Sahoo et al. [37] learned a function-space representation of the flow field and explicitly imposed an integration scheme to ensure a more stable and guided optimization process for neural flow map reconstruction. Wu et al. [38] presented a distributed volumetric neural representation approach designed for in situ visualization and analysis.

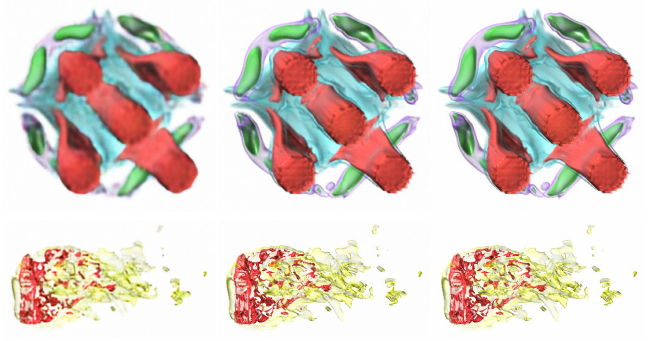
INR for image compression. Dupont et al. [39] designed COIN to demonstrate the feasibility of using INR for image compression tasks. Their key idea was to utilize MLPs that map pixel locations to RGB values to fit images. They then stored the weights of the neural network overfitted to the images instead of keeping the RGB values for each pixel. Although COIN achieves promising performance, it is not yet competitive with state-of-the-art compression methods, and its encoding and decoding speeds are rather slow. To overcome these limitations, Chen et al. [1] proposed NeRV, a novel image-wise implicit representation for video compression. Previous INR methods (e.g., [39]) used MLPs to approximate the INRs, taking the spatial or spatiotemporal coordinate as the input and outputting the signals (e.g., RGB value, volume density) at that single coordinate. Instead, NeRV trains a deep neural network composed of MLPs and convolution layers, ingesting the frame index and directly producing all the RGB values of that frame. Our proposed NeRVI leverages the idea of NeRV. To meet the needs and answer the challenges of compressing tens of thousands of visualization images, we modify the NeRV network architecture accordingly and introduce a new mask loss to address the foreground-background issue.

3. NeRVI

Most INR designs are coordinate-based [23, 24]. The model accepts coordinates (i.e., pixel locations) as input and produces RGB values at these coordinates. For each sample image, an entire feedforward pass through the network must be computed for all pixel coordinates. As such, pure MLP-based INR either often demands excessively long training time or has to limit the number of training samples (up to a few hundred images) for efficiency. To achieve a compressive representation of visualization images, we seek an alternative network design for NeRVI that can ingest a large collection of high-resolution rendering images parameterized by different parameters, e.g., viewpoints θ and ϕ, time steps t, isovalues v, etc.

Formally, given a set of data $\mathbf{D} = \{\mathbf{D}_1, \mathbf{D}_2, \cdots, \mathbf{D}_T\}$, where T is the total number of time steps, we perform visualization with a predefined isovalue or transfer function and under different viewpoints to generate a set of visualization images for data $\mathbf{D}_t$, where $t \in [1, T]$. We represent each $\mathbf{D}_t$ as a set of viewpoints and their corresponding visualization images $\mathbf{I}_t$. Considering the time step parameter, we represent $\mathbf{D} = \{\mathbf{D}_1, \mathbf{D}_2, \cdots, \mathbf{D}_T\}$ as a set of parameters (i.e., $(t;\theta;\phi)$) and their corresponding visualization images $\mathbf{I}$. Our goal is to learn a mapping from the input

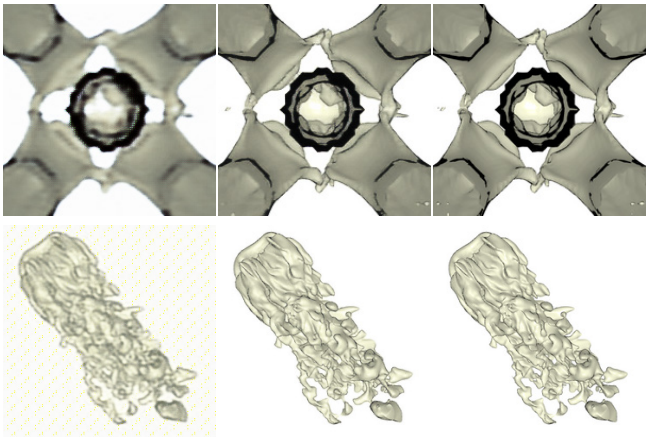
parameters to visualization images, i.e., $f(t;\theta;\phi) = I$. The main idea is to fit the images into a neural network. The network's input is a set of parameters, and its outputs are the whole image and its corresponding mask image (as shown in Figure 1 (a)). Once trained, we can directly extract the visualization image by evaluating the model given the parameters $(t;\theta;\phi)$.



(a) FC w/ ReLU (b) SIREN RB (c) GT

Fig. 2. Compressed DVR images using different MLP designs for the model. Top and bottom: five jets and Tangaroa.

Overview. Our proposed method consists of three stages: image fitting, model compression, and image extraction. In the first stage, we design NeRVI, which includes an MLP encoder and a CNN decoder, to fit all the visualization images. The trained NeRVI could represent the compressed images. In the second stage, we utilize model quantization and weight encoding on the trained model to achieve further compression, following NeRV [1]. In the last stage, the compressed images could be extracted by feeding the parameters to the quantized model. Our method could also be utilized for interpolating visualization images by supplying unseen parameters.

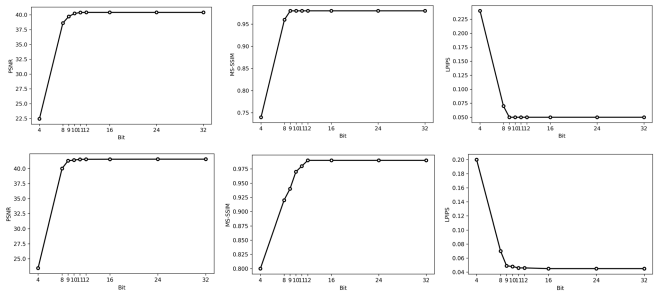


(a) w/o mask loss (b) w/ mask loss (c) GT

Fig. 3. Compressed IR images using different loss functions. Top and bottom: five jets and Tangaroa.

3.1. Network Architecture

As shown in Figure 1, our NeRVI network consists of three main components, including input embedding, SIREN-based residual blocks (SIREN RBs), and upscale blocks.



(a) PSNR vs. bit (b) MS-SSIM vs. bit (c) LPIPS vs. bit

Fig. 4. Model quantization using IR images with different bit lengths for representing parameter values. Top and bottom: vortex and ionization.

Input embedding. Deep neural networks can be utilized as universal function approximators [40]. However, directly training the network with input parameters $(t;\theta;\phi)$ results in poor performance, also observed in [24, 1]. Following [41, 24, 1], we use *positional encoding* to map the input parameters to a higher dimensional space using high-frequency functions, which enables better fitting of data that have high-frequency variation. Formally, the encoding function is defined as

$$E(x) = \left(\sin(c^0\pi x), \cos(c^0\pi x), \cdots, \sin(c^{L-1}\pi x), \cos(c^{L-1}\pi x)\right), \tag{1}$$

where x is an input parameter, c and L are constants. In our experiments, we set $c = 2$ and $L = 4$, following the suggestion of Mildenhall et al. [24]. $E(.)$ is applied separately to the parameters of time step t (normalized to $[-1, 1]$) and each of the two viewing parameters θ and ϕ (normalized to $[-1, 1]$). The outputs of positional encoding are concatenated and fed to the following network.

SIREN RB. In different from NeRV [1], which uses ReLU [42] or GELU [43] as the activation function of the fully-connected (FC) layer, we use SIREN [28] as the building block of MLP. It is an FC layer followed by $\sin(\omega x)$ as the activation function, where ω is a hyper-parameter, set as 30 in our experiments, following Sitzmann et al. [28]. SIREN offers several advantages compared with the FC layer with ReLU. First, it stabilizes the training process. Second, it speeds up network convergence because the gradient of sinusoidal activation exists almost everywhere, while others will be close to zero in some regions. Third, it fits complex signals better in both data and gradient spaces. Our task is more challenging than video compression due to more parameter input and dramatically different temporal patterns. Therefore, we build SIREN RB (shown in Figure 1 (b)) to increase network depth for performance improvement and introduce skip connections to SIREN RB to alleviate the issue of vanishing gradients. An example is shown in Figure 2.

Upscale block. We stack the convolution layer, pixel shuffle layer [44], and GELU activation layer [43] to form the upscale block [1]. By stacking multiple upscale blocks, we upscale the feature map to the original size of the image.

Architecture details. The architecture of NeRVI is shown in Figure 1. It accepts parameters $(t;\theta;\phi)$ as input and outputs the whole rendering image and the corresponding mask image. The architecture follows a hybrid of MLP encoder and CNN

decoder structure. Given the input parameters, we first apply positional encoding to embed the input parameters in a higher-dimensional space. Then two SIREN RBs with skip connection follow the embedding layer's output to learn the input parameters' representations. After that, several upscale blocks are stacked to upscale the features to the original image dimensions. Finally, a header layer (i.e., convolution layer) is leveraged to output the whole rendering image and the corresponding mask image. Note that NeRVI can handle visualization images with an arbitrary size, and the number of upscale blocks depends on the size of the represented images. Table 1 gives the architecture details of the model that compress visualization images of $1,024 \times 1,024$ resolution.

Table 1. The NeRVI architecture parameter details for visualization images of $1,024 \times 1,024$ resolution. We set the number of initial channels c_1 and c_2 for MLP and CNN as 128 and $1,024$, respectively.

layer	type	upscale factor	output size ($C \times H \times W$)
0	positional encoding	—	$12 \times 1 \times 1$
1	SIREN RB	—	$c_1 \times 1 \times 1$
2	SIREN RB & reshape	—	$4c_1 \times 2 \times 2$
3	upscale block	×4	$c_2 \times 8 \times 8$
4	upscale block	×4	$c_2/2 \times 32 \times 32$
5	upscale block	×2	$c_2/4 \times 64 \times 64$
6	upscale block	×2	$c_2/8 \times 128 \times 128$
7	upscale block	×2	$c_2/16 \times 256 \times 256$
8	upscale block	×2	$c_2/32 \times 512 \times 512$
9	upscale block	×2	$c_2/64 \times 1,024 \times 1,024$
10	header layer	—	$4 \times 1,024 \times 1,024$

Loss function. We consider the $L1$ and SSIM [45] losses in the loss function design. To address the foreground-background issue, we propose several key designs: (1) only calculating the loss over foreground pixels between the predicted image and the GT image; (2) designing the network to predict not only the RGB image but also the corresponding mask image; (3) introducing a new mask loss, which measures the difference between the predicted and GT masks. There are two clear advantages to these proposed key designs. First, the model can capture finer details because we only calculate the loss over foreground pixels. This treatment allows us to delineate salient patterns. Second, capturing small objects (e.g., small isosurface components) is a common challenge for neural representation. We address this by using mask images and computing the mask loss. Figure 3 gives such a comparison result. The final loss function is defined as

$$L = \frac{1}{N}\sum_{i=1}^{N} \alpha \left(\|M_i * (f(t;\theta;\phi) - I_i)\|_1 + \|M(t;\theta;\phi) - M_i\|_1\right) + (1-\alpha)\left(1 - \mathrm{SSIM}(f(t;\theta;\phi), I_i)\right), \tag{2}$$

where N is the total number of visualization images; $f(t;\theta;\phi)$ and $M(t;\theta;\phi)$ are, respectively, the predicted visualization image and mask image; I_i and M_i are, respectively, the GT visualization image and mask image; and α is a hyper-parameter that balances the weight for the loss terms. In this paper, we set $\alpha = 0.7$, as suggested in NeRV [1]. The three terms in Equation 2 are rendering image loss and mask image loss in L1 form and rendering image loss in SSIM form.

Inference. The inference result (i.e., inferred image) is defined as $M(t;\theta;\phi) \times f(t;\theta;\phi) + (1 - M(t;\theta;\phi)) \times \mathbf{I}$, where $\mathbf{I}$ is the identity matrix of size $3 \times 1,024 \times 1,024$.

3.2. Model Quantization

Thus far, the trained neural network becomes the images' compressed representation. Further compression can be achieved through model quantization. Unlike other recent works [46, 47, 48, 49] that apply quantization during training, NeRVI utilizes model quantization after the training. The main idea is to map each model parameter to a bit length value. Formally, given a parameter tensor μ of the model, we have

$$\mu_i = \mathrm{round}\left(\frac{\mu_i - \mu_{\min}}{2^{\mathrm{bit}}}\right) \times \frac{\mu_{\max} - \mu_{\min}}{2^{\mathrm{bit}}} + \mu_{\min}, \tag{3}$$

where $\mu_{\max}$ and $\mu_{\min}$ are the maximum and minimum values of parameter tensor μ, "bit" is the bit length for the quantized model, and "round" is the rounding value to the closest integer. A key question is: what should the number of bits be to achieve a good tradeoff between image quality and compression rate? In Figure 4, we plot PSNR, MS-SSIM, and LPIPS curves for different bit number settings using IR images of the vortex and ionization data sets. We observe a good tradeoff when the bit number is 9. Thus, we use this setting in our experiments.

3.3. Weight Encoding

We use entropy encoding to further compress the model [1]. In particular, Huffman coding [50] is applied to the quantized model for representing the data with a more efficient codec because it is lossless, which ensures a further decent compression without any impact on the reconstruction quality.

Table 2. The resolution and total sampled images of each data set.

data set	resolution ($X \times Y \times Z \times T$)	sampled T	# images
vortex [51]	$128 \times 128 \times 128 \times 90$	30	24,000
ionization [52]	$992 \times 992 \times 992 \times 100$	50	40,000
tornado [53]	$128 \times 128 \times 128 \times 48$	48	38,400
five jets	$128 \times 128 \times 128 \times 500$	1	800
Tangaroa [54]	$300 \times 180 \times 120 \times 150$	1	800

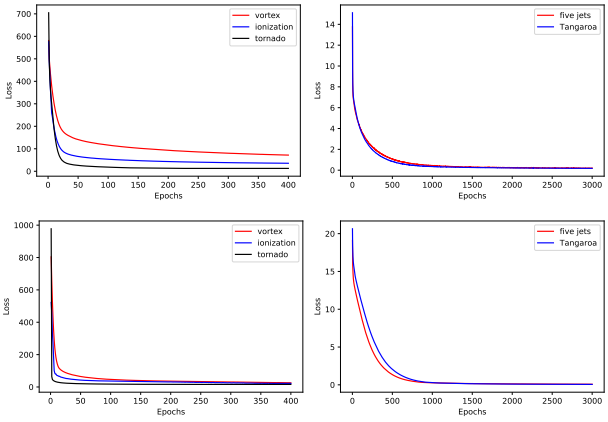


Fig. 5. Loss curves on IR images (top) and DVR images (bottom) using different data sets.

Table 3. Average PSNR (dB), MS-SSIM, and LPIPS values, total encoding/training time (ET, hours) and decoding/inference time (DT, minutes), and compression rate (CR). The best quality performances are highlighted in bold (same for the tables in the appendix).

		IR images ($1,024 \times 1,024$ resolution)						DVR images ($1,024 \times 1,024$ resolution)					
data set	method	PSNR ↑	MS-SSIM ↑	LPIPS ↓	ET	DT	CR	PSNR ↑	MS-SSIM ↑	LPIPS ↓	ET	DT	CR
vortex	InSituNet	37.479	0.957	0.082	210	106	698.89	34.327	0.944	0.092	230	130	742.57
(24,000 images)	NeRV	38.457	0.966	0.075	192	98	716.84	33.929	0.930	0.118	219	119	774.19
	NeRVI	**40.423**	**0.984**	**0.055**	264	133	705.88	**35.503**	**0.979**	**0.044**	304	162	750.00
ionization	InSituNet	38.931	0.964	0.072	258	175	1,131.43	37.130	0.976	0.087	287	206	1,250.53
(40,000 images)	NeRV	38.554	0.971	0.082	235	160	1,142.86	37.381	0.981	0.074	261	193	1,231.27
	NeRVI	**41.573**	**0.990**	**0.045**	323	220	1,142.86	**38.805**	**0.988**	**0.045**	342	234	1,263.16
tornado	InSituNet	46.314	0.998	0.008	166	170	1,080.11	41.118	0.990	0.038	179	174	1,188.12
(38,400 images)	NeRV	45.781	0.997	0.009	156	156	1,081.08	42.886	0.992	0.030	163	165	1,230.77
	NeRVI	**49.238**	**0.999**	**0.007**	213	210	1,090.91	**45.663**	**0.996**	**0.017**	230	229	1,200.00

Table 4. Average PSNR (dB), MS-SSIM, and LPIPS values, total ET (hours), and DT (seconds).

		IR images (256×256 resolution)					DVR images (256×256 resolution)				
data set	method	PSNR ↑	MS-SSIM ↑	LPIPS ↓	ET	DT	PSNR ↑	MS-SSIM ↑	LPIPS ↓	ET	DT
five jets	SIREN	36.903	0.978	0.1010	262.7	41	33.573	0.910	0.126	263.4	59
	NeRF	42.041	0.998	0.0097	264.6	58	37.120	0.993	0.033	267.0	66
(800 images)	InSituNet	41.026	0.998	0.0091	6.5	25	36.750	0.994	0.017	7.0	30
	NeRV	44.313	0.994	0.0260	6.0	23	38.308	0.994	0.012	7.2	46
	NeRVI	**50.369**	**0.999**	**0.0004**	7.0	18	**42.725**	**0.999**	**0.001**	7.1	20
Tangaroa	SIREN	35.358	0.967	0.1443	241.7	47	37.055	0.945	0.214	262.8	62
	NeRF	41.334	0.998	0.0044	245.3	51	38.965	0.987	0.033	263.1	64
(800 images)	InSituNet	38.662	0.999	0.0032	6.5	19	38.974	0.994	0.010	7.0	35
	NeRV	42.919	0.999	0.0006	6.0	18	40.754	0.980	0.049	6.0	30
	NeRVI	**49.482**	**0.999**	**0.0003**	6.5	16	**45.888**	**0.999**	**0.003**	7.0	20

4. Results and Discussion

4.1. Data Sets and Network Training

Data sets. We experimented with the data sets shown in Table 2, where T refers to the total number of time steps of the data set. To demonstrate the utility of NeRVI on large data sets, we apply CoordNet [34] to generate spatial super-resolution of the ionization data set with a scale factor of four, upscaling the resolution from 248^3 to 992^3. Except for the tornado data set, we uniformly sampled a subset of time steps to generate the visualization images. For each data set, we sample every 9-degree for θ (whose range lies in [0,180]) and ϕ (whose range lies in [0,360]) and keep the camera's zoom level unchanged. For each time step, we generate 800 visualization images, covering the full 360-degree view of the entire volume. The total number of images produced for NeRVI training is shown in Table 2.

Network training. NeRVI was implemented using PyTorch. A single NVIDIA Tesla V100 graphics card with 32 GB of memory was used for training and inference. We initialized the parameters following Sitzmann et al. [23] and utilized the Adam optimizer [55] to update the parameters ($\beta_1 = 0.9, \beta_2 = 0.999$). The learning rate was set to 2×10^{-5} to train the model. The input parameters are scaled to $[-1, 1]$ to match the value range of sin(.) activation function. We train the model with 400, 300, and 200 epochs for the vortex, ionization, and tornado data sets, respectively, and 3,000 for the five jets and Tangaroa data sets to make the model converged (see Figure 5). All these hyper-parameters are empirically decided based on experiments.

4.2. Baselines and Evaluation Metrics

Baselines. We compare NeRVI with four state-of-the-art deep learning-based methods (additional comparison results with lossy compression methods, i.e., SZ3 and TTHRESH, are presented in the appendix):

- InSituNet [10] uses a convolutional regression model to infer visualization results from simulation and visualization parameters.
- SIREN [23] is a coordinate-based INR that we use to fit images. It inputs coordinates and employs sin as the activation function.
- NeRF [24] uses an MLP that predicts volume density and view-dependent color from 3D coordinates $(x; y; z)$ and 2D viewpoints $(\theta; \phi)$. It then employs an explicit rendering process to generate novel images.
- NeRV [1] is an image-wise INR for video frame compression, and we adopt it for visualization image compression.

Note that our scenario could have tens of thousands of generated visualization images. We compare NeRVI with the deep image synthesis model (InSituNet) and image-wise INR (NeRV), with $1,024 \times 1,024$ image resolution. We modify InSituNet for fitting images with $1,024 \times 1,024$ resolution such that it has a similar number of parameters as NeRVI. We do not compare with GAN-VR [56] (GAN-based volume rendering) as He et al. [10] showed that InSituNet outperforms GAN-VR.

The coordinate-based INR methods (SIREN and NeRF) demand an excessively long training time to fit only a few hundred images. Therefore, we only consider the low-resolution images (i.e., 256×256) generated from a single time step when comparing NeRVI with SIREN and NeRF and do not report the compression rates.

Following [1], we do not choose the 3D neural rendering version of the original NeRF model, as we aim to effectively compress visualization images. Besides, such an explicit rendering

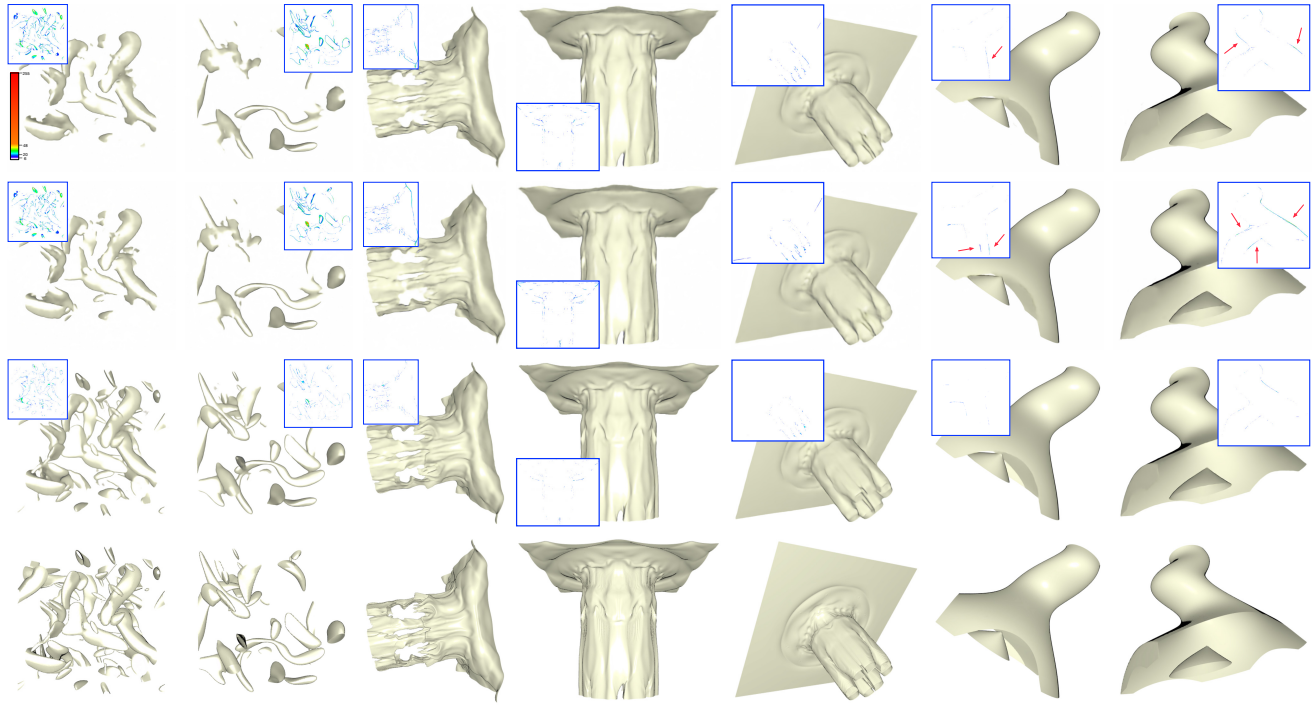


Fig. 6. Comparison of compressed IR images under $1,024 \times 1,024$ image resolution. Left to right: vortex (a and b), ionization (c to e), and tornado (f and g) under different sets of parameters. Top to bottom: InSituNet, NeRV, NeRVI, and GT. The parameter values $(t;\theta;\phi)$ are $(90;90;180)$, $(3;45;81)$, $(100;171;351)$, $(50;81;90)$, $(2;36;54)$, $(23;126;342)$, and $(48;117;243)$, respectively, from (a) to (g).

process may defeat the purpose of our main application of visualization result communication. We might render the volumetric data directly if such a 3D rendering process is entailed. For SIREN and NeRF, we use a five-layer perceptron and contain the same number of parameters as NeRVI by changing the hidden dimension. The batch size is set as $32,000$ coordinates. We scale the values to $[-1, 1]$ for SIREN and $[0, 1]$ for NeRF to match the value range of their respective activation functions.

Evaluation metrics. We evaluate the quality of compressed visualization images with three metrics: peak signal-to-noise ratio (PSNR), multi-scale structural similarity (MS-SSIM) [57], and learned perceptual image patch similarity (LPIPS) [58]. Parameter studies of NeRVI, including initial channels, input image resolution, viewpoint sampling degree, and model pruning, are presented in the appendix.

4.3. Results

Quantitative results. In Table 3, we provide a quantitative comparison among InSituNet, NeRV, and NeRVI with IR and DVR images generated from the vortex, ionization, and tornado data sets. The compression rate ranges from 705 to 1,142 for IR images and from 750 to 1,263 for DVR images, proportional to the number of images compressed. For the same method (NeRV or NeRVI), the variation of compression rate across IR and DVR images is due to weight encoding. Under a similar compression rate, NeRVI yields the best performances on all three data sets with IR and DVR images in terms of PSNR, MS-SSIM, and LPIPS values. This shows the effectiveness of NeRVI in compressing a large collection of visualization images. Regarding encoding and decoding time, NeRV has advantages over InSituNet and NeRVI on IR and DVR images of all data sets, except for InSituNet on the DVR images of the five jets data set. NeRVI falls behind NeRV in encoding and decoding speeds because predicting the mask and computing the mask loss increases the cost for NeRVI.

Table 4 gives a quantitative comparison among SIREN, NeRF, InSituNet, NeRV, and NeRVI with IR and DVR images generated from the five jets and Tangaroa data sets. All five models have the same number of network parameters (101 million). NeRVI outperforms other methods over all three metrics on both data sets, either with IR or DVR images. This demonstrates the advantages of NeRVI compared with the deep image synthesis model (InSituNet) and coordinate-based INRs (SIREN and NeRF). Like NeRV, NeRVI greatly strengthens encoding and decoding speeds over SIREN and NeRF. This is because NeRVI utilizes a hybrid MLP+CNN model to predict an image, while SIREN and NeRF use a simple MLP to output the RGB value of each pixel.

Qualitative results. In Figures 6 and 7, we show compressed rendering images for the vortex, ionization, and tornado data sets under selected $(t;\theta;\phi)$ values. Background borders are cropped for close-up comparison. To illustrate the differences, we compute the pixel-wise difference images (i.e., the Euclidean distance in the CIELUV color space) between the tested method (InSituNet, NeRV, or NeRVI) and GT, shown in blue boxes alongside the compressed images. Noticeable differences are mapped to purple, blue, green, yellow, and red, showing low to high pixel-wise differences (refer to the top-left image of Figures 6 for the colormap legend). Additionally, we place red arrows in some difference images for highlight-

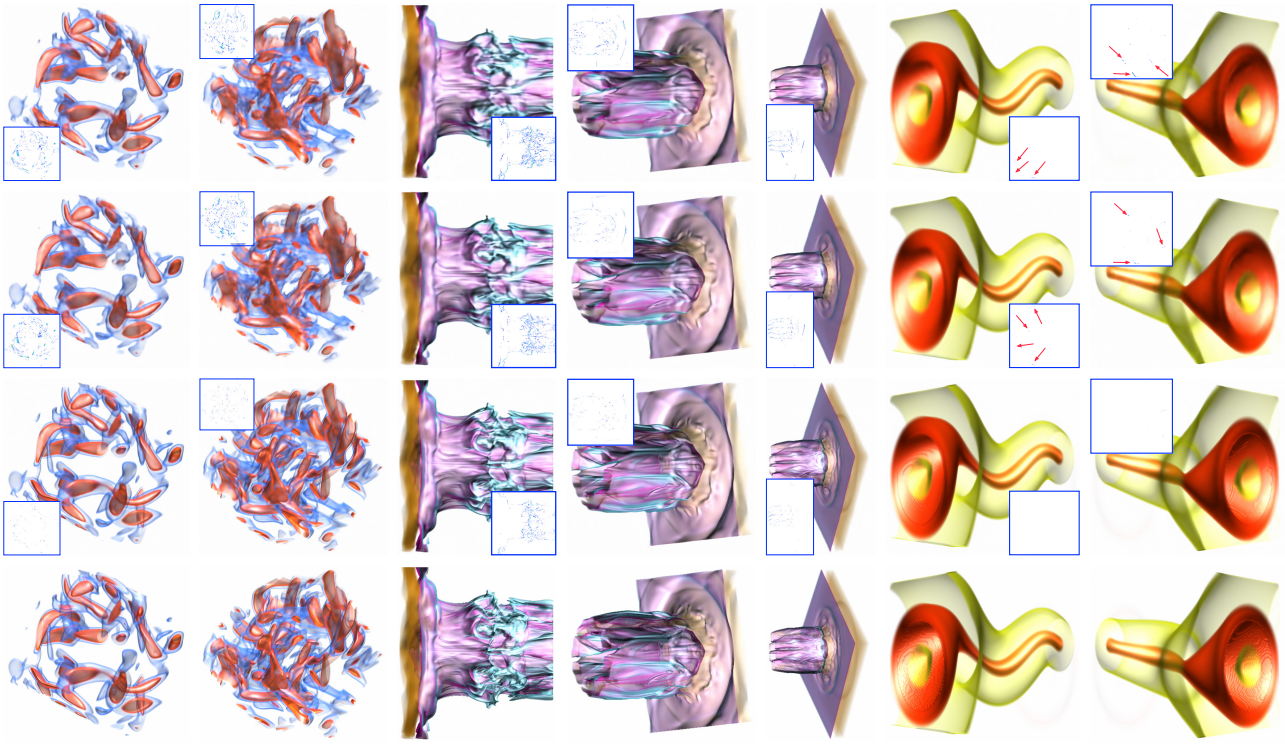


Fig. 7. Comparison of compressed DVR images under $1,024 \times 1,024$ image resolution. Left to right: vortex (a and b), ionization (c to e), and tornado (f and g) under different sets of parameters. Top to bottom: InSituNet, NeRV, NeRVI, and GT. The parameter values $(t; \theta; \phi)$ are $(3; 27; 63)$, $(90; 45; 45)$, $(100; 0; 0)$, $(50; 171; 144)$, $(2; 126; 162)$, $(24; 90; 144)$, and $(48; 0; 36)$, respectively, from (a) to (g).

ing where the variations are hard to see. From the difference images, we can see that NeRVI wins across all cases over InSituNet and NeRV. For IR images, the differences are minor for the tornado data set having one large smooth surface, medium for the ionization data set having one large surface with fine details, and significant for the vortex data set having many separate surface components. For the vortex example, InSituNet- and NeRV-compressed results miss multiple smaller components. NeRVI-compressed results preserve well these small-scale features. This indicates the strength of NeRVI over InSituNet (designed for image synthesis) and NeRV (designed for video compression) when compressing images with separate components or small features. For DVR images, the differences in InSituNet, NeRV, and NeRVI are minor for the tornado data set. NeRVI preserves fine structural details for the ionization data set, while NeRV gives blurry results. This validates the capability of NeRVI to handle data with complex structures (e.g., fine details). The differences become more significant for the vortex data set. NeRVI produces good results on both images with different numbers of feature components. InSituNet and NeRV perform well with few components (see Figure 7 (a)) but lead to rather blurry results with significantly more components (see Figure 7 (b)). This indicates that NeRVI can deal with images containing varying numbers of feature components and addresses the foreground-background issue well.

In Figure 8, we compare compressed rendering images for the five jets and Tangaroa data sets under selected $(t; \theta; \phi)$ values. NeRVI produces the best visual results for both IR and DVR images. From the difference images, we can see that NeRVI generates almost identical results as the GT since the difference images are almost white. NeRV is the second best, with noticeable differences for IR images (five jets) and DVR images (five jets and Tangaroa). SIREN generates the worst results. The compressed images are blurred and miss capturing the object's details. This is because SIREN does not apply the positional encoding layer. NeRF and InSituNet produce acceptable results; however, they miss the small components, and the quality is subpar compared with NeRVI.

Visual interface. We provide a visual interface for users to adjust viewpoint $(\theta; \phi)$ and time step (t) parameters to explore the NeRVI compressed visualization results. A screenshot of our visual interface for exploring the image collection of the ionization data set is illustrated in Figure 9. Users select the data set, rendering mode (IR or DVR), and parameter values to display a certain frame. They can also pick a parameter and animate through the frames. The interface decodes the frames on the fly with a speed of around 3 frames/second. With proper prefetching and caching, it supports interactive exploration, as demonstrated in the accompanying video.

4.4. Hyperparameter Study

We conduct a parameter study to investigate the performance of NeRVI, including initial channels, input image resolution, and viewpoint sampling degree. The appendix also includes a study of model pruning.

Initial channels. For the number of initial channels $(c_1; c_2)$ of MLP and CNN (refer to Table 1 in the paper), we ex-

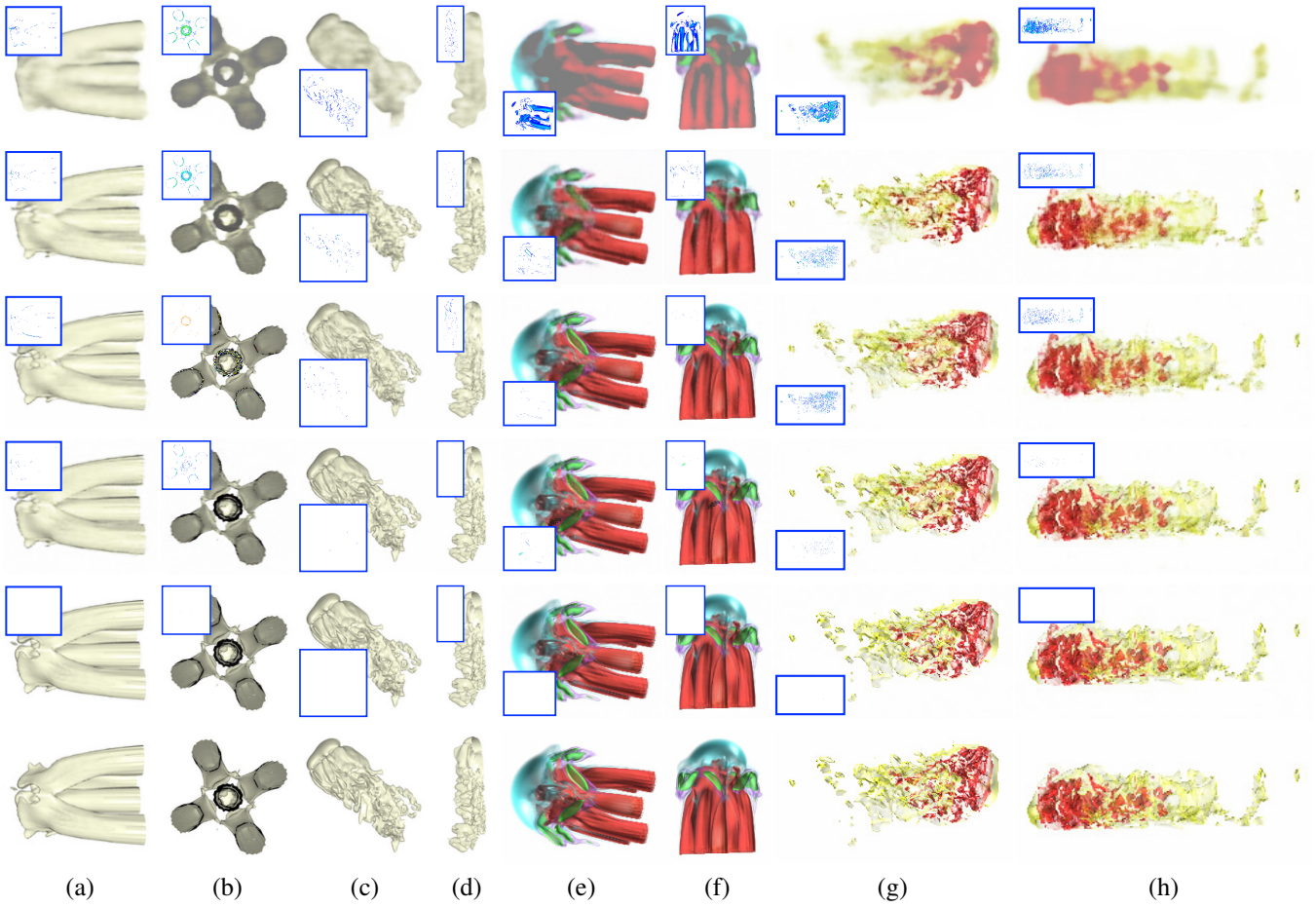


Fig. 8. Comparison of compressed IR images (a to d) and DVR images (e to h) under 256×256 image resolution. Left to right: five jets and Tangaroa under four parameter sets. Top to bottom: SIREN, NeRF, InSituNet, NeRV, NeRVI, and GT. The difference images are displayed on the corners. The parameter values $(t; \theta; \phi)$ are $(1991; 171; 108)$, $(1991; 72; 180)$, $(196; 135; 162)$, $(196; 90; 117)$, $(1991; 171; 252)$, $(1991; 90; 342)$, $(196; 18; 162)$, and $(196; 99; 0)$, respectively, from (a) to (h).

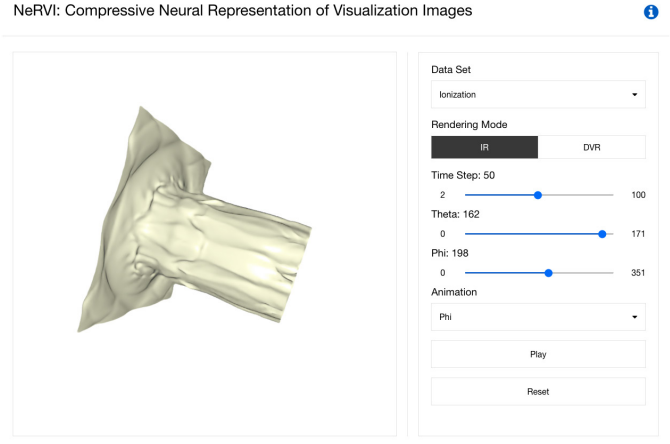
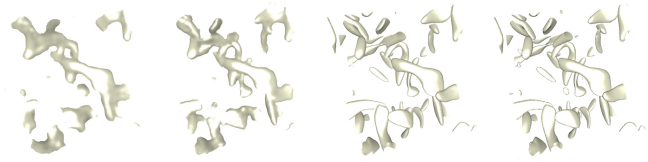


Fig. 9. NeRVI visual interface for exploring the image collection shows the predicted IR image of the ionization data set with $(t; \theta; \phi) = (50; 162; 198)$.

periment with different settings, i.e., $(32; 256)$, $(64; 512)$, and $(128; 1,024)$, on the vortex data set with IR images. Table 5 shows that the model achieves the best performance over all three metrics when setting the number of initial channels $(c_1; c_2)$ as $(128; 1,024)$. Figure 10 validates this as the compressed IR images generated by the model with $(128; 1,024)$ setting are closer to GT than the other settings do. Therefore, we set the number of initial channels $(c_1; c_2)$ of MLP and CNN of the model as $(128; 1,024)$ for all the experiments.

Table 5. Average quantitative metrics values and model parameters (MP, million) using different initial channels $(c_1; c_2)$.

data set	$(c_1; c_2)$	PSNR ↑	MS-SSIM ↑	LPIPS ↓	MP
vortex	$(32; 256)$	37.573	0.960	0.082	6
	$(64; 512)$	38.454	0.966	0.075	25
	$(128; 1,024)$	**40.423**	**0.984**	**0.055**	101



(a) $(32, 256)$ (b) $(64, 512)$ (c) $(128, 1,024)$ (d) GT

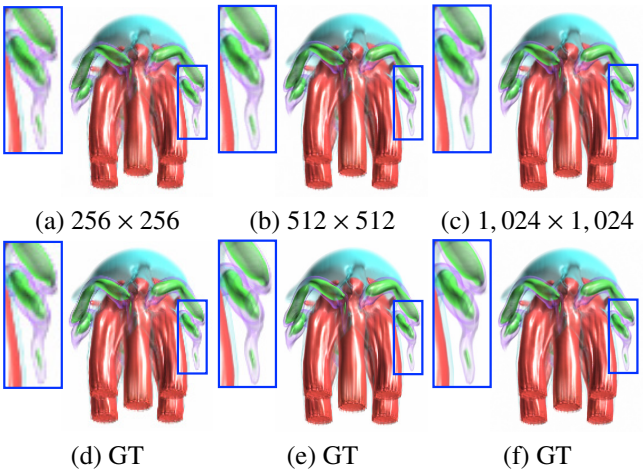
Fig. 10. Compressed IR images of the vortex data set using different numbers of initial channels.

Input image resolution. For the input image resolution, we experiment with different settings (i.e., 256×256, 512×512, and $1,024 \times 1,024$) on the five jets, Tangaroa, and ionization

data sets with DVR images. In Table 6, we show that all three metrics drop when increasing the resolution of the input images. The model performs the best with 256×256 resolution input images. However, in Figures 11, 12, and 13, we show that the compressed rendering images with high resolution are still much close to the GT. Therefore, we use input images with $1,024 \times 1,024$ resolution to achieve high compression rates while maintaining good quantitative and visual results.

Table 6. Average quantitative metrics values using different input image resolutions.

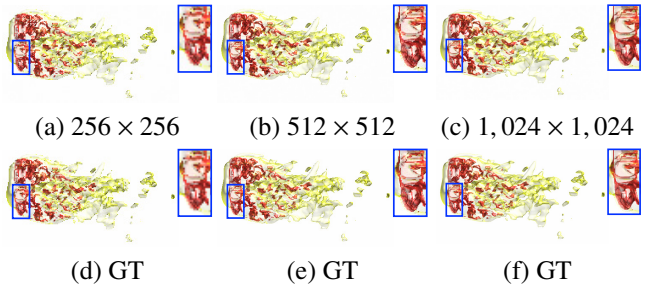
data set	resolution	PSNR ↑	MS-SSIM ↑	LPIPS ↓
five jets	256×256	**42.725**	**0.999**	**0.001**
	512×512	41.842	0.998	0.004
	$1,024 \times 1,024$	40.087	0.997	0.008
Tangaroa	256×256	**45.888**	**0.999**	**0.003**
	512×512	43.627	0.998	0.004
	$1,024 \times 1,024$	41.173	0.997	0.006
ionization	256×256	**40.820**	**0.993**	**0.028**
	512×512	39.541	0.990	0.039
	$1,024 \times 1,024$	38.805	0.988	0.045



(a) 256×256 (b) 512×512 (c) $1,024 \times 1,024$

(d) GT (e) GT (f) GT

Fig. 11. Compressed DVR images of the five jets data set using different input image resolutions.

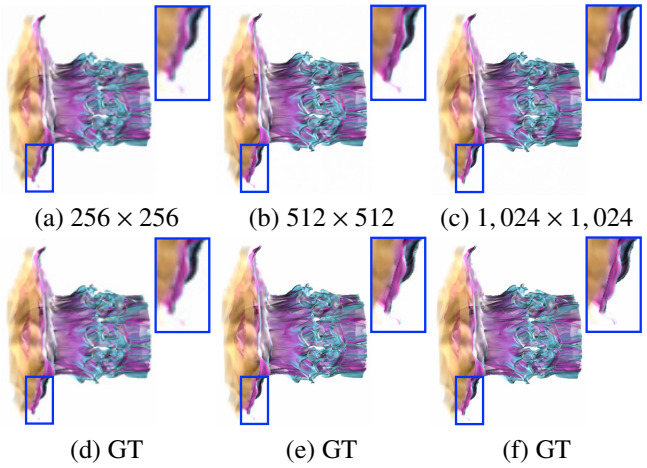


(a) 256×256 (b) 512×512 (c) $1,024 \times 1,024$

(d) GT (e) GT (f) GT

Fig. 12. Compressed DVR images of the Tangaroa data set using different input image resolutions.

Viewpoint sampling degree. We experiment with different settings for the viewpoint sampling degree, i.e., 6, 9, and 10, on the vortex data set with IR images. Table 7 shows that the performance drops when reducing the viewpoint sampling degree (this leads to more images to compress and an increase in the compression rate). In particular, the performance drops significantly when reducing the sampling degree from 9 to 6. Figure 14 shows that sampling with 6-degree produces the worst visual result, followed by 9-degree. Finally, sampling 10-degree leads to the best visual result. Considering the trade-off between quality performance and compression rate, we sample with 9-degree on all the data sets for the experiments.



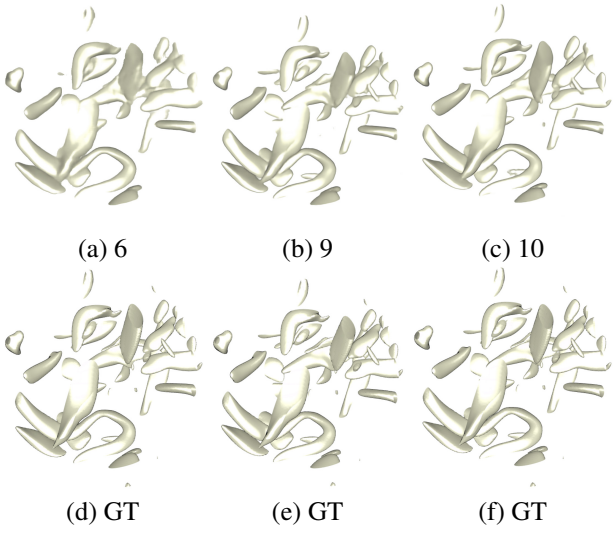
(a) 256×256 (b) 512×512 (c) $1,024 \times 1,024$

(d) GT (e) GT (f) GT

Fig. 13. Compressed DVR images of the ionization data set using different input image resolutions.

Table 7. Average quantitative metrics values and CR using different viewpoint sampling degrees.

data set	degree	PSNR ↑	MS-SSIM ↑	LPIPS ↓	CR
vortex	6	39.021	0.952	0.085	1600.00
	9	40.423	0.984	0.055	705.88
	10	**40.961**	**0.987**	**0.051**	571.43



(a) 6 (b) 9 (c) 10

(d) GT (e) GT (f) GT

Fig. 14. Compressed IR images of the vortex data set using different viewpoint sampling degrees.

4.5. Discussion

Visualization interpolation. As a side product, NeRVI can be utilized for interpolating visualization images. Specifically, we use the trained model to infer novel visualization images given unseen input parameters $(t; \theta; \phi)$. We give examples with IR images of the vortex data set. We select every third time step

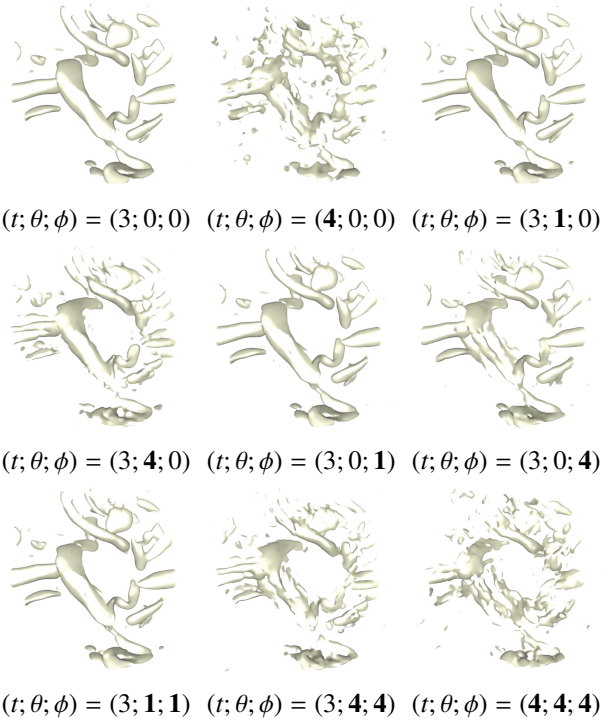


Fig. 15. Inferred novel IR images with parameters $(t;\theta;\phi)$ using the vortex data set. Unseen parameter values are highlighted in bold.

from 90 samples (i.e., $3, 6, \cdots, 90$), and sample every 9-degree for θ and ϕ. In Figure 15, we compare the visual results of unseen input parameters against the result of the seen parameter $(3;0;0)$. We can see that NeRVI can give quite reasonable predictions on the unseen parameters (i.e., $(3;1;0)$, $(3;0;1)$, and $(3;1;1)$) where their visual results have a small difference to the training image (i.e., $(3;0;0)$). The performance gets worse on the unseen parameters (i.e., $(3;4;0)$ and $(3;0;4)$) when the viewing parameter θ or ϕ is somewhere in the middle of the seen samples 0 and 9. The worst cases happen on the unseen parameters $(3;4;4)$, $(4;0;0)$, and $(4;4;4)$. In particular, NeRVI could not infer unseen time steps (either integer or fractional ones, such as 4 or 3.1) in good quality, even though θ and ϕ are seen. These results validate that our trained model is well overfitted. NeRVI essentially fulfills a compression task. The interpolation or inference task needs additional work for quality improvement.

Limitations. While NeRVI outperforms coordinate-based INR methods (SIREN and NeRF), it has the following limitations. First, to achieve a comparable performance and compression rate, NeRVI needs a longer encoding time, albeit faster than coordinate-based INR methods. Nevertheless, this is still acceptable for one-time processing, as our goal is to share and communicate tens of thousands of high-resolution, high-quality visualization images produced from large-scale scientific simulations to peer scientists or the general public. Second, NeRVI has to change the bit values of model quantization to achieve different compression rates of the generated images. The compressed results remain identical when the bit values are set from 12 to 32 (see Figure 4). Thus, it has to train the model with more parameters from scratch to produce images with lower

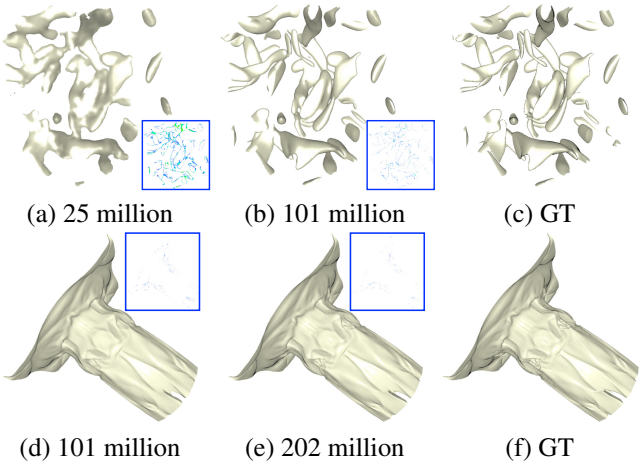


Fig. 16. Comparison of compressed IR images using models with different numbers of parameters. Top: vortex with $(t;\theta;\phi) = (45;9;144)$. Bottom: ionization with $(t;\theta;\phi) = (62;36;0)$. The difference images are displayed on the corners.

compression rates to improve the visual results. As shown in Figure 16, the quality improves significantly when training the model with more parameters (from 25 to 101 million). However, the quality only increases marginally if the model's performance is already good enough (from 101 to 202 million). Third, the isovalue or transfer function parameters are not considered in our model, and we leave this as our future work.

5. Conclusions and Future Work

We have presented, NeRVI, a new deep-learning solution for compressing a large collection of high-resolution, high-quality visualization images. NeRVI includes three stages: image fitting, model compression, and image extraction. Inspired by the image-wise INR work, we redesign the model with SIREN-based residual blocks for performance improvement and predict the mask images to capture fine structural details and small components. Model quantization and weight encoding are utilized for further model compression during post-training. Compared with the state-of-the-art methods, NeRVI produces higher-quality results on multiple data sets while maintaining high compression rates (705 to 1,263).

The future work of NeRVI includes the following. First, besides model quantization and weight encoding in the post-training stage, we would like to explore model quantization during training (e.g., [48]) to achieve an end-to-end framework. Second, we would like to consider more parameters, for example, data parameters (e.g., isovalues) for IR and visual mapping parameters (e.g., color and opacity transfer function) for DVR. This would increase the number of images from tens of thousands to hundreds of thousands or millions, further boosting the compression rate. We would explore using a GPU cluster to train the even larger collection of images. Third, we would like to expedite our method by minimizing redundant parameters in the model, such as those in the last layer of the MLP and the initial NeRV block.

Acknowledgments

This research was supported in part by the U.S. National Science Foundation through grants CNS-1629914, DUE-1833129, IIS-1955395, IIS-2101696, and OAC-2104158, and the U.S. Department of Energy through grant DE-SC0023145. The authors thank Ziang Tong for developing the visual interface and the anonymous reviewers for their insightful comments.

References

[1] Chen, H, He, B, Wang, H, Ren, Y, Lim, SN, Shrivastava, A. NeRV: Neural representations for videos. In: Proceedings of Advances in Neural Information Processing Systems. 2021, p. 21557–21568.

[2] Tikhonova, A, Correa, CD, Ma, KL. Explorable images for visualizing volume data. In: Proceedings of IEEE Pacific Visualization Symposium. 2010, p. 177–184.

[3] Tikhonova, A, Correa, CD, Ma, KL. An exploratory technique for coherent visualization of time-varying volume data. Computer Graphics Forum 2010;29(3):783–792.

[4] Tikhonova, A, Correa, CD, Ma, KL. Visualization by proxy: A novel framework for deferred interaction with volume data. IEEE Transactions on Visualization and Computer Graphics 2010;16(6):1551–1559.

[5] Frey, S, Sadlo, F, Ertl, T. Explorable volumetric depth images from raycasting. In: Proceedings of SIBGRAPI Conference on Graphics, Patterns and Images. 2013, p. 123–130.

[6] Fernandes, O, Frey, S, Sadlo, F, Ertl, T. Space-time volumetric depth images for in-situ visualization. In: Proceedings of IEEE Symposium on Large Data Analysis and Visualization. 2014, p. 59–65.

[7] Ahrens, J, Jourdain, S, O'Leary, P, Patchett, J, Rogers, DH, Petersen, M. An image-based approach to extreme scale in situ visualization and analysis. In: Proceedings of ACM/IEEE Supercomputing Conference. 2014, p. 424–434.

[8] O'Leary, P, Ahrens, J, Jourdain, S, Wittenburg, S, Rogers, DH, Petersen, M. Cinema image-based in situ analysis and visualization of MPAS-ocean simulations. Parallel Computing 2016;55:43–48.

[9] Biedert, T, Garth, C. Contour tree depth images for large data visualization. In: Proceedings of ACM/IEEE Supercomputing Conference. 2015, p. 77–86.

[10] He, W, Wang, J, Guo, H, Wang, KC, Shen, HW, Raj, M, et al. InSituNet: Deep image synthesis for parameter space exploration of ensemble simulations. IEEE Transactions on Visualization and Computer Graphics 2020;26(1):23–33.

[11] Jiao, C, Bi, C, Yang, L, Wang, Z, Xia, Z, Ono, K. ESRGAN-based visualization for large-scale volume data. Journal of Visualization 2023;26(3):649–665.

[12] Tewari, A, Thies, J, Mildenhall, B, Srinivasan, PP, Tretschk, E, Wang, Y, et al. Advances in neural rendering. Computer Graphics Forum 2022;41(2):703–735.

[13] Park, JJ, Florence, P, Straub, J, Newcombe, RA, Lovegrove, S. DeepSDF: Learning continuous signed distance functions for shape representation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2019, p. 165–174.

[14] Mescheder, LM, Oechsle, M, Niemeyer, M, Nowozin, S, Geiger, A. Occupancy Networks: Learning 3D reconstruction in function space. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2019, p. 4460–4470.

[15] Chen, Z, Zhang, H. Learning implicit fields for generative shape modeling. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2019, p. 5939–5948.

[16] Atzmon, M, Lipman, Y. SAL: Sign agnostic learning of shapes from raw data. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2020, p. 2562–2571.

[17] Gropp, A, Yariv, L, Haim, N, Atzmon, M, Lipman, Y. Implicit geometric regularization for learning shapes. In: Proceedings of International Conference on Machine Learning. 2020, p. 3789–3799.

[18] Chibane, J, Mir, A, Pons-Moll, G. Neural unsigned distance fields for implicit function learning. In: Proceedings of Advances in Neural Information Processing Systems. 2020,.

[19] Jiang, CM, Sud, A, Makadia, A, Huang, J, Nießner, M, Funkhouser, TA. Local implicit grid representations for 3D scenes. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2020, p. 6000–6009.

[20] Peng, S, Niemeyer, M, Mescheder, LM, Pollefeys, M, Geiger, A. Convolutional occupancy networks. In: Proceedings of European Conference on Computer Vision. 2020, p. 523–540.

[21] Chabra, R, Lenssen, JE, Ilg, E, Schmidt, T, Straub, J, Lovegrove, S, et al. Deep local shapes: Learning local SDF priors for detailed 3D reconstruction. In: Proceedings of European Conference on Computer Vision. 2020, p. 608–625.

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

[23] Sitzmann, V, Martel, JNP, Bergman, AW, Lindell, DB, Wetzstein, G. Implicit neural representations with periodic activation functions. In: Proceedings of Advances in Neural Information Processing Systems. 2020,.

[24] Mildenhall, B, Srinivasan, PP, Tancik, M, Barron, JT, Ramamoorthi, R, Ng, R. NeRF: Representing scenes as neural radiance fields for view synthesis. In: Proceedings of European Conference on Computer Vision. 2020, p. 405–421.

[25] Dellaert, F, Lin, YC. Neural volume rendering: NeRF and beyond. arXiv preprint arXiv:210105204 2021;.

[26] Tewari, A, Fried, O, Thies, J, Sitzmann, V, Lombardi, S, Sunkavalli, K, et al. State of the art on neural rendering. Computer Graphics Forum 2020;39(2):701–727.

[27] Niemeyer, M, Mescheder, LM, Oechsle, M, Geiger, A. Differentiable volumetric rendering: Learning implicit 3D representations without 3D supervision. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2020, p. 3501–3512.

[28] Sitzmann, V, Chan, ER, Tucker, R, Snavely, N, Wetzstein, G. MetaSDF: Meta-learning signed distance functions. In: Proceedings of Advances in Neural Information Processing Systems. 2020, p. 10136–10147.

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

[30] Guo, M, Fathi, A, Wu, J, Funkhouser, TA. Object-centric neural scene rendering. arXiv preprint arXiv:201208503 2020;.

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

[32] Lu, Y, Jiang, K, Levine, JA, Berger, M. Compressive neural representations of volumetric scalar fields. Computer Graphics Forum 2021;40(3):135–146.

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

[34] Han, J, Wang, C. CoordNet: Data generation and visualization generation for time-varying volumes via a coordinate-based neural network. IEEE Transactions on Visualization and Computer Graphics 2022;Accepted.

[35] Wu, Q, Doyle, MJ, Bauer, D, Ma, KL. Instant neural representation for interactive volume rendering. arXiv preprint arXiv:220711620 2022;.

[36] Han, M, Sane, S, Johnson, CR. Exploratory Lagrangian-based particle tracing using deep learning. Journal of Flow Visualization and Image Processing 2022;29(3):73–96.

[37] Sahoo, S, Lu, Y, Berger, M. Neural flow map reconstruction. Computer Graphics Forum 2022;41(3):391–402.

[38] Wu, Q, Insley, JA, Mateevitsi, VA, Rizzi, S, Ma, KL. Distributed volumetric neural representation for in situ visualization and analysis. In: Proceedings of IEEE Symposium on Large Data Analysis and Visualization. 2022, p. 1–2.

[39] Dupont, E, Goliński, A, Alizadeh, M, Teh, YW, Doucet, A. COIN: Compression with implicit neural representations. arXiv preprint arXiv:210303123 2021;.

[40] Hornik, K, Stinchcombe, M, White, H. Multilayer feedforward networks are universal approximators. Neural Networks 1989;2(5):359–366.

[41] Vaswani, A, Shazeer, N, Parmar, N, Uszkoreit, J, Jones, L, Gomez, AN, et al. Attention is all you need. In: Proceedings of Advances in

Neural Information Processing Systems. 2017, p. 5998–6008.
[42] Nair, V, Hinton, GE. Rectified linear units improve restricted Boltzmann machines. In: Proceedings of International Conference on Machine Learning. 2010, p. 807–814.
[43] Hendrycks, D, Gimpel, K. Gaussian error linear units (GELUs). arXiv preprint arXiv:160608415 2016;.
[44] Shi, W, Caballero, J, Huszar, F, Totz, J, Aitken, AP, Bishop, R, et al. Real-time single image and video super-resolution using an efficient subpixel convolutional neural network. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2016, p. 1874–1883.
[45] Wang, Z, Bovik, AC, Sheikh, HR, Simoncelli, EP. Image quality assessment: From error visibility to structural similarity. IEEE Transactions on Image Processing 2004;13(4):600–612.
[46] Jacob, B, Kligys, S, Chen, B, Zhu, M, Tang, M, Howard, A, et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2018, p. 2704–2713.
[47] Banner, R, Hubara, I, Hoffer, E, Soudry, D. Scalable methods for 8-bit training of neural networks. In: Proceedings of Advances in Neural Information Processing Systems. 2018, p. 5151–5159.
[48] Faghri, F, Tabrizian, I, Markov, I, Alistarh, D, Roy, DM, Ramezani-Kebrya, A. Adaptive gradient quantization for data-parallel SGD. In: Proceedings of Advances in Neural Information Processing Systems. 2020, p. 3174–3185.
[49] Wang, N, Choi, J, Brand, D, Chen, CY, Gopalakrishnan, K. Training deep neural networks with 8-bit floating point numbers. In: Proceedings of Advances in Neural Information Processing Systems. 2018, p. 7686–7695.
[50] Huffman, DA. A method for the construction of minimum-redundancy codes. Proceedings of the IRE 1952;40(9):1098–1101.
[51] Silver, D, Wang, X. Tracking and visualizing turbulent 3D features. IEEE Transactions on Visualization and Computer Graphics 1997;3(2):129–141.
[52] Whalen, D, Norman, ML. Ionization front instabilities in primordial h ii regions. The Astrophysical Journal 2008;673(2):664–675.
[53] Crawfis, RA, Max, N. Texture splats for 3D scalar and vector field visualization. In: Proceedings of IEEE Visualization Conference. 1993, p. 261–266.
[54] Popinet, S, Smith, M, Stevens, C. Experimental and numerical study of the turbulence characteristics of airflow around a research vessel. Journal of Atmospheric and Oceanic Technology 2004;21(10):1575–1589.
[55] Kingma, DP, Ba, J. Adam: A method for stochastic optimization. In: Proceedings of International Conference for Learning Representations. 2015,.
[56] Berger, M, Li, J, Levine, JA. A generative model for volume rendering. IEEE Transactions on Visualization and Computer Graphics 2019;25(4):1636–1650.
[57] Wang, Z, Simoncelli, EP, Bovik, AC. Multiscale structural similarity for image quality assessment. In: Proceedings of Asilomar Conference on Signals, Systems Computers. 2003, p. 1398–1402.
[58] Zhang, R, Isola, P, Efros, AA, Shechtman, E, Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2018, p. 586–595.

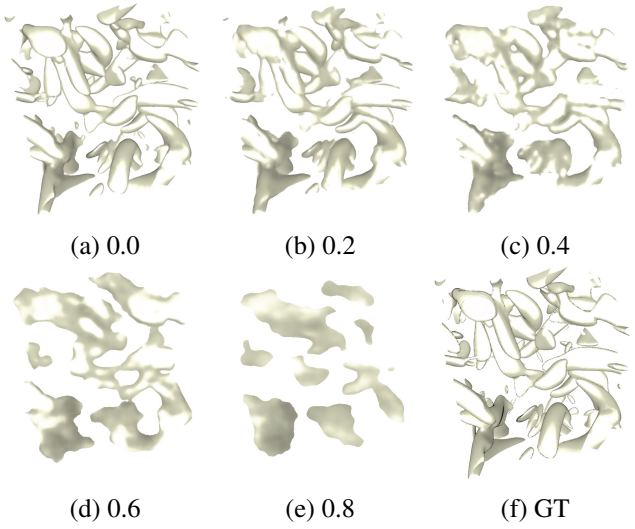
Appendix

In addition to the main results presented in the paper, we conduct experiments to explore the possibility of model pruning. We also compare NeRVI with two lossy compression methods: SZ3 and TTHRESH.

Model pruning. For model pruning, we experiment with different pruning ratio settings (i.e., 0.2, 0.4, 0.6, and 0.8) for global unstructured pruning on the vortex data set with IR images. As shown in Table 1, when increasing the pruning ratio, the performance drops consistently, and the compressed IR images become much worse (refer to Figure 1). Even with a smaller pruning ratio (i.e., 0.2), the compressed IR image fails to capture some large isosurface components' structures and misses smaller ones completely. Therefore, we do not leverage model pruning to compress the model further.

Table 1. Average quantitative metrics values using different pruning ratios.

data set	ratio	PSNR ↑	MS-SSIM ↑	LPIPS ↓
vortex	0.0	**40.423**	**0.984**	**0.055**
	0.2	38.226	0.952	0.138
	0.4	36.707	0.921	0.207
	0.6	34.629	0.909	0.279
	0.8	32.389	0.894	0.346



(a) 0.0 (b) 0.2 (c) 0.4

(d) 0.6 (e) 0.8 (f) GT

Fig. 1. Compressed IR images of the vortex data set using different pruning ratios for global unstructured pruning.

Comparison with lossy compression methods. Besides comparing NeRVI with the deep learning-based methods (InSituNet and NeRV), we also compare it with two lossy compression methods:

- SZ3 [1] is an error-bounded lossy compression method for scientific data reduction.
- TTHRESH [2] is a tensor decomposition method for the lossy compression of scientific data.

We consider two scenarios: (1) SZ3 and TTHRESH directly compress the volumetric data at a similar compression rate as NeRVI, denoted as SZ3-v and TTHRESH-v. We then render the decompressed volumes (for DVR) and the isosurfaces extracted from the decompressed volumes (for IR). (2) SZ3 and TTHRESH are leveraged to compress the volume visualization images (i.e., IR and DVR images) at a similar compression rate as NeRVI, denoted as SZ3-i and TTHRESH-i.

Table 3 provides quantitative results of SZ3-v, TTHRESH-v, SZ3-i, and TTHRESH-i on IR and DVR images generated from the vortex, ionization, and tornado data sets. Under a similar compression rate, NeRVI (refer to Table 3 in the paper) outperforms SZ3-v, TTHRESH-v, SZ3-i, and TTHRESH-i on all three data sets across the evaluation metrics, except for TTHRESH-v on the IR images of the ionization data set. Regarding encoding and decoding time, SZ3-i and TTHRESH-i have great strength in encoding speed and a slight advantage in decoding speed over NeRVI. Since SZ3-v and TTHRESH-v only compress a small number of volumetric data and involve the subsequent IR and DVR process, out of fairness, we do not report their encoding and decoding times.

Figures 2 and 3 show rendering of compressed volumes (SZ3-v and TTHRESH-v) and compressed rendering images (SZ3-i and TTHRESH-i) for the vortex, ionization, and tornado data sets under selected $(t;\theta;\phi)$ values. For IR and DVR images, SZ3-v and TTHRESH-v results on the ionization data set are closer to GT, while those on the vortex and tornado data sets show blocky artifacts. Across all cases, SZ3-i and TTHRESH-i results show noise and color shift.

Table 2. Comparison of different methods over encoding speed (ES), decoding speed (DS), compression rate (CR), and image quality (IQ).

method	ES	DS	CR	IQ
SZ3	fast	medium	high	low/medium
TTHRESH	fast	medium	high	low/medium
InSituNet	medium	medium	medium	medium
SIREN	very slow	very slow	low	low/medium
NeRF	very slow	very slow	low	low/medium
NeRV	medium	medium	medium	medium
NeRVI	medium	medium	medium	high

Comparison. Table 2 compares all seven methods across four metrics. SZ3 and TTHRESH have an excellent advantage in encoding speed; however, the quality of the compressed images cannot be guaranteed, especially when the compression rate is high. Due to the rather slow encoding and decoding speed, coordinate-based INR methods (SIREN and NeRF) are only suitable for compressing a few hundred low-resolution images. InSituNet, NeRV, and NeRVI are in the middle concerning encoding and decoding speeds; however, they can compress tens of thousands of high-resolution images while maintaining good quality. InSituNet performs similarly to NeRV when handling tens of thousands of images (vortex, ionization, and tornado) and falls behind with only hundreds of images (five jets and Tangaroa). Compared with NeRV, our NeRVI is more capable of compressing a large collection of high-resolution images due to the added SIREN-based residual block and mask loss.

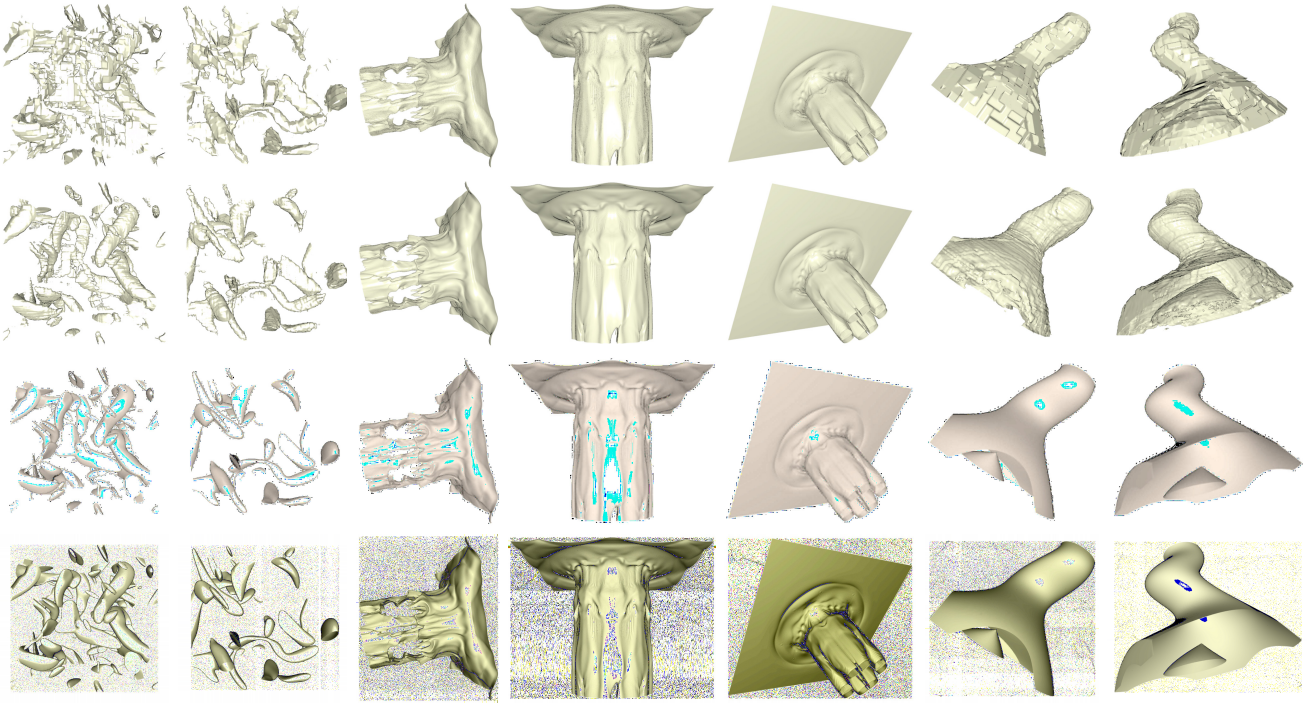
References

[1] Liang, X, Zhao, K, Di, S, Li, S, Underwood, R, Gok, AM, et al. SZ3: A modular framework for composing prediction-based error-bounded lossy compressors. IEEE Transactions on Big Data 2023;9(2):485–498.

[2] Ballester-Ripoll, R, Lindstrom, P, Pajarola, R. TTHRESH: Tensor compression for multidimensional visual data. IEEE Transactions on Visualization and Computer Graphics 2020;26(9):2891–2903.

Table 3. Average PSNR (dB), MS-SSIM, and LPIPS values, total ET (hours) and ET (minutes), and CR. Refer to Table 3 in the paper for results using InSituNet, NeRV, and NeRVI.

		IR images ($1,024 \times 1,024$ resolution)						DVR images ($1,024 \times 1,024$ resolution)					
data set	method	PSNR ↑	MS-SSIM ↑	LPIPS ↓	ET	DT	CR	PSNR ↑	MS-SSIM ↑	LPIPS ↓	ET	DT	CR
vortex ($24,000$ images)	SZ3-v	33.386	0.605	0.310	—	—	705.98	32.644	0.774	0.203	—	—	737.61
	TTHRESH-v	34.077	0.662	0.268	—	—	710.30	**33.957**	0.890	0.139	—	—	746.19
	SZ3-i	36.098	0.942	0.145	4.64	127	715.41	33.489	0.921	0.142	5.24	153	725.35
	TTHRESH-i	**36.123**	**0.969**	**0.070**	5.51	120	736.78	33.862	**0.931**	**0.094**	10.28	156	779.16
ionization ($40,000$ images)	SZ3-v	40.144	0.988	0.066	—	—	1131.49	35.726	0.930	0.089	—	—	1288.26
	TTHRESH-v	**44.343**	**0.998**	**0.009**	—	—	1192.71	**36.780**	0.947	**0.077**	—	—	1248.00
	SZ3-i	35.825	0.940	0.123	7.54	188	1,146.38	35.204	0.965	0.143	7.81	247	1,248.82
	TTHRESH-i	36.324	0.952	0.096	8.41	192	1,096.43	35.123	**0.966**	0.134	9.89	224	1,229.49
tornado ($38,400$ images)	SZ3-v	35.162	0.838	0.169	—	—	1023.19	34.642	0.766	0.231	—	—	1200.77
	TTHRESH-v	36.474	0.864	0.151	—	—	1055.73	35.657	0.835	0.170	—	—	1204.91
	SZ3-i	38.113	0.959	0.126	6.99	178	1048.47	38.155	0.933	0.172	7.37	183	1212.61
	TTHRESH-i	**44.457**	**0.961**	**0.114**	7.11	196	1,064.35	**41.437**	**0.990**	**0.043**	12.49	209	1,157.26



(a) (b) (c) (d) (e) (f) (g)

Fig. 2. Comparison of compressed IR images under $1,024 \times 1,024$ image resolution. Left to right: vortex (a and b), ionization (c to e), and tornado (f and g) under different sets of parameters. Top to bottom: SZ3-v, TTHRESH-v, SZ3-i, and TTHRESH-i. The parameter values $(t; \theta; \phi)$ are $(90; 90; 180)$, $(3; 45; 81)$, $(100; 171; 351)$, $(50; 81; 90)$, $(2; 36; 54)$, $(23; 126; 342)$, and $(48; 117; 243)$, respectively, from (a) to (g). Refer to Figure 6 in the paper for results using InSituNet, NeRV, and NeRVI.

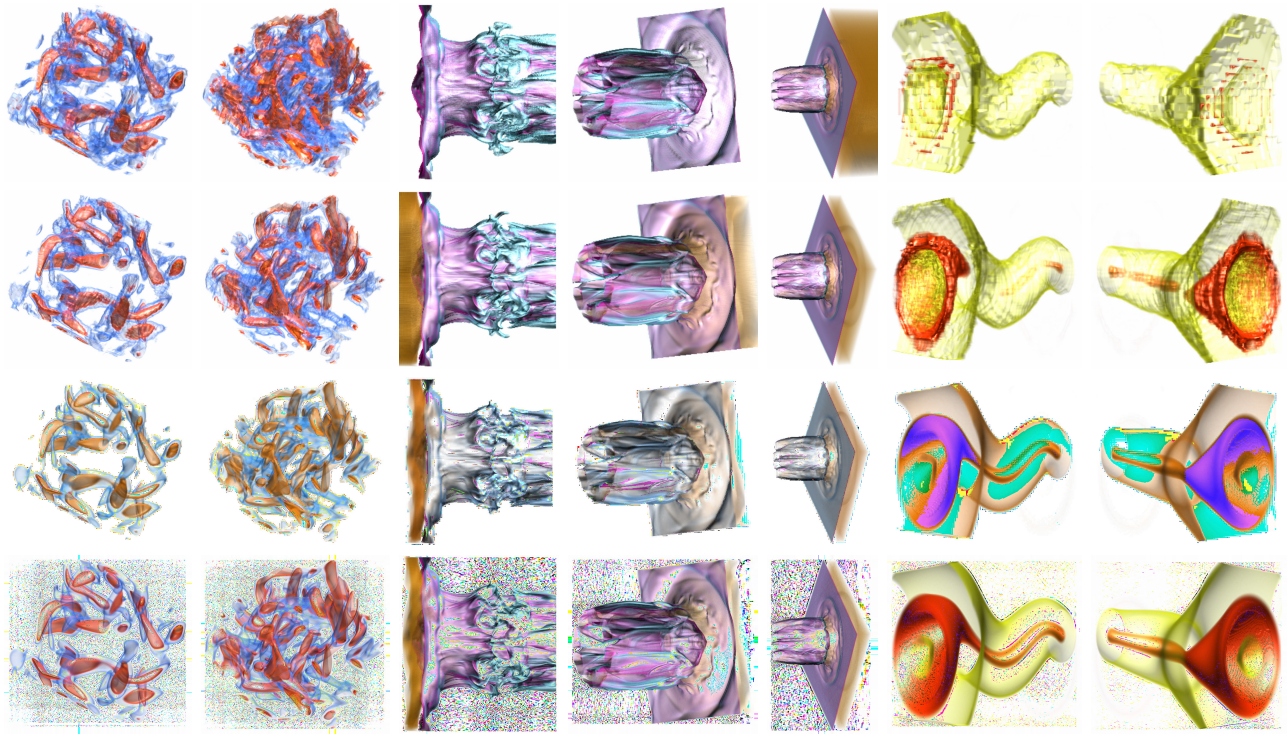


Fig. 3. Comparison of compressed DVR images under $1,024 \times 1,024$ image resolution. Left to right: vortex (a and b), ionization (c to e), and tornado (f and g) under different sets of parameters. Top to bottom: SZ3-v, TTHRESH-v, SZ3-i, and TTHRESH-i. The parameter values $(t; \theta; \phi)$ are $(3; 27; 63)$, $(90; 45; 45)$, $(100; 0; 0)$, $(50; 171; 144)$, $(2; 126; 162)$, $(24; 90; 144)$, and $(48; 0; 36)$, respectively, from (a) to (g). Refer to Figure 7 in the paper for results using InSituNet, NeRV, and NeRVI.