Computers & Graphics (2022)

Contents lists available at ScienceDirect

Computers & Graphics

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

TSR-VFD: Generating Temporal Super-Resolution for Unsteady Vector Field Data

Jun Han and Chaoli Wang

Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, USA

ARTICLE INFO

Article history:
Received February 8, 2022

Keywords: Temporal super-resolution, unsteady vector field, data reconstruction, deep learning

ABSTRACT

We present TSR-VFD, a novel deep learning solution that recovers temporal super-resolution (TSR) of three-dimensional vector field data (VFD) for unsteady flow. In scientific visualization, TSR-VFD is the first work that leverages deep neural nets to interpolate intermediate vector fields from temporally sparsely sampled unsteady vector fields. The core of TSR-VFD lies in using two networks: InterpolationNet and MaskNet, that process the vector components of different scales from sampled vector fields as input and jointly output synthesized intermediate vector fields. To demonstrate our approach's effectiveness, we report qualitative and quantitative results with several data sets and compare TSR-VFD against vector field interpolation using linear interpolation (LERP), generative adversarial network (GAN), and recurrent neural network (RNN). In addition, we compare TSR-VFD with a lossy compression (LC) scheme. Finally, we conduct a comprehensive study to evaluate critical parameter settings and network designs.

1. Introduction

Over the past three decades, flow visualization has been a core research area and remains a vibrant one in scientific visualization. Among many methods developed, the integration-based method stands out as the most popular form for visualizing three-dimensional vector field data (VFD). For this method, we place particles or seeds in the domain to trace flow lines (*streamlines* for *steady* flow and *pathlines* for *unsteady* flow) from the underlying vector fields for visual representation, rendering, and understanding. In this paper, we place our focus on unsteady vector fields as they are the most general output from computational fluid dynamics simulations.

For large-scale scientific simulations, scientists usually could only afford to store a fraction of the simulation data in the reduced form. As high-performance computing systems are often constrained with respect to data movement and storage, deciding what data are the most essential to store for post hoc analysis becomes a prevalent task for many scientists running their simulations. As data reduction becomes inevitable, data reconstruction or restoration provides a necessary means to recover data resolutions and details from the reduced data. We consider a typical scenario for unsteady vector fields where scientists store VFD sparsely (e.g., every tenth time step) during the simulation. Our goal is to restore unsaved intermediate time steps as accurately as possible during postprocessing.

Interpolating temporally resolved vector fields from sparsely sampled ones poses several key challenges. First, unlike images where each of the *rgb* channels keeps the same value range, VFD could have dramatically different value ranges for each of the *uvw* components (e.g., $u \in [-40, 30]$, $v \in [-0.1, 0.7]$, and $w \in [-10^{-4}, 10^{-3}]$). Directly stacking these components together and feeding them into *convolutional neural networks* (CNNs), *recurrent neural networks* (RNNs), or *generative adversarial networks* (GANs) would not work. This is because it eliminates the small-range component as the large-range component dominates the values in the convolutional operations. Second, the dynamic change of VFD over time is typically *nonlinear*. Conventional approaches employ standard *linear interpolation* (LERP) to generate intermediate vector fields. These interpolations are only based on *local* information around the

interpolated position and, therefore, may not capture the complex, *global* evolution and nonlinear changes of vectors. Third, synthesizing small-scale values in VFD (e.g., at a scale of 10^{-7}) due to high numerical precision is challenging.

To respond, we propose TSR-VFD, a deep learning solution for unsteady VFD reconstruction. TSR-VFD can learn the data nonlinearly in both *spatial* and *temporal* aspects among different vector fields for generating high-quality intermediate vector fields. Besides, to solve the dramatically different value ranges in VFD, we include two designs in TSR-VFD. First, we leverage two networks: InterpolationNet and MaskNet to learn the different scales of the vector data. Second, these two networks include three independent neural nets for training u, v, and w components separately. Once TSR-VFD converges, it can interpolate the missing intermediate vector fields conditioned on the given vector fields at both ends.

To demonstrate the effectiveness of TSR-VFD, we show quantitative and qualitative results with several vector field data sets of different characteristics. We compare our approach against LERP and two solutions based on GAN and RNN, respectively. We show that TSR-VFD achieves better reconstruction quality in terms of *peak signal-to-noise ratio* (PSNR), *average angle difference* (AAD), *root-absolute-error* (RAE), *pathline distance* (PD). It also wins over other methods judged from streamline and pathline visualization, both objectively via *learned perceptual image patch similarity* (LPIPS) [1] and subjectively via side-by-side comparison of integral line rendering and visualization of derived physical quantities and error. In addition, we compare TSR-VFD against a lossy compression (LC) scheme.

The contributions of this paper are as follows. First, to the best of our knowledge, in scientific visualization, our work is the first that applies deep learning solutions for generating missing intermediate time steps for unsteady flow. Second, to ensure high-quality results, we propose two specific designs for unsteady vector data interpolation, which are different from the designs commonly used in image and volume super-resolution tasks. Third, we investigate several parameter settings and network designs to analyze how they would impact the performance of TSR-VFD.

2. Related Work

Vector field reconstruction. Reconstructing VFD has been studied extensively, starting from the 1990s. Mussa-Ivaldi [2] presented a least-square based method to reconstruct 2D vector fields from samples. Gouesbet and Letellier [3] used a multivariate polynomial approximation for global vector field reconstruction of time-continuous dynamical systems. Another algorithm to reconstruct 2D vector fields was given by Lage et al. [4]. They used a sparse set of samples to first fit a polynomial locally to describe each velocity component and then approximated the vector field globally using partition of unity. Lettellier et al. [5] extracted topological features to characterize and reconstruct 3D vector fields for experimental electrochemical systems. Chen et al. [6] applied triangulation to reconstruct velocities from samples along streamlines and then linearly interpolated the vector field's grid points from the triangulation. Xu et al. [7] used Shannon's entropy to compute every voxel's information content in a vector field to guide the streamline seeding process. Xu and Prince [8] introduced *gradient vector flow*, which has been used by Tao et al. [9] for vector field reconstruction from a set of representative streamlines. This two-stage algorithm recovers a vector field from streamlines by first estimating velocities for voxels with streamlines passing and then interpolating the missing voxels by minimizing the Laplacian over the whole vector field. Recently, Han et al. [10] reconstructed steady vector fields from streamlines using deep learning. Our work aims to leverage deep learning techniques to reconstruct missing intermediate vector fields for unsteady flow, which has not been attempted previously.

Deep learning in scientific visualization. With deep learning techniques successfully tackling problems in different research areas, they recently found their applications in scientific visualization. Part of these works has tackled problems related to volume visualization. For example, Zhou et al. [11] applied a CNN for volume upscaling. Berger et al. [12] used a GAN to generate volume rendering images conditioned on viewpoint and transfer function. Cheng et al. [13] presented a deep learning approach to assist feature extraction from complex structures for volume visualization. Hong et al. [14] proposed DNN-VolVis that generates novel volume rendering with a given goal effect and a new viewpoint given an original rendering. He et al. [15] presented InSituNet that synthesizes rendering images for parameter space exploration of ensemble simulations. Weiss et al. [16] designed an image-based deep learning solution that generates high-resolution isosurface rendering images from low-resolution ones. Han et al. [17] utilized a GAN to translate a variable sequence to another one. Han and Wang [18, 19] proposed TSR-TVD and SSR-TVD for generating temporal and spatial super-resolution for time-varying volumetric data. Han et al. [20] also presented STNet, an end-to-end framework that generates spatiotemporal super-resolution for time-varying data. Our work is similar to TSR-TVD. However, TSR-TVD focuses on interpolating missing time steps for multivariate time-varying data sets while our work focuses on unsteady vector fields. The solution for TSR-TVD is not suitable for unsteady vector fields due to the different data types and rendering methods.

For flow visualization, Hong et al. [21] leveraged *long short-term memory* (LSTM) to estimate the access pattern for parallel particle tracing. Xie et al. [22] designed tempoGAN that upscales time-varying flow data in the spatial dimension. Han et al. [10] introduced a two-stage approach to reconstruct steady vector fields from a set of representative streamlines, which was extended by Gu et al. [23] to reconstruct unsteady vector fields. Wiewel et al. [24] proposed an LSTM-based solution to predict the changes of pressure fields for learning the temporal evolution of fluid flows. Kim and Günther [25] extracted steady reference frames from unsteady 2D vector fields through a two-step CNN. Werhahn et al. [26] proposed a generative model that spatially upscales flow data by learning information of the xy-plane then refining along the z axis. Han et al. [27] used deep learning techniques to select and cluster streamlines and sur-

faces by extracting their features in a latent space using autoencoders. Jakob et al. [28] designed a CNN for interpolating flow maps by creating a large 2D fluid flow field as training samples. Guo et al. [29] designed CNNs that generate super-resolution of 3D VFD in the spatial domain with a scaling factor of 4 or 8. Wiewel et al. [30] combined CNN and RNN to predict future time steps of flow data. Sahoo and Berger [31] proposed a deep learning-based spatial super-resolution approach that considers both vector and streamline errors during optimization. Unlike the previous works, which aim to learn representations of flows, reconstruct flow data in the spatial dimension, or predict future time steps, we aim to recover the missing intermediate time steps for unsteady flow data.

CNN for sequence learning. CNN has achieved impressive results in sequence learning tasks. Niklaus et al. [32] introduced a CNN for frame interpolation where the network learns a kernel through the input frames and then applies the learned kernel to generate the missing frames. Nguyen et al. [33] proposed a deep linear embedding model to interpolate the intermediate frames. They transformed each frame into a feature space, then linearly interpolated the intermediate frames in the feature space, and finally recovered the interpolated features to the corresponding frames. Jiang et al. [34] established CNNs to estimate the forward and backward optical flows via two given frames. They then wrapped these optical flows into the frames to generate arbitrary in-between frames. Gehring et al. [35] conducted a solely CNN-based experiment for sequence to sequence learning (e.g., machine translation, abstractive summarization). Yu et al. [36] presented QANet, a CNN and self-attention based solution, that is seven times faster and achieves similar performance compared with RNN in reading comprehension task. Our work differs from these works mentioned above in two ways. First, we leverage CNN for unsteady vector field interpolation in the context of scientific visualization. Second, we take into account the different value ranges in the three vector components for network design.

3. TSR-VFD

3.1. Overview

Let us denote $\mathbf{F}^T = \{\mathbf{F}_1, \cdots, \mathbf{F}_n\}$ as a sequence of n vector fields used for *training*, and $\mathbf{F}^I = \{\mathbf{F}_n, \mathbf{F}_{n+s+1}, \cdots, \mathbf{F}_{n+m(s+1)}\}$ as a sequence of vector fields used for *inference*, where s is the interpolation interval. An example is shown in Figure 1 (a). Each $\mathbf{F}_i$ can be decomposed into three velocity components, $\mathbf{F}_{u,i}$, $\mathbf{F}_{v,i}$, and $\mathbf{F}_{w,i}$. Assume L, H, and W are the three spatial dimensions of the vector fields $\mathbf{F}$. $\theta_{\text{Interpolation}}$ and θ_{Mask} are the learnable parameters of InterpolationNet and MaskNet.

As shown in Figure 1 (b), TSR-VFD accepts two sampled vector fields (i.e., $\mathbf{F}_i$ and $\mathbf{F}_j$) as input. It leverages two modules (i.e., InterpolationNet and MaskNet) to process the different-scale vector components. Each module contains three separate neural nets to interpolate the three vector components individually, and then jointly produces the intermediate vector fields $\hat{\mathbf{F}}$. InterpolationNet serves to estimate the velocity at each voxel for the intermediate vector fields. MaskNet aims

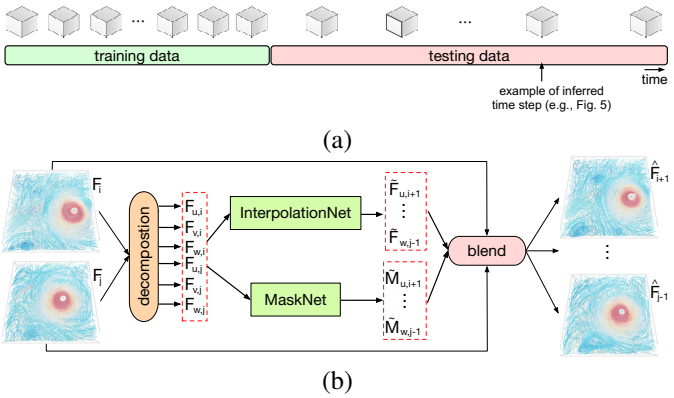


Fig. 1: (a) The training and testing time steps from the vector field sequence. (b) Overview of TSR-VFD. Two sampled vectors are each decomposed into three components. Each component is separately processed by one InterpolationNet and one MaskNet. Finally, the synthesized high-resolution components are concatenated to generate the intermediate vector fields. In this example, $j = i+s+1$.

to score the confidence of each voxel generated by InterpolationNet. The interpolated velocities and corresponding confidence scores will be fed into the blending block to output the intermediate vector fields. In particular, we aim to seek an interpolation function $\mathscr{I}$ so that $\mathscr{I}(\mathbf{F}_i, \mathbf{F}_{i+s+1}) \approx \{\mathbf{F}_{i+1}, \cdots, \mathbf{F}_{i+s}\}$, where $i \in [n, n+s+1, \cdots, n+(m-1)(s+1)]$. That is, given the vector fields at both ends: $\mathbf{F}_i$ and $\mathbf{F}_{i+s+1}$, we interpolate s intermediate vector fields: $\mathbf{F}_{i+1}$ to $\mathbf{F}_{i+s}$. Each network contains three individual nets (i.e., u-net, v-net, and w-net) that process u, v, and w components separately. TSR-VFD accepts $\mathbf{F}_i$ and $\mathbf{F}_{i+s+1}$ as input and decomposes each vector into three components: $\mathbf{F}_{u,i}$, $\mathbf{F}_{v,i}$, $\mathbf{F}_{w,i}$, $\mathbf{F}_{u,i+s+1}$, $\mathbf{F}_{v,i+s+1}$, and $\mathbf{F}_{w,i+s+1}$. These six components are processed by six independent neural nets: $\text{Interpolation}_{u-Net}$, $\text{Interpolation}_{v-Net}$, $\text{Interpolation}_{w-Net}$, Mask_{u-Net}, Mask_{v-Net}, and Mask_{w-Net}, and interpolated vector components are produced

$$\{\hat{\mathbf{F}}_{u,i+1}, \cdots, \hat{\mathbf{F}}_{u,i+s}\} \approx \mathscr{I}_u(\mathbf{F}_{u,i}, \mathbf{F}_{u,i+s+1}), \quad (1)$$

$$\{\hat{\mathbf{F}}_{v,i+1}, \cdots, \hat{\mathbf{F}}_{v,i+s}\} \approx \mathscr{I}_v(\mathbf{F}_{v,i}, \mathbf{F}_{v,i+s+1}), \quad (2)$$

$$\{\hat{\mathbf{F}}_{w,i+1}, \cdots, \hat{\mathbf{F}}_{w,i+s}\} \approx \mathscr{I}_w(\mathbf{F}_{w,i}, \mathbf{F}_{w,i+s+1}). \quad (3)$$

Finally, we concatenate $\{\hat{\mathbf{F}}_{u,i+1}, \cdots, \hat{\mathbf{F}}_{u,i+s}\}$, $\{\hat{\mathbf{F}}_{v,i+1}, \cdots, \hat{\mathbf{F}}_{v,i+s}\}$, and $\{\hat{\mathbf{F}}_{w,i+1}, \cdots, \hat{\mathbf{F}}_{w,i+s}\}$ into $\{\hat{\mathbf{F}}_{i+1}, \cdots, \hat{\mathbf{F}}_{i+s}\}$. To minimize the difference between the interpolated vector field $\hat{\mathbf{F}}$ and ground truth (GT) vector field $\mathbf{F}^T$, we consider L_2 norm to estimate the difference. The computed difference will be propagated to InterpolationNet and MaskNet for searching the globally optimized solutions of $\theta_{\text{Interpolation}}$ and θ_{Mask}.

Straightforwardly, we can stack u, v, and w components together into one InterpolationNet to produce the intermediate vector fields. However, this practice will bring two problems, as sketched in Figure 2. First, considering a local region of one component (e.g., u component) in a vector field, the large values will dominate the small values in convolutional (Conv) operations, as shown in Figure 2 (a). Second, considering stacking three components together, the large components will dominate the small component, as described in Figure 2 (b). To address

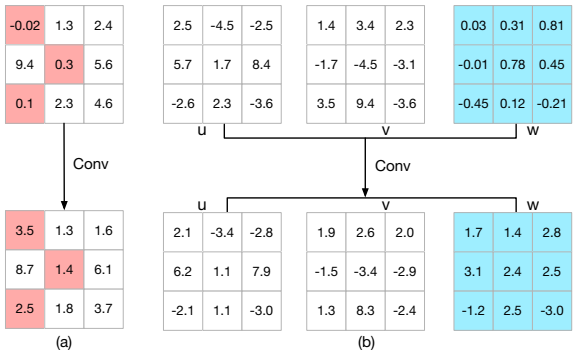


Fig. 2: Data distribution leads to inaccurate predictions using Conv operation. (a) Large values will eliminate small values (highlighted in red) in a local region in one vector component. (b) Large components will eliminate small components (highlighted in blue) if three components are stacked together and fed into one neural net.

these problems, we need both InterpolationNet and MaskNet (addressing the first problem) and train u, v, and w components individually (addressing the second problem).

In the following, we first describe the details of TSR-VFD, including the architecture of TSR-VFD and the definition of the loss function. Then, we provide optimization details for training TSR-VFD.

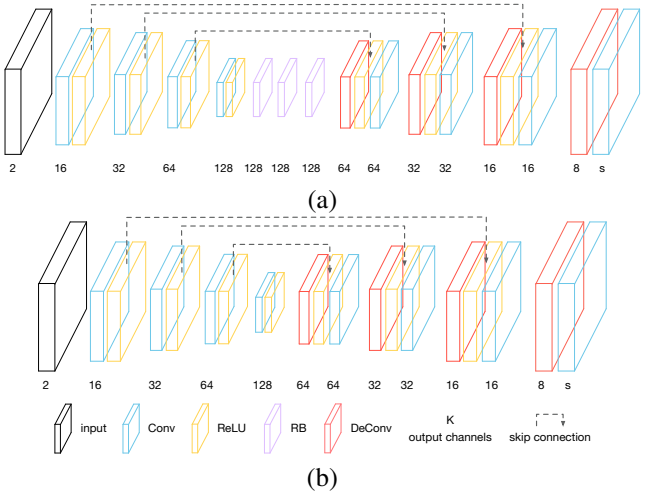


Fig. 3: Network architecture of TSR-VFD. (a) and (b) show one neural net in InterpolationNet and MaskNet, respectively.

3.2. Network Architecture

To guarantee network performance, we leverage two techniques in TSR-VFD: (1) adding *residual blocks* (RBs) [37] to alleviate the gradient vanishing problem (i.e., the gradients are close to zero after several iterations of optimization) and (2) applying *skip connection* [38] to enhance the quality of interpolated vector fields by incorporating temporal information. Residual block and skip connection can increase gradient computation paths, allowing the network to have multiple paths for parameter updates.

The RBs connect the feature maps from earlier layers to later layers so that the gradient can be calculated through multiple paths, which will alleviate the gradient vanishing problem during training. In an RB, the input is convoluted with several Conv layers without changing its dimension in one path. In the other path, one Conv layer is applied to the input. These two outputs are merged by addition. Skip connection is used to increase gradient computation paths to alleviate gradient vanishing. Besides, it incorporates temporal information from the vector field sequence at different generation stages rather than only paying attention to the two input vector fields' deep spatial features during interpolation. That is, with the vector fields at both ends as input, the input vector fields' information will flow into the generation process of the intermediate vector fields at different feature scales by skip connection.

Table 1: Network architecture parameter details for one neural net. "ker", "str", and "out chs" stand for the kernel, stride, and output channels.

InterpolationNet				MaskNet			
type	ker size	str	out chs	type	ker size	str	out chs
input	N/A	N/A	2	input	N/A	N/A	2
Conv+ReLU	4	2	16	Conv+ReLU	4	2	16
Conv+ReLU	4	2	32	Conv+ReLU	4	2	32
Conv+ReLU	4	2	64	Conv+ReLU	4	2	64
Conv+ReLU	4	2	128	Conv+ReLU	4	2	128
3 RBs	3	1	128	DeConv+ReLU	4	2	64
DeConv+ReLU	4	2	64	Conv+ReLU	3	1	64
Conv+ReLU	3	1	64	DeConv+ReLU	4	2	32
DeConv+ReLU	4	2	32	Conv+ReLU	3	1	32
Conv+ReLU	3	1	32	DeConv+ReLU	4	2	16
DeConv+ReLU	4	2	16	Conv+ReLU	3	1	16
Conv+ReLU	3	1	16	DeConv+ReLU	4	2	8
DeConv+ReLU	4	2	8	Conv+sigmoid	3	1	s
Conv	3	1	s				

InterpolationNet. As shown in Figure 3 (a), InterpolationNet has three individual neural nets, and each neural net is composed of an *encoding* path, a *refining* path, and a *decoding* path. There are four Conv layers in the encoding path, three RBs in the refining path, and four composites of deconvolutional (DeConv) and Conv layers in the decoding path. In the encoding path, each of the four Conv layers reduces the input dimension by half. In the refining path, several RBs are utilized for refining and cleaning the feature maps from the encoding path. In the decoding path, DeConv layers are utilized to upscale the feature maps' dimension by two. We also utilize skip connection to fuse the features maps from the encoding and decoding paths. After that, one Conv layer is used to refine the feature maps. Rectified linear unit (ReLU) [39] is applied after each Conv and DeConv. Note that there is no activation function in the last Conv layer because the vector field's data range is unbounded. Refer to Table 1 for parameter details.

MaskNet. The architecture of MaskNet is similar to that of InterpolationNet. The only difference is that there are no RBs in MaskNet since the goal of MaskNet is to estimate a *confidence map* for the synthesized vector fields, which estimate how accurate each voxel is in InterpolationNet. As shown in Figure 3 (b), MaskNet also has three individual neural nets, which process u, v, and w components separately. In each neural net, four Conv layers are applied to downscale the input by 16 times. Then four DeConv layers are utilized to upscale the learned feature maps to the original dimension. After each DeConv layer, skip connection and one Conv layer are leveraged to traverse information faster. Finally, sigmoid$(\cdot)$ is added to normalize the data range to $[0, 1]$. Note that the output from MaskNet should not be a binary mask; otherwise, the network is difficult to optimize

because the gradient cannot be propagated layer by layer. The parameter details are listed in Table 1.

Blending. Once we collect the outputs from InterpolationNet and MaskNet, namely, $\{\bar{\mathbf{F}}_{u,i+1}, \bar{\mathbf{F}}_{v,i+1}, \bar{\mathbf{F}}_{w,i+1}\}$, $\cdots$, $\{\bar{\mathbf{F}}_{u,i+s}, \bar{\mathbf{F}}_{v,i+s}, \bar{\mathbf{F}}_{w,i+s}\}$ and $\{\bar{\mathbf{M}}_{u,i+1}, \bar{\mathbf{M}}_{v,i+1}, \bar{\mathbf{M}}_{w,i+1}\}$, $\cdots$, $\{\bar{\mathbf{M}}_{u,i+s}, \bar{\mathbf{M}}_{v,i+s}, \bar{\mathbf{M}}_{w,i+s}\}$, we compute the synthesized vector field $\hat{\mathbf{F}}_j$ using the following blending equations

$$\hat{\mathbf{F}}_{u,j} = \bar{\mathbf{M}}_{u,j} \odot \bar{\mathbf{F}}_{u,j} + (\mathbf{1} - \bar{\mathbf{M}}_{u,j}) \odot \bar{\mathbf{L}}_{u,j}, \quad (4)$$

$$\hat{\mathbf{F}}_{v,j} = \bar{\mathbf{M}}_{v,j} \odot \bar{\mathbf{F}}_{v,j} + (\mathbf{1} - \bar{\mathbf{M}}_{v,j}) \odot \bar{\mathbf{L}}_{v,j}, \quad (5)$$

$$\hat{\mathbf{F}}_{w,j} = \bar{\mathbf{M}}_{w,j} \odot \bar{\mathbf{F}}_{w,j} + (\mathbf{1} - \bar{\mathbf{M}}_{w,j}) \odot \bar{\mathbf{L}}_{w,j}, \quad (6)$$

$$\hat{\mathbf{F}}_j = [\hat{\mathbf{F}}_{u,j}, \hat{\mathbf{F}}_{v,j}, \hat{\mathbf{F}}_{w,j}], \quad (7)$$

where $\odot$ is voxel-wise multiplication, $\bar{\mathbf{L}}_j$ is the corresponding linear interpolation result, and $j \in \{i+1, \cdots, i+s\}$. The rationale behind this blending equation is that since deep learning has the numerical precision limitations in predicting extreme small values (e.g., at a scale of 10^{-7}) [40], more confidence should be given for linear interpolation result in these voxels. While in the large value regions, deep learning can achieve better quality; thus, more confidence is supported for the outputs of InterpolationNet.

Loss function. To train TSR-VFD in an end-to-end fashion, we define the loss function as follows

$$\mathscr{L} = \frac{1}{3 \times k \times L \times H \times W} \sum_{j=1}^{k} \sum_{i \in \{u,v,w\}} ||\mathbf{F}_{i,j} - \hat{\mathbf{F}}_{i,j}||_2, \quad (8)$$

where k is the number of training samples, $||\cdot||$ denotes L_2 norm, and $\mathbf{F}_{i,j}$ and $\hat{\mathbf{F}}_{i,j}$ are, respectively, the GT and synthesized vector component values at the i-th vector component of the j-th training sample. L, H, and W are the dimensions of the vector fields.

4. Results and Discussion

4.1. Data Sets and Training Details

Table 2 shows the simulation data sets we experimented with. A single NVIDIA TESLA P100 GPU was used for training TSR-VFD. Note that we can apply TSR-VFD to vector fields of arbitrary size as it is fully convolutional. For optimization, we initialized parameters in TSR-VFD using those suggested by He et al. [41] and applied the Adam optimizer [42] to update the parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$). We set one training sample per mini-batch and the learning rate to 10^{-4}. All these parameters are determined based on experiments. We sampled 40% data for training, and the rest of the data are used for inference. We train $1,500$ epochs for each data set.

Table 2: The dimensions of each data set.

data set	dimension ($x \times y \times z \times t$)
half-cylinder	$640 \times 240 \times 80 \times 100$
hurricane	$256 \times 256 \times 56 \times 48$
solar plume	$64 \times 64 \times 256 \times 27$
supernova	$128 \times 128 \times 128 \times 100$
tornado	$128 \times 128 \times 128 \times 48$

4.2. Results

Baselines. We use three baselines for comparison with TSR-VFD:

- LERP: Linear interpolation (LERP) is a common method for interpolating intermediate vector fields.
- GAN: GAN includes a generator and a discriminator. The generator comprises InterpolationNet and MaskNet, and the discriminator follows that of Isola et al. [43]. We optimize GAN through adversarial loss and the loss defined in Equation 8. The weights of these two losses are set to 0.01 and 1, respectively. Such a setting can stabilize the training process and achieve better performance (e.g., higher PSNR). If adversarial loss dominates the whole training process, the training process is unstable, and the model is prone to collapse, which means that results generated by GAN would be unacceptable.
- RNN [18]: RNN is a deep learning solution for interpolating time-varying scalar data. Here we modify it for the vector field interpolation task. We leverage RNN to train u, v, and w separately, and add MaskNet into RNN to combine LERP and RNN results.

All pathline and streamline visualization results presented in the paper are synthesized by TSR-VFD, which are the inferred results (i.e., the network does not see these vector fields during training). For streamline visualization, these inferred results are from a pair of vector fields far away from the training data. Within the interpolation interval, we select the time step in the middle (i.e., we show the worst possible TSR-VFD results). All visualization results for the same data set use the same setting (i.e., the same set of randomly placed seeds and the same viewing parameters). In reference to the GT, we compare TSR-VFD results against those generated by LERP, GAN, and RNN. Due to the page limit, we only report the interpolation interval evaluation in this paper and leave the study of the other four parameter settings and network designs in the appendix. The appendix also includes additional quantitative comparison and error visualization results.

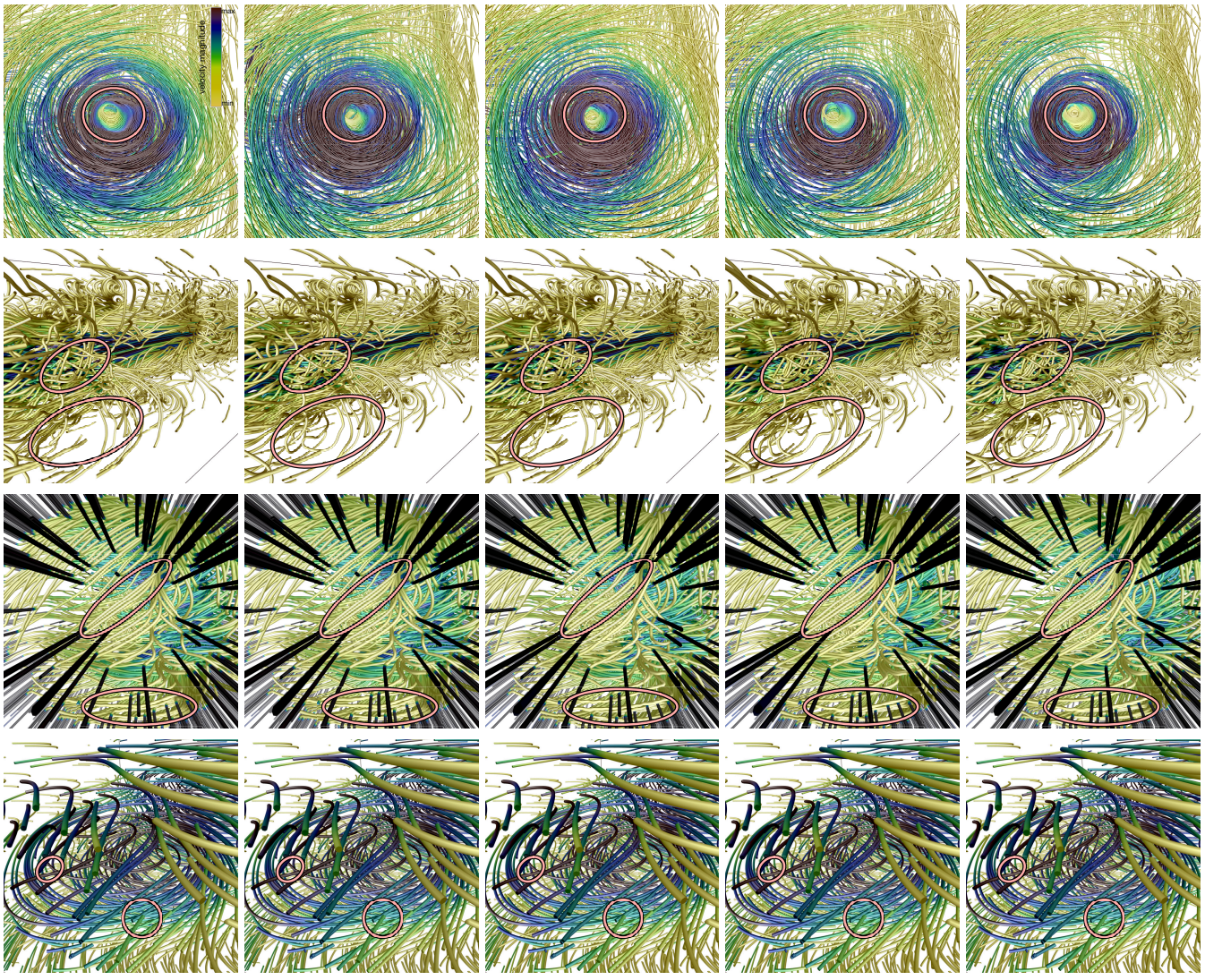
Evaluation metrics. Following Han et al. [27], we utilize peak signal-to-noise ratio (PSNR) and average angle difference (AAD) to evaluate the quality of synthesized vector fields. In addition, we leverage root-absolute-error (RAE) to measure the relative error between the synthesized vector field $\hat{\mathbf{F}}$ and GT $\mathbf{F}$ vector field. RAE is defined as

$$\mathrm{RAE}(\mathbf{F}, \hat{\mathbf{F}}) = \sqrt{\frac{\sum_{i=1}^{L \times H \times W} \sum_{j \in \{u,v,w\}} ||\mathbf{F}_{i,j} - \hat{\mathbf{F}}_{i,j}||_1}{\sum_{i=1}^{L \times H \times W} \sum_{j \in \{u,v,w\}} ||\mathbf{F}_{i,j}||_1}}, \quad (9)$$

where $||\cdot||_1$ is L_1 norm. Besides these data-level metrics, we also define pathline distance (PD) to measure the similarity of pathlines traced from GT and synthesized vector fields. PD is defined as follows

$$\mathrm{PD}(\mathbf{P}, \hat{\mathbf{P}}) = \frac{1}{\sqrt{L^2 + H^2 + W^2}} \frac{1}{N} \sum_{i=1}^{N} \frac{1}{n_i} \sum_{j=1}^{n_i} \min_{\mathbf{p} \in \mathbf{P}_i} ||\mathbf{p} - \hat{\mathbf{p}}_{ij}||_2, \quad (10)$$

where $\mathbf{P}$ and $\hat{\mathbf{P}}$ are the pathlines traced from GT and synthesized vector fields, N is the number of pathlines, n_i is the number of



(a) LERP (b) RNN (c) GAN (d) TSR-VFD (e) GT

Fig. 4: Pathline rendering results. Top to bottom: hurricane, solar plume, supernova, and tornado. The numbers of pathlines traced for each data set are 1,000, 1,000, 1,000, and 2,000, respectively.

points at the i-th pathline, $||\cdot||_2$ is L_2 norm, $\hat{\mathbf{p}}_{ij}$ is the j-th point at the i-th pathline traced from the synthesized vector field, and $\mathbf{P}_i$ is the i-th pathline traced from the GT vector field.

We also leverage three physical quantities: divergence, vorticity, and acceleration to evaluate the quality of these synthesized vector fields. Divergence represents the volume density of the outward flux of each vector. Vorticity describes the local spinning motion of each vector. Acceleration demonstrates the rate of increase of each vector with respect to time. Divergence is defined as

$$\mathrm{div}(\mathbf{F}) = \frac{\partial \mathbf{F_u}}{\partial x} + \frac{\partial \mathbf{F_v}}{\partial y} + \frac{\partial \mathbf{F_w}}{\partial z}. \tag{11}$$

The difference of divergence between the GT and synthesized vector fields is defined as

$$\mathrm{diff}(\mathbf{F}, \hat{\mathbf{F}}) = \frac{1}{L \times H \times W} \sum_{i=1}^{L \times H \times W} |\mathrm{div}(\mathbf{F})_i - \mathrm{div}(\hat{\mathbf{F}})_i|. \tag{12}$$

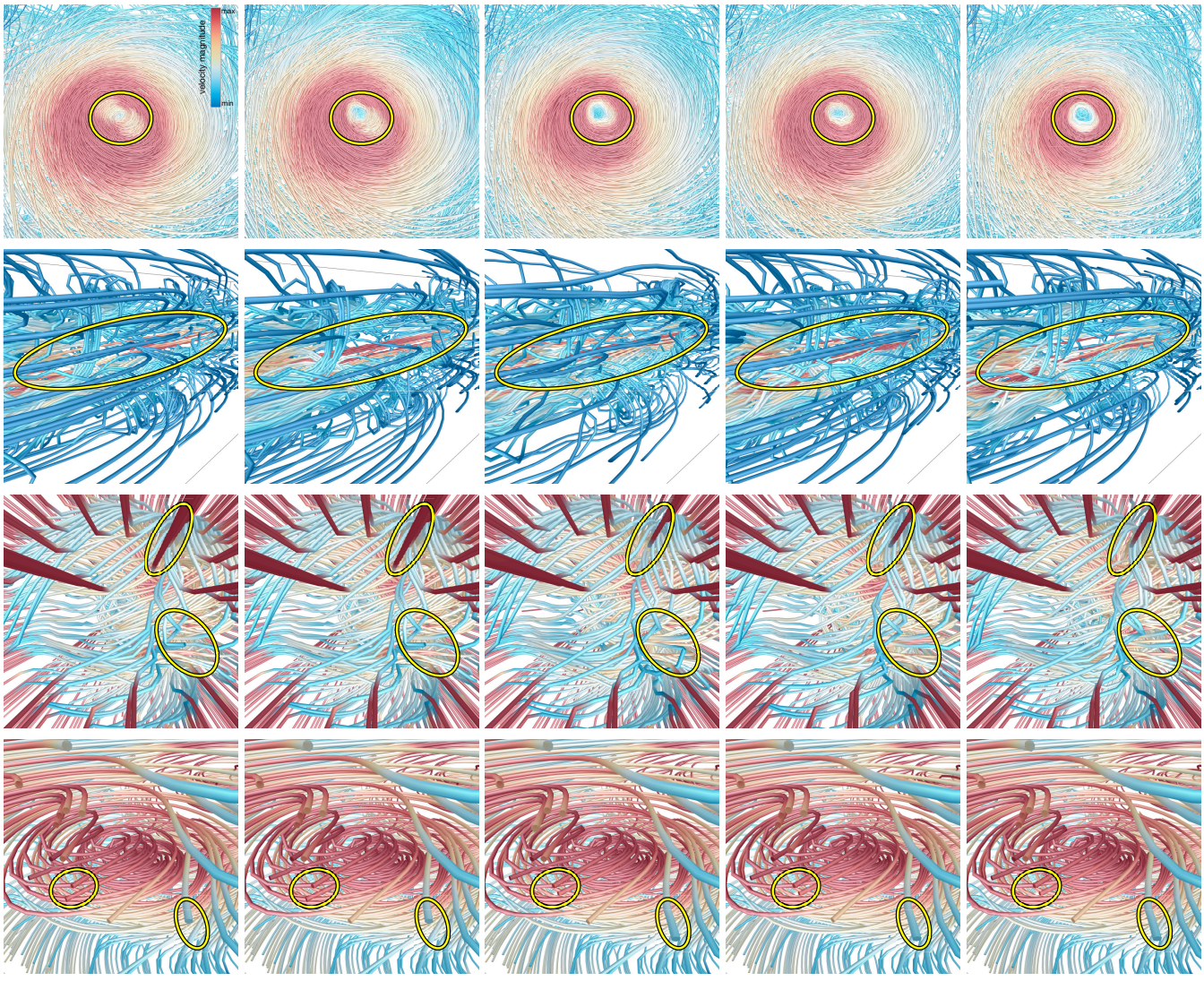
Vorticity is defined as

$$\boldsymbol{\omega}(\mathbf{F}) = \left[\frac{\partial \mathbf{F_w}}{\partial y} - \frac{\partial \mathbf{F_v}}{\partial z}, \frac{\partial \mathbf{F_u}}{\partial z} - \frac{\partial \mathbf{F_w}}{\partial x}, \frac{\partial \mathbf{F_v}}{\partial x} - \frac{\partial \mathbf{F_u}}{\partial y}\right]^T. \tag{13}$$

Acceleration is defined as

$$\begin{aligned} \mathrm{acc}(\mathbf{F}) &= \left[\frac{d\mathbf{F_u}}{dt}, \frac{d\mathbf{F_v}}{dt}, \frac{d\mathbf{F_w}}{dt}\right]^T \\ &= \left[\mathbf{F_u}\frac{\partial \mathbf{F_u}}{\partial x} + \mathbf{F_v}\frac{\partial \mathbf{F_u}}{\partial y} + \mathbf{F_w}\frac{\partial \mathbf{F_u}}{\partial z} + \frac{\partial \mathbf{F_u}}{\partial t}, \right. \\ &\quad \mathbf{F_u}\frac{\partial \mathbf{F_v}}{\partial x} + \mathbf{F_v}\frac{\partial \mathbf{F_v}}{\partial y} + \mathbf{F_w}\frac{\partial \mathbf{F_v}}{\partial z} + \frac{\partial \mathbf{F_v}}{\partial t}, \\ &\quad \left. \mathbf{F_u}\frac{\partial \mathbf{F_w}}{\partial x} + \mathbf{F_v}\frac{\partial \mathbf{F_w}}{\partial y} + \mathbf{F_w}\frac{\partial \mathbf{F_w}}{\partial z} + \frac{\partial \mathbf{F_w}}{\partial t}\right]^T. \end{aligned} \tag{14}$$

Furthermore, we leverage learned perceptual image patch similarity (LPIPS) [1], an image-space quality metric, to evaluate the quality of rendering images between the synthesized and GT vector fields.



(a) LERP (b) RNN (c) GAN (d) TSR-VFD (e) GT

Fig. 5: Streamline rendering results. Top to bottom: hurricane, solar plume, supernova, and tornado. The numbers of streamlines traced for each data set are 500, 100, 500, and 500, respectively.

Quantitative and qualitative analysis. In Figure 4, we compare pathline rendering results of the synthesized vector fields generated by LERP, RNN, GAN, and TSR-VFD. For the hurricane data set, compared with LERP and RNN, GAN and TSR-VFD preserve a better flow pattern around the hurricane's eye. For the solar plume data set, TSR-VFD generates more accurate flow patterns at the central region and the bottom-left corner. For the supernova data set, TSR-VFD leads to more details around the supernova's center and bottom corner. For instance, the pathlines produced by LERP and RNN around the supernova's center are squeezed closer compared with GT, and the pathlines produced by GAN at the bottom corner are jaggy. For the tornado data set, TSR-VFD yields more details compared with LERP, RNN, and GAN. For example, the pathlines generated by LERP, RNN, and GAN are cross together at the bottom corner, while those generated by TSR-VFD and GT are still separated. Besides, the pathlines synthesized by LERP and RNN are shifted a little bit around the tornado's center.

In Figure 5, we compare streamline rendering results of the synthesized vector fields generated by LERP, RNN, GAN, and TSR-VFD. For the hurricane data set, TSR-VFD and GAN preserve the flow patterns better around the hurricane's eye. LERP and RNN do not capture such patterns: the streamlines generated by LERP and RNN exhibit an ellipse shape instead of a circle shape. For the solar plume data set, LERP, RNN, and GAN do not accurately trace the streamlines at the central region, i.e., there are more streamlines traced from the central region compared with those generated by TSR-VFD. For the supernova data set, TSR-VFD yields a better visual result at the top region, while LERP and RNN do not recover these streamlines, and GAN produces more streamline at the bottom-right corner. For the tornado data set, TSR-VFD yields more details compared with LERP, RNN, and GAN. For example, LERP and RNN produce longer streamlines around the tornado's center and shorter streamlines at the bottom-right corner, while GAN and TSR-VFD still preserve these details compared with GT

Table 3: Average PSNR (dB), AAD, RAE, PD, difference of divergence, LPIPS values of streamline rendering (SR) and pathline rendering (PR), training time per epoch (in second), and inference time (in second) under interpolation interval $s = 5$. The best ones are highlighted in bold (same for Tables 4 to 7).

data set	method	PSNR	AAD	RAE	PD	diff of div	LPIPS (SR)	LPIPS (PR)	train	infer
hurricane	LERP	38.26	0.060	0.429	0.0413	7.03×10^{-5}	0.293	0.362	—	—
	GAN	40.45	0.049	0.389	0.0342	8.21×10^{-4}	0.274	0.322	126.08	23.2
	RNN	38.46	0.061	0.428	0.0534	6.03×10^{-4}	0.285	0.349	130.05	63.1
	TSR-VFD	**41.70**	**0.043**	**0.362**	**0.0274**	7.38×10^{-5}	**0.269**	**0.299**	55.61	23.2
solar plume	LERP	40.75	**0.006**	0.367	0.0013	1.48×10^{-7}	0.392	0.137	—	—
	GAN	42.65	0.051	0.386	0.0045	1.88×10^{-5}	0.449	0.222	23.97	0.08
	RNN	41.78	0.008	0.352	0.0015	2.35×10^{-6}	0.419	0.113	19.74	0.23
	TSR-VFD	**45.39**	0.010	**0.302**	**0.0012**	1.79×10^{-7}	**0.386**	**0.113**	8.81	0.08
supernova	LERP	43.84	0.0047	0.0933	0.0043	6.78×10^{-8}	0.227	0.142	—	—
	GAN	46.84	0.0056	0.1144	**0.0038**	1.36×10^{-5}	0.229	0.149	91.06	0.20
	RNN	43.83	0.0047	0.0933	0.0043	9.50×10^{-7}	0.227	0.134	88.83	0.57
	TSR-VFD	**48.43**	**0.0039**	**0.0806**	**0.0038**	1.62×10^{-7}	**0.223**	**0.134**	38.20	0.20
tornado	LERP	51.03	0.00031	0.0498	0.00006	3.22×10^{-7}	0.039	0.029	—	—
	GAN	51.24	0.00184	0.0850	0.00012	3.18×10^{-6}	0.043	0.030	36.06	0.20
	RNN	51.04	0.00030	0.0498	0.00006	3.24×10^{-7}	0.039	0.029	34.95	0.57
	TSR-VFD	**53.77**	**0.00026**	**0.0425**	**0.00003**	1.86×10^{-7}	**0.016**	**0.004**	15.45	0.20

streamlines.

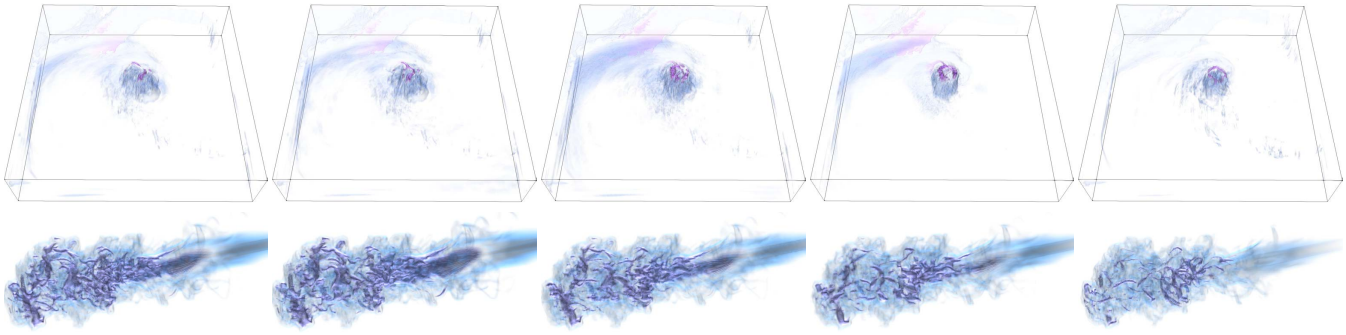
In Table 3, we report the average PSNR (higher is better), AAD (lower is better), and RAE (lower is better) values over the entire vector field sequence for LERP, GAN, RNN, and TSR-VFD. Overall, TSR-VFD performs the best in all except for one case (where LERP performs the best in terms of AAD for the solar plume data set). In addition, Table 3 reports PD values for LERP, GAN, RNN, and TSR-VFD. TSR-VFD achieves a PD of 0.0274 for the hurricane data set, while LERP achieves a PD of 0.0413. As for the tornado data set, TSR-VFD reports a PD of 0.00003, while LERP reports a PD of 0.00006. Overall, TSR-VFD produces the lowest PD values for all the data sets. The average divergence values are also reported in Table 3. In general, LERP can achieve the lowest divergence values except for the tornado data set.

Table 3 also shows the average training time per epoch and average inference time. In terms of training time, we can observe that TSR-VFD takes the shortest time compared with RNN and GAN since RNN needs to unroll the recurrent layer s times, and GAN needs to go through two networks during training. For TSR-VFD, the entire training times for the hurricane, solar plume, supernova, and tornado data sets are 23.17, 3.67, 15.92, and 6.44 hours, respectively. Regardless of different data sizes, the model size of TSR-VFD is 84 MB, while that of RNN is 117 MB. In terms of inference time, TSR-VFD is 3× faster compared with RNN. Therefore, TSR-VFD achieves the best quality in terms of speed and performance. In Table 3, we also report LPIPS values of streamline rendering and pathline rendering images. TSR-VFD achieves the lowest LPIPS values of both rendering images across all four data sets, producing the closest visual results compared with GT.

In summary, TSR-VFD achieves the highest PSNR values and lowest AAD, RAE, and PD values, and LERP achieves the lowest PSNR values and highest AAD, RAE, and PD values among all approaches. This is because LERP only assumes that unsteady flow changes linearly. Therefore, it does not capture the dynamic flow patterns as time step goes, especially for complex flows (e.g., the hurricane and supernova data sets). RNN achieves similar results compared with LERP and still does not produce high-quality vector fields since RNN runs into gradient vanishing when the interpolation step is large. GAN does not generate high-fidelity vector fields either and performs unstably on evaluation metrics. For example, GAN achieves a 51.24 dB PSNR value but with a 0.00184 AAD value for the tornado data set. This is because using adversarial loss will disturb the direction and magnitude of velocity, and this small disturbance will significantly hurt the performance.

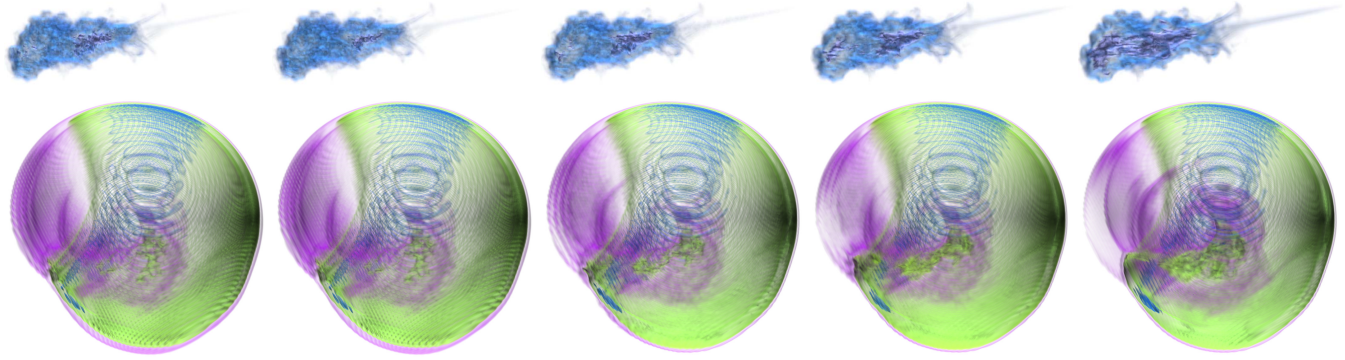
In Figures 6 and 7, we compare volume rendering results of vorticity magnitude and acceleration magnitude of the synthesized vector fields generated by LERP, RNN, GAN, TSR-VFD, and GT. In Figure 6, for the hurricane data set, compared with LERP, RNN, and GAN, the vorticity magnitude produced by TSR-VFD around the hurricane's eye is closer to the GT. For the solar plume data set, TSR-VFD produces closer features at the center of the solar plume (i.e., the purple part) compared with LERP, RNN, and GAN. In Figure 7, for the solar plume data set, TSR-VFD produces smoother acceleration magnitude results at the boundary of the solar plume (i.e., the blue part) compared with LERP, RNN, and GAN. For the supernova data set, TSR-VFD and GAN produce more similar results at the supernova's core compared with those generated by LERP and RNN.

Comparison against compression. In Figure 8, we compare the rendering results of streamlines and pathlines traced from the synthesized vector fields generated by TSR-VFD and the vector fields compressed then decompressed using a lossy compression (LC) scheme [44]. We choose this LC scheme as it can effectively control data distortion while significantly reducing data size. For a fair comparison, we consider two scenarios: (1) keeping the same PSNR value (i.e., 45.5 dB) for the solar plume data set and (2) controlling the same compression ratio (i.e., 9×) for the supernova data set. For the first scenario, LC does not preserve the flow patterns around the solar plume's boundary. The top table of Table 4 reports the compression ratios, average AAD and RAE values for both methods. Under the same PSNR value, although LC achieves a much higher compression ratio, TSR-VFD achieves lower AAD and RAE values for the synthesized vector fields compared with those recovered from LC. For the second scenario, LC does not preserve the flow patterns well around the supernova's center, and the pathlines are rather jaggy upon closer examination. The bottom table of Table 4 reports average PSNR, AAD, and RAE values for both methods. Under the same compression ratio, TSR-VFD achieves higher PSNR, lower AAD and RAE values.



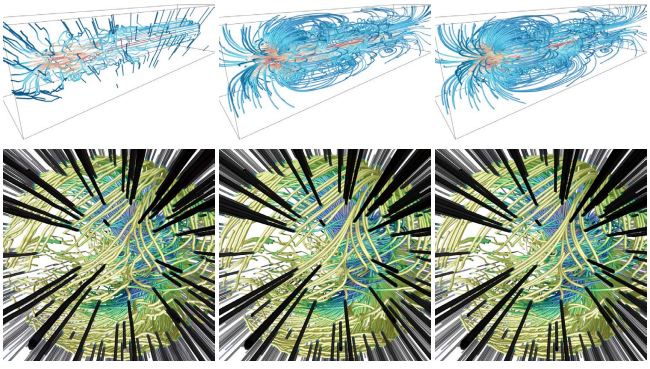
(a) LERP (b) RNN (c) GAN (d) TSR-VFD (e) GT

Fig. 6: Volume rendering results of vorticity magnitude of the synthesized vector fields using the hurricane (top) and solar plume (bottom) data sets.



(a) LERP (b) RNN (c) GAN (d) TSR-VFD (e) GT

Fig. 7: Volume rendering results of acceleration magnitude of the synthesized vector fields using the solar plume (top) and supernova (bottom) data sets.



(a) LC (b) TSR-VFD (c) GT

Fig. 8: Zoom-in streamline (top row) and pathline (bottom row) rendering results using the solar plume and supernova data sets, respectively. 100 streamlines and 1,000 pathlines are traced from the respective vector fields recovered from lossy compression (LC) and synthesized by TSR-VFD.

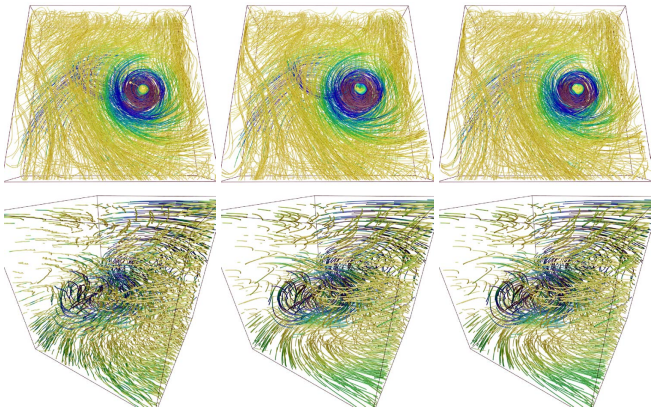
Table 4: Comparison of lossy compression (LC) and TSR-VFD. Top: average AAD and RAE values under the same PSNR value (i.e., 45.5 dB). Bottom: average PSNR (dB), AAD, and RAE values under the same compression ratio (i.e., 9×).

data set	method	comp. ratio	AAD	RAE
solar plume	LC	**36**×	0.492	0.547
	TSR-VFD	5×	**0.010**	**0.302**

data set	method	PSNR	AAD	RAE
supernova	LC	35.31	0.0249	0.2954
	TSR-VFD	**45.32**	**0.0053**	**0.0967**

Cross-dataset and ensemble evaluation. To evaluate the generalization of TSR-VFD, we consider two training cases: cross-dataset and ensemble data set. For cross-dataset, we perform joint training using the hurricane and tornado data sets. The number of epochs is the same as the one used in separate training. The pathline rendering results are shown in Figure 9. For the hurricane data set, the pathlines generated from joint training are worse than those generated from separate training. For example, the joint model does not capture the flow pattern at the hurricane's eye. For the tornado data set, the joint model performs poorly: it leads to jaggy pathlines everywhere and does not accurately recover the velocity magnitude information. We can also observe that separate training leads to higher PSNR, lower AAD, and lower RAE values, as reported in Table 5. For the ensemble data set, we use the half-cylinder data set with different Reynolds numbers to train TSR-VFD jointly. As shown in Figure 10, the streamlines generated by our results are closer to the GT compared with these generated by LERP. For example, for the Reynolds number of 320, LERP produces two swirls at the domain's center while TSR-VFD and GT only yield one swirl. As for pathline rendering, LERP leads to fewer pathlines at the middle-right corner. For vorticity of the Reynolds number of 6,400, TSR-VFD produces more details. For quantitative results, TSR-VFD achieves higher PSNR values and lower AAD, RAE, and PD values compared with LERP under different Reynolds numbers, as shown in Table 6.

Evaluation of interpolation interval. To investigate the interpolation ability of TSR-VFD, we conduct an experiment



(a) joint training (b) separate training (c) GT

Fig. 9: Different ways of training the hurricane (top row) and tornado (bottom row) data sets. 1,000 and 1,500 pathlines are traced, respectively.

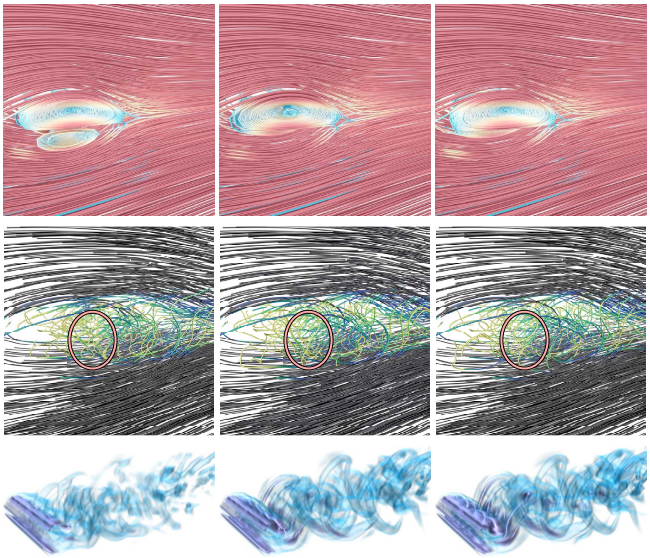
Table 5: Average PSNR (dB), AAD, and RAE values under $s = 5$ using different ways of training.

data set	way	PSNR	AAD	RAE
hurricane	joint	41.20	0.045	0.373
	separate	**41.70**	**0.043**	**0.362**
tornado	joint	29.45	0.08532	0.5306
	separate	**53.77**	**0.00026**	**0.0425**

that trains TSR-VFD with different interpolation intervals. As shown in Figure 11, we render the pathlines and streamlines from the vector fields synthesized from different interpolation intervals. For the hurricane data set, under $s = 9$ and $s = 13$, the hurricane's eye cannot be recovered well, especially for the eye's shape. Moreover, under $s = 13$, the pathlines do not capture the pattern of the hurricane's eye. However, under $s = 5$, this feature can be preserved much better in streamline and pathline rendering results. For the supernova and tornado data sets, all results are similar to GT. But taking a closer look, we can observe that several streamlines of the supernova data set are missed at the top-left corner. Several pathlines of the tornado data set are shifted in the central region. Therefore, the appropriate value for s is 5 for complex data sets, where the flow pattern moves as the time step goes (e.g., the hurricane data set). For simple data sets where the flow pattern is localized (e.g., the tornado and the supernova data sets), s could be 9. Average PSNR, AAD, and RAE values under different interpolation intervals are reported in Table 7. We can observe that TSR-VFD produces higher PSNR, lower AAD, and lower RAE values for different data sets compared to LERP. The only exception is AAD for the solar plume data set. Note that for the solar plume data set, only 12 training samples are offered. Therefore, setting s to 13 is not an option.

Table 6: Average PSNR (dB), AAD, RAE, and PD values under $s = 5$ (1st and 2nd rows) and $s = 3$ (3rd row).

Reynolds number	method	PSNR	AAD	RAE	PD
160	LERP	49.07	0.005	0.107	0.00071
	TSR-VFD	**53.36**	**0.003**	**0.090**	**0.00032**
320	LERP	42.25	0.009	0.163	0.00232
	TSR-VFD	**47.22**	**0.007**	**0.143**	**0.00217**
6,400	LERP	33.63	0.014	0.259	0.00367
	TSR-VFD	**47.90**	**0.002**	**0.135**	**0.00145**



(a) LERP (b) TSR-VFD (c) GT

Fig. 10: Ensemble evaluation of TSR-VFD using the half-cylinder data set with Reynolds numbers of 320 (1st row), 160 (2nd row), and 6,400 (3rd row). 1st row: 1,000 streamlines, 2nd row: 2,000 pathlines, and 3rd row: vorticity.

Table 7: Average PSNR (dB), AAD, and RAE values under different interpolation intervals s.

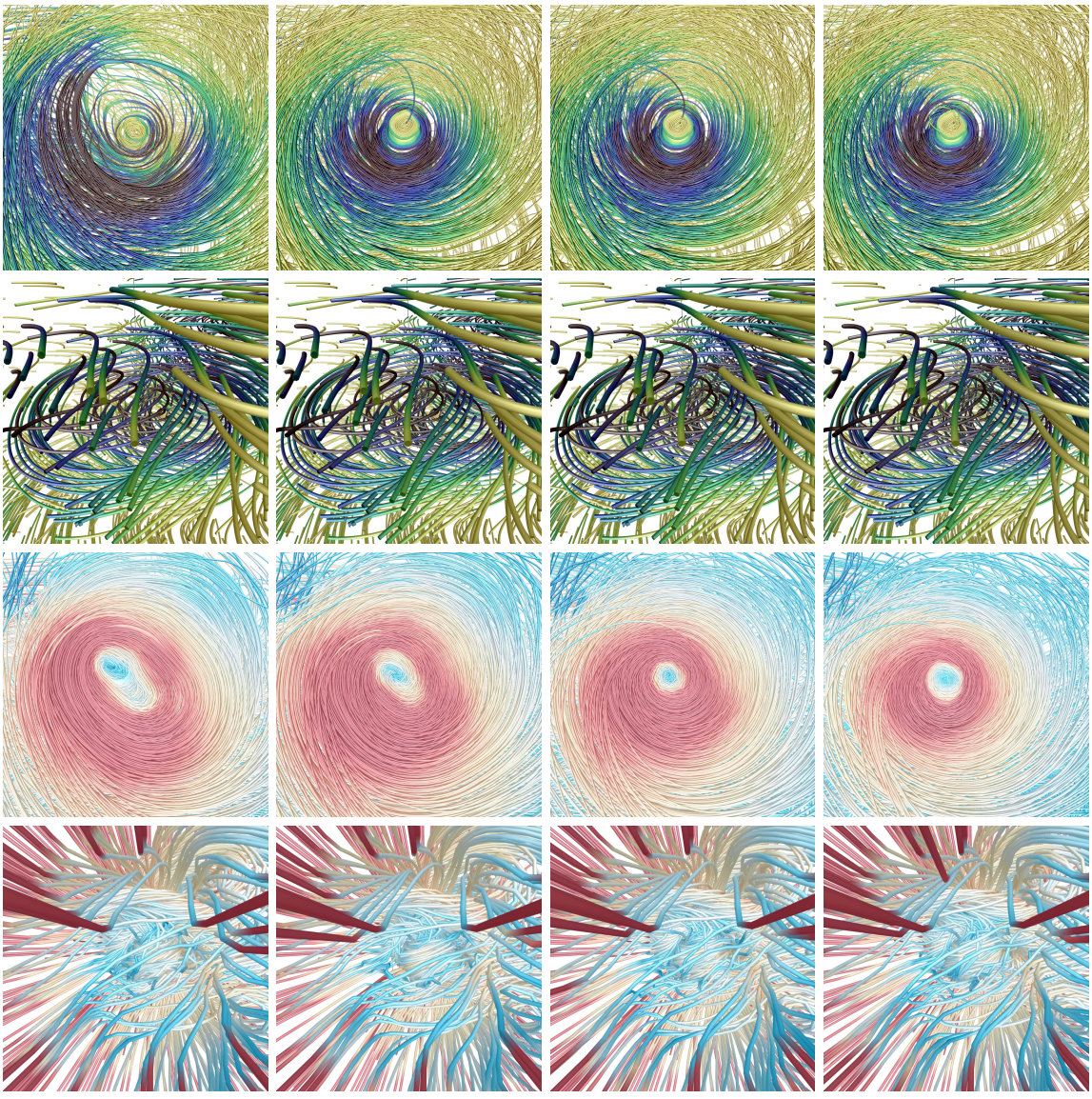
		PSNR		AAD		RAE	
data set	s	LERP	TSR-VFD	LERP	TSR-VFD	LERP	TSR-VFD
hurricane	5	38.26	**41.70**	0.060	**0.043**	0.429	**0.362**
	9	36.05	**39.07**	0.076	**0.060**	0.488	**0.427**
	13	34.30	**37.67**	0.086	**0.070**	0.529	**0.464**
solar plume	5	40.75	**45.39**	**0.006**	0.010	0.367	**0.302**
	9	38.79	**43.47**	**0.010**	0.014	0.430	**0.354**
supernova	5	43.84	**48.43**	0.0047	**0.0039**	0.0933	**0.0806**
	9	40.49	**45.32**	0.0064	**0.0053**	0.1143	**0.0967**
	13	38.69	**43.37**	0.0081	**0.0066**	0.1287	**0.1086**
tornado	5	51.03	**53.77**	0.00031	**0.00026**	0.0498	**0.0425**
	9	48.55	**53.54**	0.00082	**0.00038**	0.0723	**0.0502**
	13	46.62	**51.71**	0.00142	**0.00067**	0.0932	**0.0614**

4.3. Discussion

Our approach assumes that the intermediate time steps do not exhibit flow characteristics that are not observed in the two ending time steps. Our approach cannot capture such patterns if a flow pattern appears and disappears only in the interpolated time steps. Compared with LERP, TSR-VFD offers two advantages. First, it achieves better quantitive and qualitative results, including physics-based measurements, such as vorticity and acceleration. Second, it preserves important flow patterns more accurately, such as the supernova's core and the hurricane's eye. However, TSR-VFD requires some successive time steps for training. TSR-VFD better interpolates the flows with complex patterns, such as turbulence (e.g., Reynolds number of 6,400).

5. Conclusions and Future Work

We have presented TSR-VFD, a novel deep learning solution for interpolating missing vector fields for VFD analysis and visualization. Leveraging InterpolationNet and MaskNet, TSR-VFD can produce high-quality unsteady vector fields given vector fields at both ends as input. Compared to LERP, RNN, and



(a) $s = 13$ (b) $s = 9$ (c) $s = 5$ (d) GT

Fig. 11: Pathline (1,000, 1st and 2nd rows) and streamline (500, 3rd and 4th rows) rendering results under different interpolation intervals. Top to bottom: hurricane, supernova, hurricane, and tornado.

GAN, TSR-VFD yields synthesized vector fields of better visual quality, both qualitatively and quantitatively. We also compare TSR-VFD against a lossy compression scheme. TSR-VFD can be applied to the in-situ situation. In this scenario, at simulation time, we can store the early time steps for TSR-VFD training while saving the later time steps sparsely (e.g., storing one time step for every ten time steps simulated) for storage saving. During postprocessing, the network is trained with the early time steps only. Once trained, we can recover the missing intermediate vector fields with high fidelity, given the sparsely-output later time steps.

In the future, we would improve TSR-VFD in two aspects. (1) Few-shot learning. The current version of TSR-VFD requires 40% data for training, which could hinder its utilization in real applications. With few-shot learning techniques [45], we can significantly reduce the training samples while preserving the quality of interpolated vector fields. (2) Incorporating physical constraints. So far, TSR-VFD is purely a data-driven solution for VFD. We want to incorporate physical constraints into optimization, which can further improve the quality of synthesized vector fields.

Acknowledgements

This research was supported in part by the U.S. National Science Foundation through grants IIS-1455886, CNS-1629914, DUE-1833129, IIS-1955395, IIS-2101696, OAC-2104158, and the NVIDIA GPU Grant Program. The authors would like to thank the anonymous reviewers for their insightful comments.

References

[1] 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.

[2] Mussa-Ivaldi, FA. From basis functions to basis fields: Vector field approximation from sparse data. Biological Cybernetics 1992;67(6):479–489.

[3] Gouesbet, G, Letellier, C. Global vector-field reconstruction by using a multivariate polynomial L2 approximation on nets. Physical Review E 1994;49(6):4955–4972.

[4] Lage, M, Petronetto, F, Paiva, A, Lopes, H, Lewiner, T, Tavares, G. Vector field reconstruction from sparse samples with applications. In: Proceedings of Brazilian Symposium on Computer Graphics and Image Processing. 2006, p. 297–306.

[5] Letellier, C, Le Sceller, L, Dutertre, P, Gouesbet, G, Fei, Z, Hudson, J. Topological characterization and global vector field reconstruction of an experimental electrochemical system. The Journal of Physical Chemistry 1995;99(18):7016–7027.

[6] Chen, Y, Cohen, J, Krolik, J. Similarity-guided streamline placement with error evaluation. IEEE Transactions on Visualization and Computer Graphics 2007;13(6):1448–1455.

[7] Xu, L, Lee, TY, Shen, HW. An information-theoretic framework for flow visualization. IEEE Transactions on Visualization and Computer Graphics 2010;16(6):1216–1224.

[8] Xu, C, Prince, JL. Gradient vector flow: A new external force for snakes. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 1997, p. 66–71.

[9] Tao, J, Ma, J, Wang, C, Shene, CK. A unified approach to streamline selection and viewpoint selection for 3D flow visualization. IEEE Transactions on Visualization and Computer Graphics 2013;19(3):393–406.

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

[11] Zhou, Z, Hou, Y, Wang, Q, Chen, G, Lu, J, Tao, Y, et al. Volume upscaling with convolutional neural networks. In: Proceedings of Computer Graphics International. 2017, p. 38:1–38:6.

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

[13] Cheng, HC, Cardone, A, Jain, S, Krokos, E, Narayan, K, Subramaniam, S, et al. Deep-learning-assisted volume visualization. IEEE Transactions on Visualization and Computer Graphics 2019;25(2):1378–1391.

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

[15] 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.

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

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

[18] Han, J, Wang, C. TSR-TVD: Temporal super-resolution for time-varying data analysis and visualization. IEEE Transactions on Visualization and Computer Graphics 2020;26(1):205–215.

[19] Han, J, Wang, C. SSR-TVD: Spatial super-resolution for time-varying data analysis and visualization. IEEE Transactions on Visualization and Computer Graphics 2020;Accepted.

[20] Han, J, Zheng, H, Chen, DZ, Wang, C. STNet: An end-to-end generative framework for synthesizing spatiotemporal super-resolution volumes. IEEE Transactions on Visualization and Computer Graphics 2022;28(1):270–280.

[21] Hong, F, Zhang, J, Yuan, X. Access pattern learning with long short-term memory for parallel particle tracing. In: Proceedings of IEEE Pacific Visualization Symposium. 2018, p. 76–85.

[22] Xie, Y, Franz, E, Chu, M, Thuerey, N. tempoGAN: A temporally coherent, volumetric GAN for super-resolution fluid flow. ACM Transactions on Graphics 2018;37(4):95:1–95:15.

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

[24] Wiewel, S, Becher, M, Thuerey, N. Latent-space physics: Towards learning the temporal evolution of fluid flow. Computer Graphics Forum 2019;38(2):71–82.

[25] Kim, B, Günther, T. Robust reference frame extraction from unsteady 2D vector fields with convolutional neural networks. Computer Graphics Forum 2019;38(3):285–295.

[26] Werhahn, M, Xie, Y, Chu, M, Thuerey, N. A multi-pass GAN for fluid flow super-resolution. Proceedings of the ACM on Computer Graphics and Interactive Techniques 2019;2(2):1–21.

[27] Han, J, Tao, J, Wang, C. FlowNet: A deep learning framework for clustering and selection of streamlines and stream surfaces. IEEE Transactions on Visualization and Computer Graphics 2020;26(4):1732–1744.

[28] Jakob, J, Gross, M, Günther, T. A fluid flow data set for machine learning and its application to neural flow map interpolation. IEEE Transactions on Visualization and Computer Graphics 2021;27(2):1279–1289.

[29] Guo, L, Ye, S, Han, J, Zheng, H, Gao, H, Chen, DZ, et al. SSR-VFD: Spatial super-resolution for vector field data analysis and visualization. In: Proceedings of IEEE Pacific Visualization Symposium. 2020, p. 71–80.

[30] Wiewel, S, Kim, B, Azevedo, VC, Solenthaler, B, Thuerey, N. Latent space subdivision: stable and controllable time predictions for fluid flow. Computer Graphics Forum 2020;39(8):15–25.

[31] Sahoo, S, Berger, M. Integration-Aware Vector Field Super Resolution. In: Proceedings of Eurographics Visualization Conference (Short Papers). 2021, p. 49–53.

[32] Niklaus, S, Mai, L, Liu, F. Video frame interpolation via adaptive convolution. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2017, p. 670–679.

[33] Nguyen, AD, Kim, W, Kim, J, Lee, S. Video frame interpolation by plug-and-play deep locally linear embedding. arXiv preprint arXiv:180701462 2018;.

[34] Jiang, H, Sun, D, Jampani, V, Yang, MH, Learned-Miller, E, Kautz, J. Super SloMo: High quality estimation of multiple intermediate frames for video interpolation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2018, p. 9000–9008.

[35] Gehring, J, Auli, M, Grangier, D, Yarats, D, Dauphin, YN. Convolutional sequence to sequence learning. In: Proceedings of International Conference on Machine Learning. 2017, p. 1243–1252.

[36] Yu, AW, Dohan, D, Luong, MT, Zhao, R, Chen, K, Norouzi, M, et al. QANet: Combining local convolution with global self-attention for reading comprehension. In: Proceedings of International Conference for Learning Representations. 2018,.

[37] He, K, Zhang, X, Ren, S, Sun, J. Deep residual learning for image recognition. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2016, p. 770–778.

[38] Ronneberger, O, Fischer, P, Brox, T. U-Net: Convolutional networks for biomedical image segmentation. In: Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention. 2015, p. 234–241.

[39] Nair, V, Hinton, GE. Rectified linear units improve restricted Boltzmann machines. In: Proceedings of International Conference on Machine Learning. 2010, p. 807–814.

[40] Gupta, S, Agrawal, A, Gopalakrishnan, K, Narayanan, P. Deep learning with limited numerical precision. In: Proceedings of International Conference on Machine Learning. 2015, p. 1737–1746.

[41] He, K, Zhang, X, Ren, S, Sun, J. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In: Proceedings of IEEE International Conference on Computer Vision. 2015, p. 1026–1034.

[42] Kingma, D, Ba, J. Adam: A method for stochastic optimization. In: Proceedings of International Conference for Learning Representations. 2015,.

[43] Isola, P, Zhu, JY, Zhou, T, Efros, AA. Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017, p. 1125–1134.

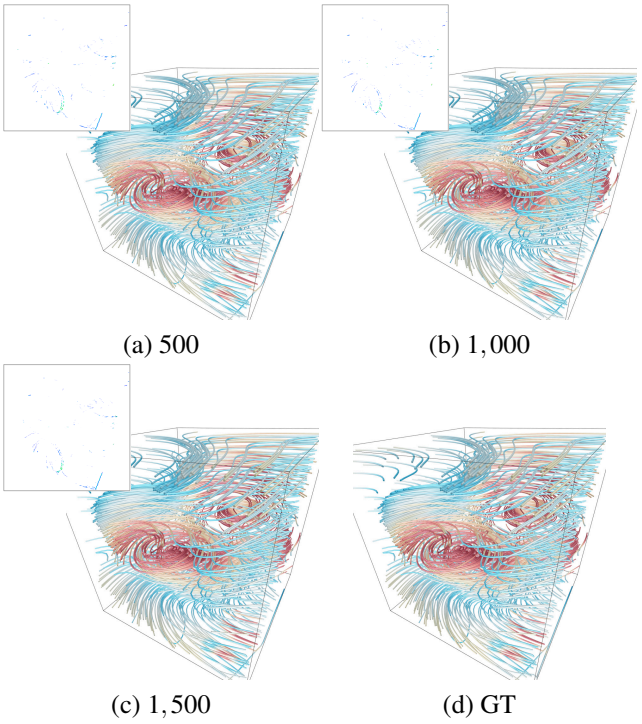
[44] Liang, X, Di, S, Tao, D, Chen, Z, Cappello, F. An efficient transformation scheme for lossy data compression with point-wise relative error bound. In: Proceedings of IEEE International Conference on Cluster Computing. 2018, p. 179–189.

[45] Finn, C, Abbeel, P, Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In: Proceedings of International Conference on Machine Learning. 2017, p. 1126–1135.

Appendix

1. Network Parameter and Design Study

To evaluate TSR-VFD, we analyze the following parameter settings and network designs: the number of training epochs, crop size for large vector fields, architecture design, and impact of MaskNet. A detailed discussion is as follows.



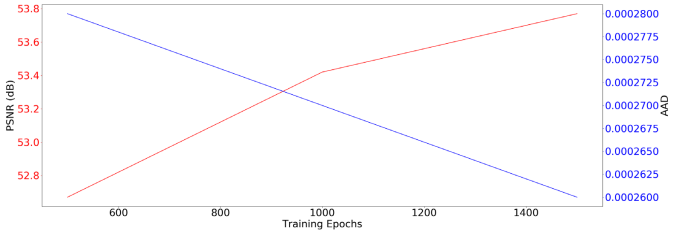
(a) 500 (b) 1,000

(c) 1,500 (d) GT

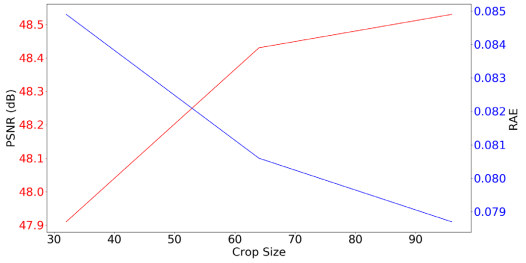
Fig. 1: Streamline rendering results under different training epochs using the tornado data set. The best match with GT is the result with 1,500 epochs. 500 streamlines are traced.

Training epochs. We study how the quality of the synthesized vector field using TSR-VFD evolves with the increase of training epochs. Streamline rendering results of the tornado data set after different numbers of training epochs are shown in Figures 1. For a clear comparison, we display a difference image at the top-left corner. The difference image is computed in the CIELUV color space [**?**], where pixels with $\Delta \geq 6.0$ are mapped to non-white colors. As we can see, there is little visual difference at either the central or the bottom region of the tornado. Nevertheless, the average PSNR and AAD values can be improved with more training epochs, as shown in Figure 2 (a). Moreover, we observe that after 1,500 epochs, there is no significant difference among synthesized results when increasing the number of epochs further. Therefore, we choose 1,500 epochs to train all the data sets.

Crop size. TSR-VFD cannot afford enough memory to process the entire vector field simultaneously if a larger interpolation interval is required or larger vector fields are provided. Therefore, we crop subvolumes to train TSR-VFD. We perform training with subvolume sizes of 32^3, 64^3, and 96^3 using the supernova data set under $s = 5$. The average PSNR and AAD curves are shown in Figure 2 (b). We can observe that training TSR-VFD benefits from a larger subvolume size since an



(a) tornado



(b) supernova

Fig. 2: Comparison of parameter settings. (a) Average PSNR and AAD under different training epochs. (b) Average PSNR and RAE under different crop sizes.

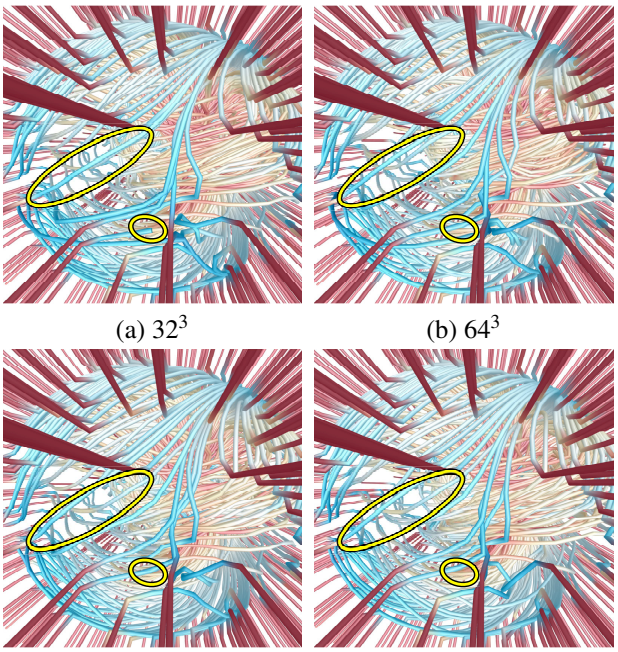
enlarged receptive field helps the network capture more information of the vector fields. As for visual quality, from Figure 3, we can see that under the subvolume size of 32^3, several streamlines are shifted around the supernova's center, and one streamline is not traced into the center. However, we do not observe a significant difference in the streamline rendering results under the subvolume size of 64^3 and 96^3, compared with GT. However, a larger subvolume size takes more time to train. Hence, we suggest that using a 64^3 subvolume size to train TSR-VFD could achieve a good tradeoff.

Architecture design. To investigate the effectiveness of TSR-VFD, we conduct an experiment that trains TSR-VFD without separating the three components using different data sets. As shown in Figure 4, we render the streamlines and pathlines from the synthesized vector fields generated by these two different architectures using the solar plume data set. Compared with TSR-VFD without separation, TSR-VFD better captures the flow features in the surrounding region for pathline rendering and in the central region for streamline rendering. For a clearer comparison, we compute PSNR and RAE values for each component as well as AAD, as shown in Table 1. As we can see, separating the vector field brings higher PSNR values and lower RAE values for each vector component for both the solar plume and supernova data sets. In addition, the benefit reduces for small components. For example, for the solar plume data set, the PSNR value of the w component increases from 41.28 dB to 43.29 dB, and the RAE value of the w component decreases from 0.233 to 0.215 due to separation training.

Impact of MaskNet. To study the impact of MaskNet, we design an experiment that trains TSR-VFD with and without MaskNet using the solar plume data set. As shown in Figure 5, we render streamlines and pathlines from the vector fields generated from these two models. As we can see, adding MaskNet into TSR-VFD improves visual quality in terms of traced streamlines and pathlines. For example, the streamlines

Table 1: Average PSNR (dB), AAD, and RAE values under different architectures.

	TSR-VFD without separated neural nets									TSR-VFD with separated neural nets								
data set	$PSNR_u$	$PSNR_v$	$PSNR_w$	PSNR	AAD	RAE_u	RAE_v	RAE_w	RAE	$PSNR_u$	$PSNR_v$	$PSNR_w$	PSNR	AAD	RAE_u	RAE_v	RAE_w	RAE
solar plume	38.73	38.19	41.28	43.35	0.013	0.478	0.480	0.233	0.331	**40.76**	**40.64**	**43.29**	**45.39**	**0.010**	**0.425**	**0.421**	**0.215**	**0.302**
supernova	46.68	45.66	45.73	46.88	0.0046	0.085	0.084	0.095	0.095	**48.71**	**47.52**	**46.78**	**48.43**	**0.0039**	**0.068**	**0.069**	**0.082**	**0.081**



(a) 32^3 (b) 64^3 (c) 96^3 (d) GT

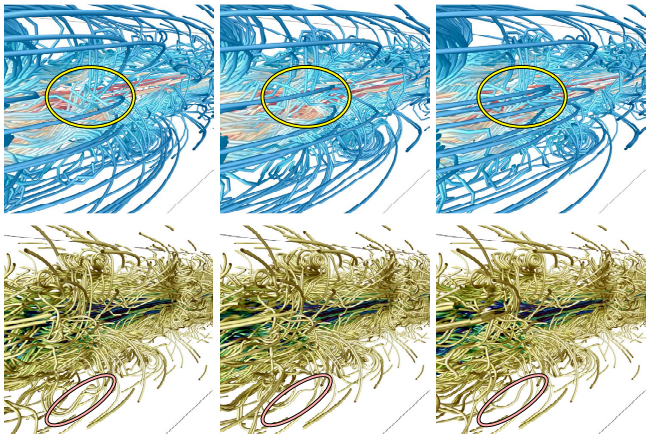
Fig. 3: Zoomed-in streamline rendering results with different crop sizes using the supernova data set. 500 streamlines are traced.

and pathlines traced from vector fields synthesized by TSR-VFD without MaskNet do not capture the surrounding flow pattern. This is because the surrounding region's velocities have smaller values compared with those in the central region. Without MaskNet, TSR-VFD could not capture these small-scale velocities due to numerical precision in deep learning. These results indicate the effectiveness of MaskNet in improving the quality of reconstructed unsteady vector fields.

2. Additional Evaluation

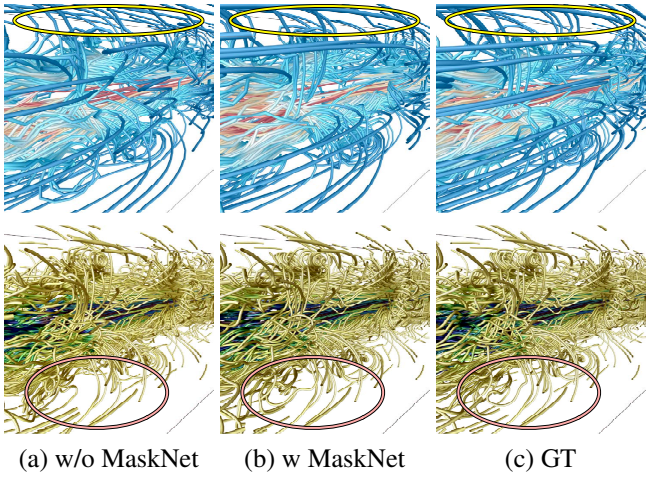
Quantitative comparison. In Figure 6, we quantitatively compare TSR-VFD results against those generated by LERP, GAN, and RNN using PSNR (higher is better), AAD (lower is better), and RAE (lower is better). TSR-VFD generally achieves the best performance compared with LERP, GAN, and RNN in terms of the highest PSNR, lowest AAD, and lowest RAE at each interpolated time step. We also observe that, except for the hurricane data set, GAN achieves the worst results in terms of AAD and RAE. This may be due to the adversarial loss, which does not train GAN as an AAD- and RAE-driven model.

Error visualization. In Figure 7, we compare volume rendering results of errors introduced by the synthesized vector fields generated by LERP, RNN, GAN, and TSR-VFD. These error volumes are calculated based on the synthesized and GT



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

Fig. 4: Zoomed-in streamline (top row) and pathline (bottom row) rendering results under different TSR-VFD architecture designs using the solar plume data data set. 100 streamlines and 1,000 pathlines are traced, respectively.



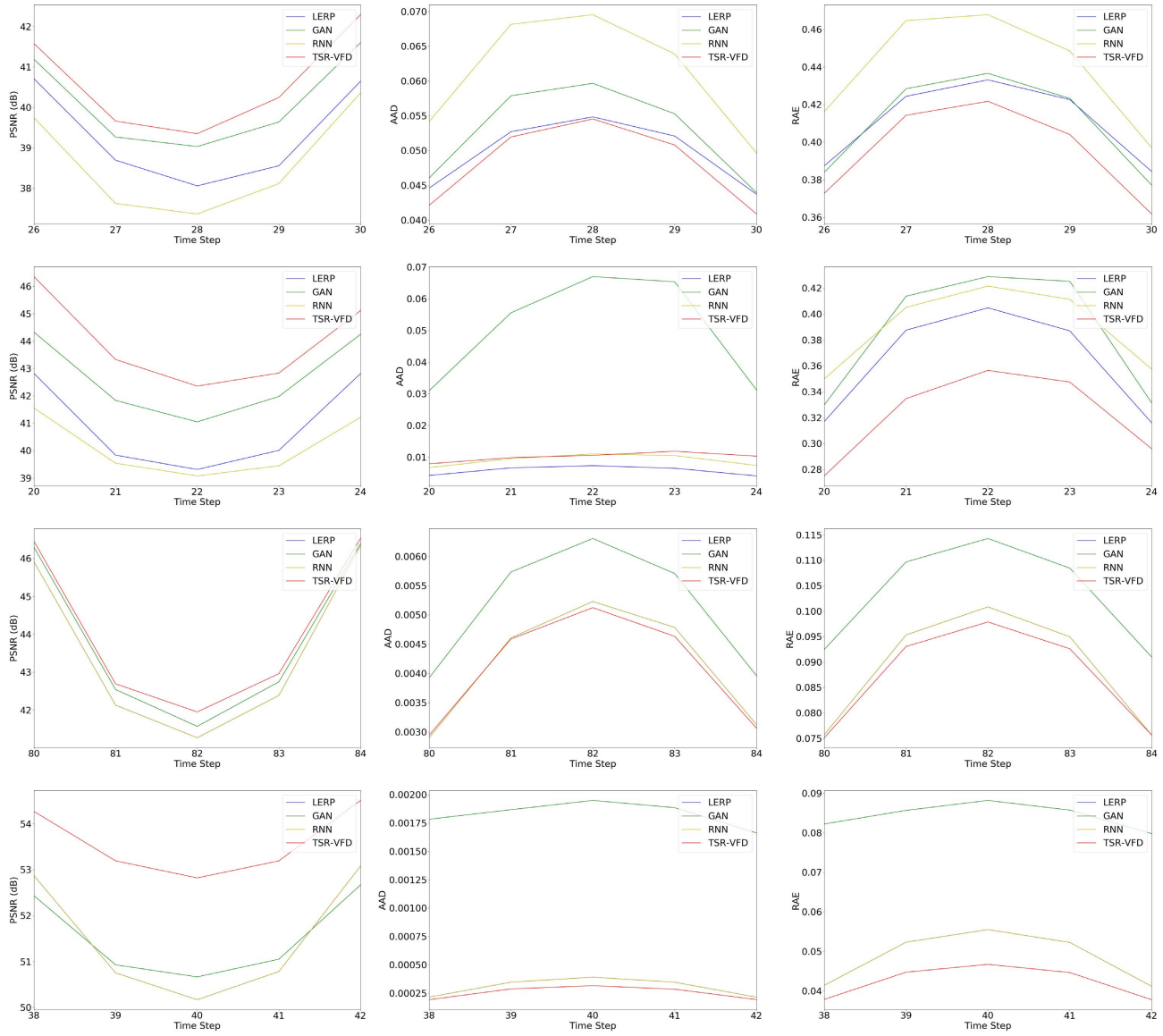
(a) w/o MaskNet (b) w MaskNet (c) GT

Fig. 5: Zoomed-in streamline (top row) and pathline (bottom row) rendering results under different TSR-VFD models using the solar plume data set. 100 streamlines and 1,000 pathlines are traced, respectively.

vector fields. We define the error e_i at the i-th voxel as

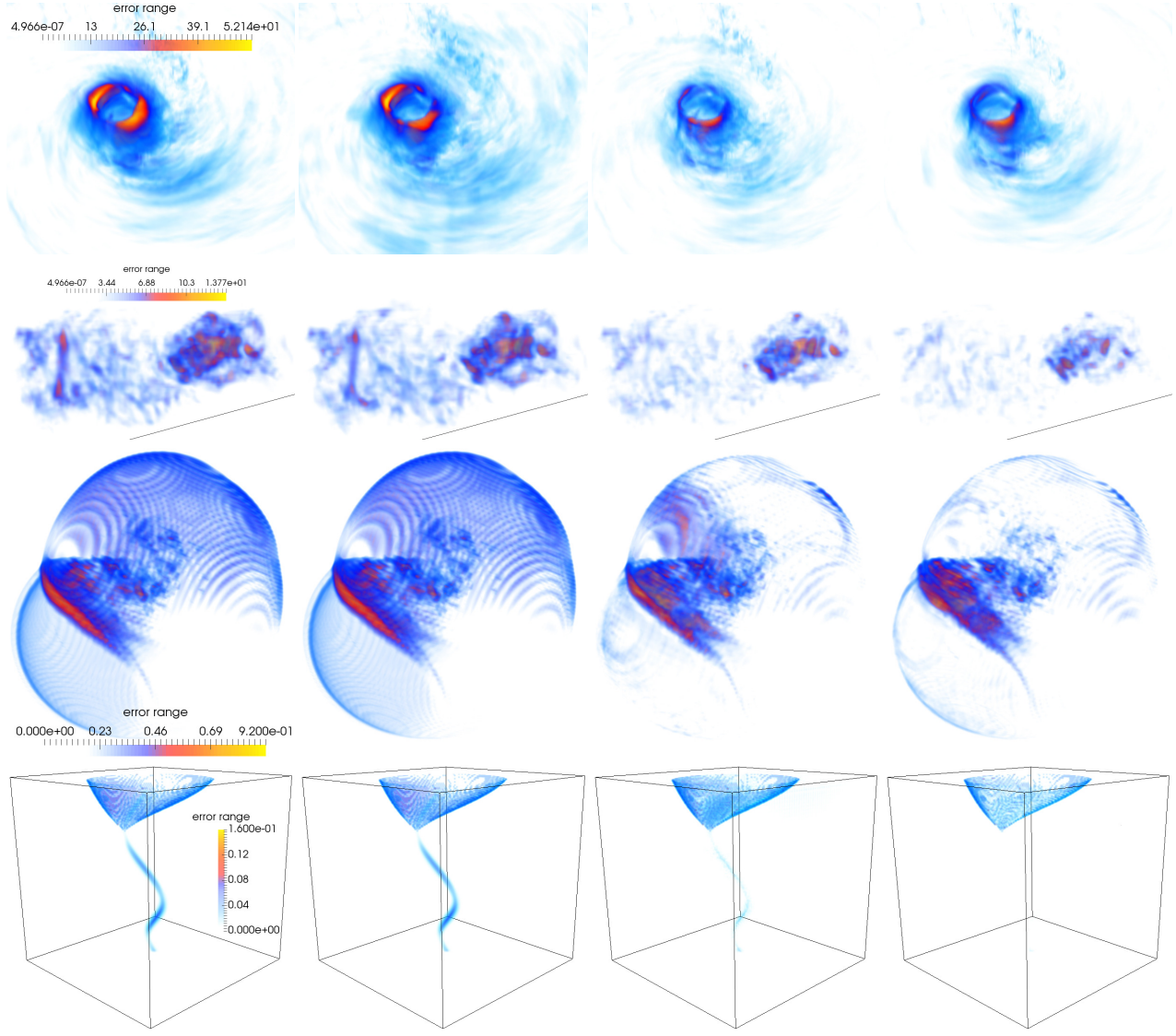
$$e_i(\mathbf{F},\hat{\mathbf{F}}) = \sqrt{\sum_{j\in u,v,w} (\mathbf{F}_{i,j} - \hat{\mathbf{F}}_{i,j})^2}. \quad (1)$$

For the hurricane data set, compared with LERP and RNN, TSR-VFD and GAN introduce fewer errors around the hurricane's eye. For the solar plume and tornado data sets, compared with LERP, RNN, and GAN, TSR-VFD leads to fewer errors in the plume's inner body and at the tornado's vortex core. For the supernova data set, TSR-VFD produces fewer errors at the core and boundary compared with LERP, RNN, and GAN.



(a) PSNR (b) AAD (c) RAE

Fig. 6: PSNR, AAD, and RAE of synthesized vector fields using LERP, GAN, RNN, and TSR-VFD. Top to bottom: hurricane, solar plume, supernova, and tornado. For the supernova and tornado data sets, PSNR, AAD, and RAE values of LERP and RNN are rather close. Therefore, the curves are overlapped.



(a) LERP (b) RNN (c) GAN (d) TSR-VFD

Fig. 7: Volume rendering results of errors introduced by the synthesized vector fields. Top to bottom: hurricane, solar plume, supernova, and tornado.