Flow Field Reduction via Reconstructing Vector Data from 3D Streamlines Using Deep Learning

Jun Han[1], Jun Tao[1], Hao Zheng[1], Hanqi Guo[2], Danny Z. Chen[1], and Chaoli Wang[1]

[1]University of Notre Dame
[2]Argonne National Laboratory

Abstract

We present a new approach for streamline-based flow field representation and reduction. Our method can work in the in situ visualization setting by tracing streamlines from each time step of the simulation and storing compressed streamlines for post hoc analysis where users can afford longer reconstruction time for higher reconstruction quality using decompressed streamlines. At the heart of our approach is a deep learning method for vector field reconstruction that takes the streamlines traced from the original vector fields as input and applies a two-stage process to reconstruct high-quality vector fields. To demonstrate the effectiveness of our approach, we show qualitative and quantitative results with several data sets and compare our method against the de facto method of gradient vector flow in terms of speed and quality tradeoff.

Keywords

Flow visualization, data reduction, vector field reconstruction, deep learning.

1 Introduction

Understanding fluid dynamics is critical to many fields of scientific research ranging from aerodynamics to oceanography. To meet this need, flow visualization has been a central topic of scientific visualization over the past three decades. Integral flow lines such as streamlines intuitively demonstrate the flow patterns, and therefore, become the most widely used visual means to represent and visualize flow fields. Most of the existing works focused on how to produce informative streamlines from flow fields via seed placement or streamline selection. However, the *inverse* problem of reconstructing flow fields as accurate as possible from streamlines is also intriguing but much less explored. Previous works [1, 17] reconstructed the vector field from a set of streamlines and compared the reconstructed vector field against the original one to evaluate the reconstruction quality. These works only applied *linear* reconstruction techniques, which are simple but often good enough to serve the purpose.

In this paper, we target a more practical and challenging application of flow field representation and reduction which includes vector field reconstruction as a key step. Using streamlines for such a purpose is handy, especially in the scenario of in situ visualization. In this scenario, a large number of time steps are generated from the simulation but only a small number of time steps are stored to disk for space efficiency. Meanwhile, during the simulation, streamlines are often produced to illustrate the simulation results for visual debugging and steering. Therefore, the overhead of flow field reduction using the already traced streamlines is minimum: we only need to save the streamlines in a compressed form to disk *in situ* so that the underlying vector fields can be recovered with acceptable fidelity for *post hoc* analysis. Saving compressed streamlines rather than compressed vector fields is advantageous due to the relative sparsity of streamlines and the existence of effective ways to compress streamlines as sequential data. As we will show in the results section, using lossy compression, compressing streamlines is more reliable than compressing vector fields in terms of quality preservation.

Although handy, using streamlines for data reduction also poses great challenges to vector field reconstruction: the reconstructed field has to be accurate within a certain error bound so that flow features and properties can be reasonably preserved. Achieving this goal brings up two major challenges. The first challenge comes from the uneven distribution of streamline samples: even if the given streamlines pass the neighborhood of a voxel, the streamlines may not evenly distribute around that voxel. Therefore, simple strategies that estimate the velocity at that voxel (e.g., computing the weighted average of velocities from the neighboring streamlines) will likely produce a biased result. The second challenge is that the sample streamlines should maintain a good coverage over the reconstructed field. While reconstructing a high-resolution field is necessary to preserve small-scale features, it requires a large number of streamlines (low reduction rate) to cover most of the voxels. Otherwise, the reconstructed velocities at uncovered voxels cannot be verified using the given streamlines, which will lead to unpredicted results.

To address the aforementioned challenges, we propose a two-stage deep learning framework to reconstruct the high-resolution vector field using only a reasonable number of streamlines. The main idea of our approach is as follows. First, instead of interpolating the velocities at voxels based on the ones along sample points on streamlines [1, 17],

we adopt a different strategy by evaluating whether the reconstructed field produces exactly the same streamlines as the given ones. This strategy is not sensitive to the distribution of samples in the neighborhood of a voxel, and therefore, avoids biased results. Second, instead of reconstructing a high-resolution vector field directly, we take two stages: the first stage estimates a low-resolution field so that the given streamlines have a reasonable coverage over the voxels; the second stage upscales the low-resolution field using a convolutional neural network to produce a high-resolution one. To demonstrate the effectiveness of our approach, we show qualitative and quantitative results with several data sets. We compare our vector field reconstruction method against the de facto method of gradient vector flow in terms of speed and quality tradeoff. Our results successfully demonstrate an initial step toward in situ flow field reduction and visualization.

2 Related Work

2.1 Vector Field Reconstruction

Reconstructing vector fields from samples (such as streamlines) has been studied since the 1990s. Mussa-Ivaldi et al. [11] presented a method for 2D vector field reconstruction through a least-squares scheme. In their method, the reconstruction is performed by reconstructing the rotational-free and divergence-free parts of the vector field. Xu and Prince [16] introduced the gradient vector flow (GVF), a two-step algorithm for recovering a vector field from a set of streamlines. GVF first estimates the velocities at voxels where the streamlines pass. Then it fills the empty voxels by minimizing the Laplacian over the entire field. Lage et al. [7] proposed an algorithm for 2D vector field reconstruction from sparse samples. They adapted a polynomial locally for each velocity component and then obtained the global approximation by a partition of unity. Chen et al. [1] computed the triangulation of streamline samples. The velocities at samples are estimated using local differences, and the velocities at grid points are linearly interpolated based on the triangulation.

Our approach differs from the previous work in three aspects. First, instead of linearly combining the samples [1, 16], we apply machine learning techniques to adjust a randomly generated vector field, so that it produces the same velocities in the vector field as those on sample points along the given streamlines. Second, by reconstructing low-resolution and high-resolution vector fields in order, our approach captures both large-scale and small-scale features. Third, to the best of our knowledge, our approach is the first one that leverages the power of deep learning for vector field reconstruction.

2.2 Scientific Data Reduction

Scientific data reduction is a focused research area in both HPC and visualization communities. A comprehensive review of scientific data reduction can be found in [9]. To the best of our knowledge, vector field reduction has not been specially studied. For lossy compression of scalar fields, Lindstrom [10] designed ZFP compression, a transform-based compressor which transforms the original data into another space through a customized orthogonal transform, keeping the difference between the original and decompressed data small under a given error bound. Di and Cappello [2] proposed SZ compression, a prediction-based compressor which fits the data with the best fit selection of curve fitting methods. If the data can be predicted under a user-specific error bound, then it will be replaced by a binary code of the corresponding curve-fitting model. Otherwise, a binary representation is stored to indicate the data.

2.3 CNNs in Image Super-Resolution and Completion

Convolutional neural networks (CNNs) based image super-resolution (SR) algorithms have shown a great success. Wang et al. [15] encoded a sparse representation as their feed-forward network input based on the learned iterative shrinkage and thresholding algorithm. Dong et al. [3] used bicubic interpolations to upscale an input image and trained a three-layer deep fully convolutional network (FCN) to achieve the state-of-the-art SR performance. Ledig et al. [8] utilized a bicubic kernel to downscale an image and trained a generative adversarial network (GAN) to infer photorealistic natural images for an upscaling factor of $4\times$ along each dimension.

CNNs have also been utilized for image completing tasks. Iizuka et al. [5] introduced global and local context discriminators for image completion. The global discriminator looks at the entire image to assess if it is coherent as a whole, while the local discriminator only looks at a small area centered at the completed region to ensure the local consistency of the generated patches. Pathak et al. [13] established an encoder-decoder framework to generate the content of an arbitrary image region conditioned on its surrounding. Yang et al. [18] also proposed a multi-scale neural patch synthesis approach based on the joint optimization of image content and texture constraints.

Our approach combines the ideas of SR and completion. We use a single network to simultaneously upscale a low-resolution vector field and fill the gaps between the input streamlines. Unlike the previous SR and completion

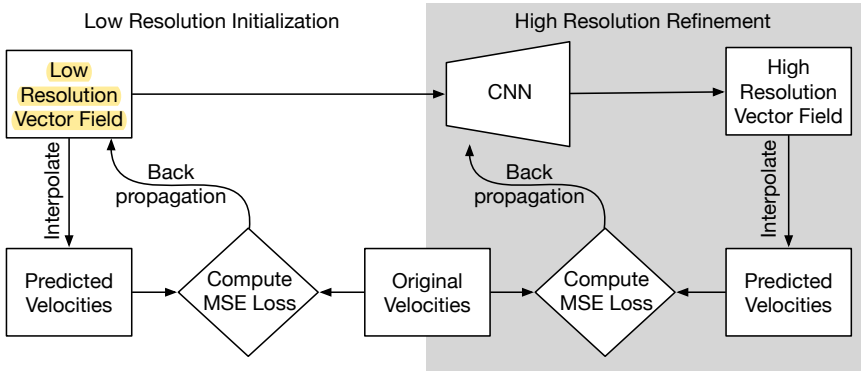


Figure 1: The pipeline of our two-stage vector field reconstruction.

tasks, where deep neural net is designed for a single task, we design a multi-task CNN using residual blocks combined with a velocity-based loss that encourages the generation of high-fidelity vector fields.

3 Approach

3.1 Notations

Let us denote $S = \{s_1, s_2, \cdots, s_n\}$ as a set of streamlines, where $s_i = \{p_1, p_2, \cdots, p_{k_i}\}$ is a set of points on streamline s_i, $\mathbf{v}_p$ and $\mathbf{v'}_p$ are, respectively, the original and predicted velocities at point p. We denote $\mathbf{V}$ as the original vector field with size $L \times W \times H$, where L, W, and H stand for length, width, and height, respectively. We also define $\mathbf{V}_{\text{low}}$ and $\mathbf{V}_{\text{high}}$, which will be used for vector field reconstruction. $\mathbf{V}_{\text{low}}$ is a low-resolution reconstructed vector field with size $L_{\text{low}} \times W_{\text{low}} \times H_{\text{low}}$, where $L_{\text{low}} < L$, $W_{\text{low}} < W$, and $H_{\text{low}} < H$. $\mathbf{V}_{\text{high}}$ is a high-resolution reconstructed vector field with size $L_{\text{high}} \times W_{\text{high}} \times H_{\text{high}}$, where $L_{\text{high}} = L$, $W_{\text{high}} = W$, and $H_{\text{high}} = H$. $\mathbf{W}$ are the total parameters in the CNN model.

3.2 Streamline Selection and Compression

To achieve a good balance between the number of streamlines used for vector field reduction and the quality of vector field reconstruction, we randomly trace a large number of streamlines and from which we select a subset as the representatives. In this paper, we select streamlines using three different methods of Tao et al. [14]: $p(s)$, $I(s;V)$, and REP. The selected streamlines are compressed and later used to reconstruct the vector field via deep learning. Given one streamline $s_i = \{p_1, p_2, \cdots, p_{k_i}\}$, we compress it to an array $\{b, \rho\}$ in the binary form and later decompress $\{b, \rho\}$ to $s'_i = \{p'_1, p'_2, \cdots, p'_{k_i}\}$ via SZ compression [2], where $b[i]$ denotes whether p_i can be predicted by either of the three fitting schemes (preceding neighbor fitting, linear-curve fitting, or quadratic-curve fitting), and $\rho[i]$ denotes the binary form of p_i if p_i cannot be predicted by either fitting scheme. The goal of compression is to control the overall error while achieving a good data reduction rate. Compared to ZFP compression [10], SZ compression can achieve a better compression rate with the same level of peak signal-to-noise ratio.

To measure the error introduced during SZ compression, we compute the difference between s_i and s'_i. Since we only record discrete sample points along a streamline, we measure the displacements of these sample points. We define the error of compressing one streamline as

$$\varepsilon(s_i) = \max_{1 \le j \le k_i} ||p_j - p'_j||, \tag{1}$$

where $||\cdot||$ denotes the distance between the two sample points, and $\varepsilon(s_i)$ should meet the following condition

$$\varepsilon(s_i) \le \big(\max_{1 \le j \le k_i} (p_{j_x}, p_{j_y}, p_{j_z}) - \min_{1 \le j \le k_i} (p_{j_x}, p_{j_y}, p_{j_z}) \big) \times \varepsilon, \tag{2}$$

where p_{j_x}, p_{j_y}, and p_{j_z} are the x, y, and z components of p_j, and ε is the user-specified error bound. In the results section, we specify different error bounds to study the tradeoff between compression rate and reconstruction quality.

3.3 Vector Field Reconstruction

Overview. As shown in Figure 1, we reconstruct a vector field based on a set of streamlines in two stages: *low-resolution initialization* and *high-resolution refinement*. The first stage adopts backpropagation to produce a low-resolution vector field $\mathbf{V}_{\text{low}}$ that approximates the overall flow patterns of the given set of streamlines. The second

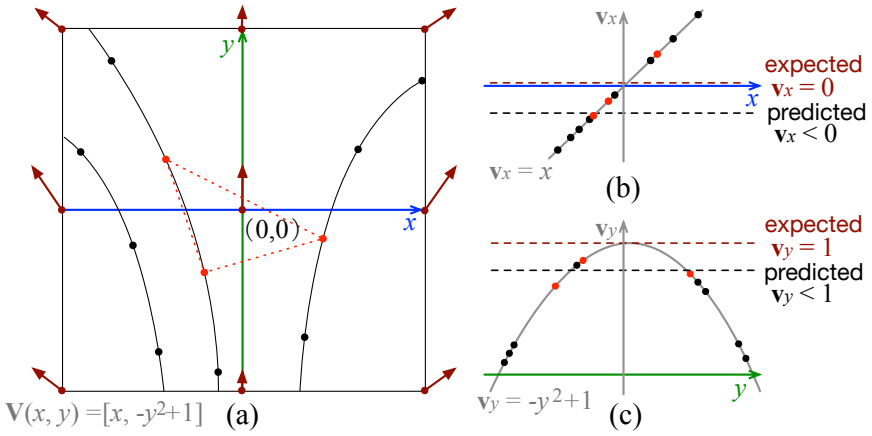


Figure 2: The distribution of samples leads to inaccurate prediction using existing methods. (a) Original vector field and sample streamlines. (b) and (c) The sample points together with $\mathbf{v}_x$ along the x-axis, and $\mathbf{v}_y$ along the y-axis, respectively.

stage leverages a CNN to upscale $\mathbf{V}_{\text{low}}$ to a high-resolution one $\mathbf{V}_{\text{high}}$ which preserves the velocity information for voxels intersected by streamlines while filling the empty voxels based on the low-resolution field.

The rationale for the two-stage design is as follows. The first stage aims to circumvent the problem of insufficient samples: when reconstructing a high-resolution field directly, the given streamlines only cover a small portion of the voxels, leaving most of the voxels empty. Since the reconstructed vectors at the empty voxels cannot be evaluated using the given streamlines, this problem will lead to unexpected reconstruction results. By reconstructing a low-resolution field, we maintain a reasonable coverage of voxels, which produces more reliable results. Then, the second stage upscales the low-resolution field and adds extra, previously-uncaptured flow details via deep neural net. Refer to Figure 4 (d) and (f) for an example.

The core idea of our approach is that, instead of combining the velocities sampled on the streamlines to construct a vector field, we verify and adjust our reconstructed field based on the input streamlines. This strategy can estimate the vectors accurately, regardless of the spatial distribution of sample points. Figure 2 illustrates an example of predicting the vector at the central grid point $(0, 0)$ in a simple vector field $\mathbf{V}(x, y) = [x, -y^2 + 1]$. The existing techniques use linear combination of samples for prediction (e.g., interpolating the vectors from triangulation [1] and averaging the samples [16]), which will lead to inaccurate results. Figure 2 (b) plots $\mathbf{v}_x$ along the x-axis. There are more samples on the left side with negative $\mathbf{v}_x$. Both averaging all the sample points in the neighborhood and interpolating the three samples highlighted in red will lead to a negative value. For predicting $\mathbf{v}_y$, even if there are similar numbers of samples above and below $y = 0$, averaging and interpolation still fail to produce the accurate result, since all sample points have $\mathbf{v}_y$ values smaller than the expected $\mathbf{v}_y$, as shown in Figure 2 (c). In contrast, our approach does not suffer from this problem: inaccurate predictions will be adjusted since they fail to produce the vectors on the samples, leading to large losses.

Loss function. For both stages, we use the mean squared error (MSE) to evaluate the loss of the reconstructed vector field given the streamline samples. The MSE loss at point p is given by the difference between the original velocity $\mathbf{v}_p$ and predicted velocity $\mathbf{v'}_p$, and the MSE loss of streamline s_i is defined as the summation of MSE losses along all of its sample points

$$\mathcal{L}(s_i) = \frac{1}{k_i} \sum_{j=1}^{k_i} ||\mathbf{v}_{p_j} - \mathbf{v'}_{p_j}||_2, \tag{3}$$

where $|| \cdot ||_2$ is the L^2 norm. Our goal is to minimize the total loss, i.e., $\mathcal{L} = \sum_{i=1}^{n} \mathcal{L}(s_i)$.

Low-resolution initialization. As illustrated on the left side of Figure 1, low-resolution initialization iteratively updates a low-resolution vector field $\mathbf{V}_{\text{low}}$ until it produces the same velocities as the input streamlines. We denote $\mathbf{V}_{\text{low}} = [\mathbf{v'}_i]$ where $\mathbf{v'}_i$ is the predicted velocity at voxel i and $1 \leq i \leq L_{\text{low}} \times H_{\text{low}} \times W_{\text{low}}$. All $\mathbf{v'}_i \in \mathbf{V}_{\text{low}}$ are regarded as learnable parameters. Starting from a randomly initialized $\mathbf{V}_{\text{low}}$, in each epoch, we first interpolate $\mathbf{V}_{\text{low}}$ to obtain the predicted velocities on the input streamlines. Given a point p on an input streamline, we estimate its velocity

$$\mathbf{v'}_p = F(\mathbf{V}_{\text{low}}, p), \tag{4}$$

where F is a function that trilinearly interpolates $\mathbf{v'}_p$ based on the velocities at the eight neighboring voxels of p. Then, we compare the predicted velocities with the input ones to derive the loss of the current $\mathbf{V}_{\text{low}}$. Finally, based

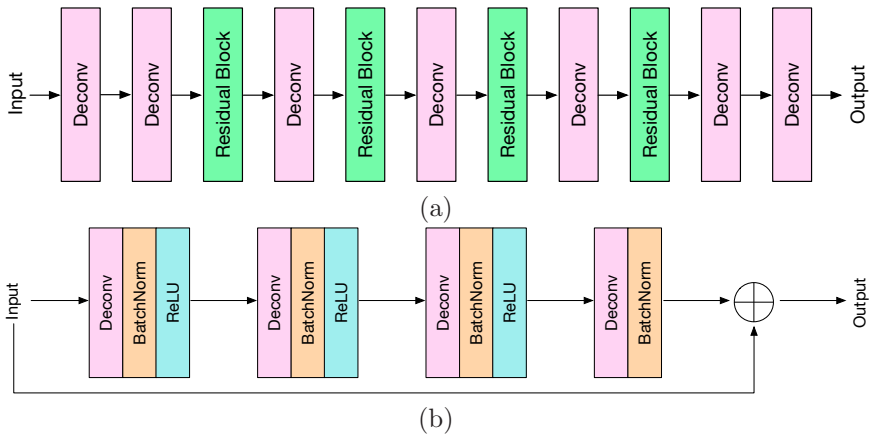


Figure 3: (a) The input to the CNN is a low-resolution vector field, consisting of seven DeConv layers and four residual blocks. (b) An example of a residual block.

Table 1: The dimension of each data set and its corresponding kernel size.

Data Set	Original Dimension	Low-Resolution Dimension	Kernel Size, Stride (DeCov1-DeCov3)	Kernel Size, Stride (DeCov4)	Kernel Size, Stride (DeCov5-DeCov7)
cloud	$128 \times 128 \times 128$	$16 \times 16 \times 16$	$64/128/256 \times 4 \times 4 \times 4$, 1	$128 \times 4 \times 4 \times 4$, 5	$64/64/64 \times 3 \times 3 \times 3$, 1
five critical pts	$51 \times 51 \times 51$	$16 \times 16 \times 16$	$64/128/256 \times 3 \times 3 \times 3$, 1	$128 \times 3 \times 3 \times 3$, 2	$64/64/64 \times 3 \times 3 \times 3$, 1
supercurrent	$256 \times 128 \times 32$	$32 \times 16 \times 4$	$64/128/256 \times 4 \times 2 \times 1$, 1	$128 \times 4 \times 2 \times 2$, 6	$64/64/64 \times 5 \times 7 \times 5$, 1
supernova	$100 \times 100 \times 100$	$12 \times 12 \times 12$	$64/128/256 \times 3 \times 3 \times 3$, 1	$128 \times 3 \times 3 \times 3$, 5	$64/64/64 \times 5 \times 5 \times 5$, 1

on the loss, we update $\mathbf{V}_{\text{low}}$ through backpropagation. Gradient descent is applied to update these velocities, which means that in each epoch, $\mathbf{v'}_i$ is updated based on the following rule

$$\mathbf{v'}_i = \mathbf{v'}_i - \alpha \frac{\partial \mathcal{L}}{\partial \mathbf{v'}_i}, \tag{5}$$

where α is the learning rate.

High-resolution refinement. As shown on the right side of Figure 1, high-resolution refinement builds a CNN to upscale $\mathbf{V}_{\text{low}}$, and at the same time, fills the empty voxels through a nonlinear combination of their neighborhoods. The input to this network is $\mathbf{V}_{\text{low}}$ which is obtained by low-resolution initialization and the output is $\mathbf{V}_{\text{high}}$. We design seven deconvolutional (DeConv) layers and four residual blocks [4] in the network. We also apply the rectified linear unit (ReLU) [12] at each DeConv layer except the last DeConv layer. The detailed architecture is shown in Figure 3 (a). An example of a residual block is shown in Figure 3 (b). It includes three DeConv layers with batch normalization layers and the ReLU added in between, and one DeConv layer only followed by a batch normalization layer. The reason for adding the residual block is that it can prevent CNN from overfitting and gradient vanishing. In this process, we first use the CNN to generate $\mathbf{V}_{\text{high}}$, and then estimate velocities of the given streamlines and compute the MSE between the predicted and original velocities to measure the accuracy of $\mathbf{V}_{\text{high}}$.

Training process. Given a low-resolution vector field $\mathbf{V}_{\text{low}}$ and a set of streamlines S, we first upscale $\mathbf{V}_{\text{low}}$ through CNN to generate $\mathbf{V}_{\text{high}}$. Then given a point p from S, we can estimate its velocity

$$\mathbf{v}_p^{'} = F(\mathbf{V}_{\text{high}}, p), \tag{6}$$

where F is a function that trilinearly interpolates $\mathbf{v}_p^e$ based on the velocities at the eight neighboring voxels of p. Then, we compare the difference between the predicted and ground truth velocities using Equation 3. Finally, we update $\mathbf{W}$ based on the following formula

$$\mathbf{W} = \mathbf{W} - \alpha \frac{\partial \mathcal{L}}{\partial \mathbf{W}}, \tag{7}$$

where α is the learning rate.

Verification of two-stage reconstruction. To verify the effectiveness of our two-stage algorithm, we reconstruct vector fields from the original and low resolutions. We use random and fixed tracing to trace streamlines from these reconstructed vector fields, as shown in Figure 4 (b) to (f). Random tracing means that the seeding points are placed randomly in the entire volume while fixed tracing means that the seeding points are placed only on the input streamlines. Figure 4 (b) and (c) show the streamlines using random and fixed tracing under the original resolution.

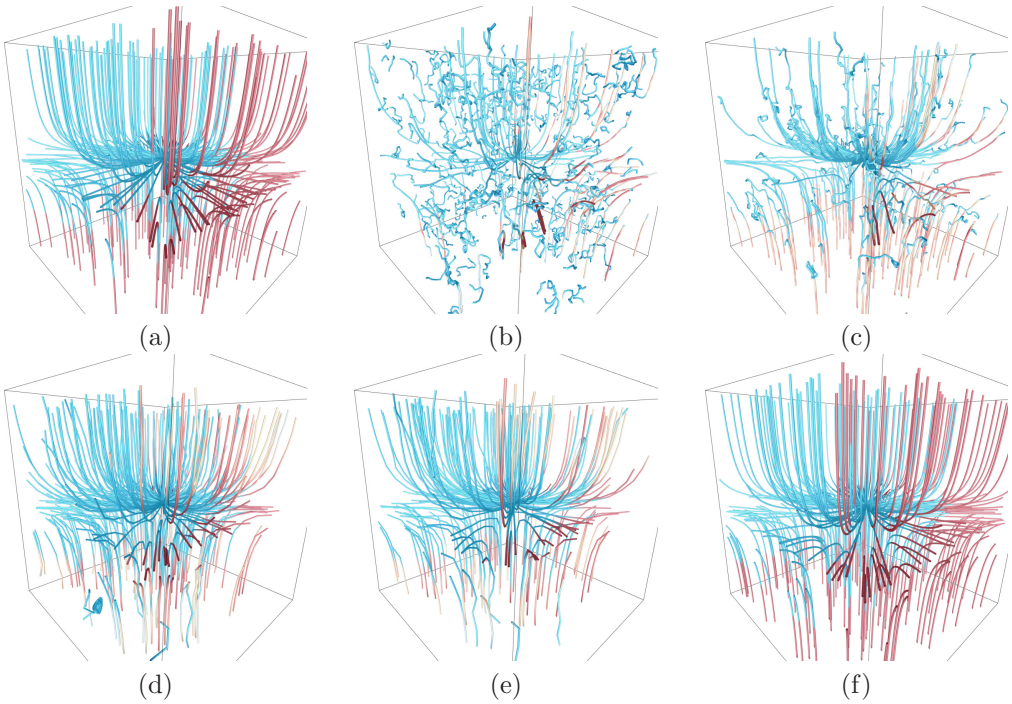


Figure 4: Stage-wise comparison of vector field reconstruction results of the vector field $\mathbf{V}(x, y, z) = [x, -y^2 + 1, z]$. (a) Input streamlines for vector field reconstruction. (b) Original resolution with random tracing at the first stage. (c) Original resolution with fixed tracing at the first stage. (d) Low resolution with random tracing at the first stage. (e) Low resolution with fixed tracing at the first stage. (f) High (original) resolution with random tracing at the second stage. For (b) to (f), 300 streamlines are traced from the reconstructed field.

The results show that the streamlines are not smooth enough in the regions where no samples pass. This is because if no streamline passes certain voxels, then the velocities of these voxels are never updated during the first-stage training process. Figure 4 (d) and (e) show the streamlines using random and fixed tracing under the low resolution. The results show that the entire vector field can be reconstructed well, although domain coverage is not as good as the original one shown in Figure 4 (a). This is due to the errors involved in the low-resolution vector field: some seeding points around the domain boundary cannot be traced to form streamlines. Figure 4 (f) shows randomly traced streamlines after our two-stage algorithm. It is clear that now the traced streamlines visually match well with the input streamlines, in terms of both streamline smoothness and domain coverage.

4 Results and Discussion

4.1 Training Details

We tested with our approach using the data sets listed in Table 1. Based on our experience, in a low-resolution vector field, the voxels passed by streamlines should be more than 80% of the total voxels. We follow this criterion to decide the resolution for the low-resolution vector field. The deep-learning-based vector field reconstruction was implemented in PyTorch using one single NVIDIA TITAN Xp 1080 GPU for training. In the first initialization stage, we initialized $\mathbf{V}_{\text{low}}$ using the uniform distribution $\mathbb{U}(-0.1, 0.1)$ and a learning rate of 10^{-3}. We trained 30 epochs. In the second refinement stage, we initialized all weights using $\mathcal{N}(0, 0.01)$ and the learning rate with 10^{-4}. We set the batch size to 128. We trained 100 epochs. The Adam optimizer [6] was applied in these two stages. However, we decayed the learning rate by a factor of 10 after every 40 epochs during high-resolution refinement.

4.2 Quality of Streamline Sets

We used the proposed vector field reconstruction method to evaluate the quality of selected streamlines in representing the underlying flow field. From a pool of 3000 randomly traced streamlines, we selected 100, 200, and 300 streamline samples using three different methods of Tao et al. [14]: $p(s)$, $I(s; V)$, and REP. The selected streamlines are the training data to reconstruct the vector field. The results of 500 streamlines randomly traced from the reconstructed field are shown in Figure 5 where obvious reconstruction errors are highlighted in black ellipses. We found that when the number of selected streamlines is small, REP achieves the best performance since the streamlines selected by REP are representative of the entire domain and can therefore cover different parts of the domain equally well. In contrast, the streamlines selected by $p(s)$ and $I(s; V)$ mostly favor the parts where the flow field is complex (e.g., the

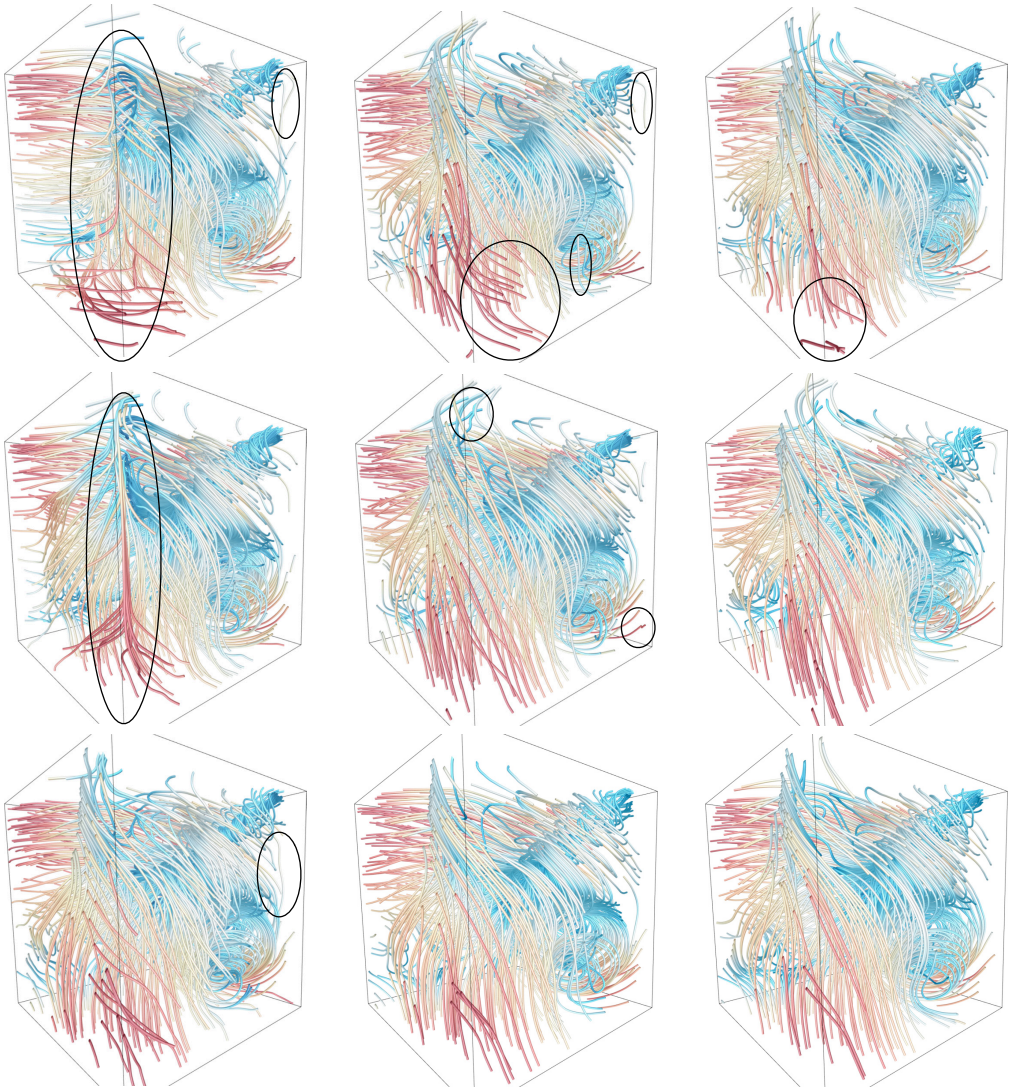


Figure 5: Comparison of vector field reconstruction results of the five critical points data set under different streamline selection methods. The reconstruction is based on streamline selection results from the same pool of 3000 streamlines using three different methods of Tao et al. [14]: $p(s)$ (top row), $I(s;V)$ (middle row), and REP (bottom row). 100, 200, and 300 streamlines are selected for (a), (b), and (c), respectively. 500 streamlines are randomly traced from the reconstructed field.

vicinity of critical points), which leads to the difficulty of estimating vectors in under-sampled parts. However, as the selected streamline samples increase, the quality gaps among these three methods shrink and all three methods can select streamlines that well represent the underlying flow field. Therefore, we choose REP as our streamline selection method for the following experiments due to its superior performance in representing the flow field under a small number of selected streamlines.

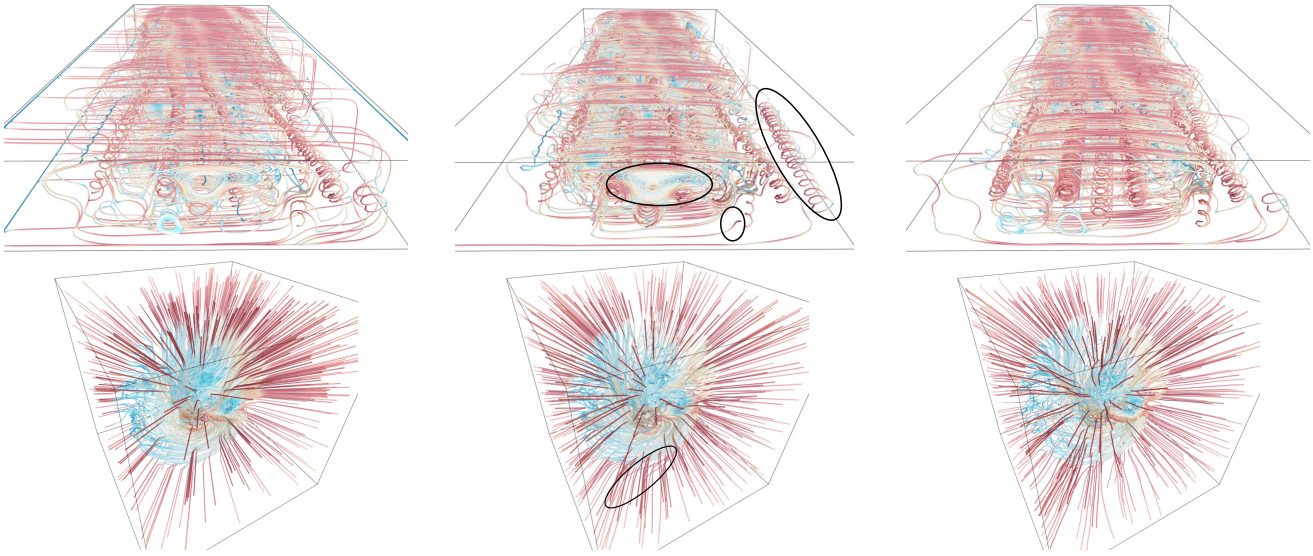
4.3 Vector Field Reconstruction and Reduction

Evaluation metric. We use the peak signal-to-noise ratio (PSNR) to evaluate the quality of reconstructed vector field. PSNR is defined as

$$\mathrm{PSNR}(\mathbf{V},\mathbf{V}') = 20\log_{10} I(\mathbf{V}) - 10\log_{10}\mathrm{MSE}(\mathbf{V},\mathbf{V}'), \tag{8}$$

where $\mathbf{V}$ and $\mathbf{V}'$ are, respectively, the original and reconstructed vector fields; $I(\mathbf{V})$ is the difference between the maximum and minimum values of $\mathbf{V}$

$$I(\mathbf{V}) = \max(\mathbf{V}_x,\mathbf{V}_y,\mathbf{V}_z) - \min(\mathbf{V}_x,\mathbf{V}_y,\mathbf{V}_z), \tag{9}$$



(a) (b) (c)

Figure 6: Comparison of vector field reconstruction results of supercurrent (top row) and supernova (bottom row) data sets under two different methods. (a) input streamlines for vector field reconstruction (supercurrent: 300 streamlines, supernova: 800 streamlines). (b) GVF with fixed tracing (supercurrent: PSNR = 23.58 db, supernova: PSNR = 19.44 db). (c) our method with fixed tracing (supercurrent: PSNR = 34.20 db, supernova: PSNR = 30.05 db). 150 and 500 streamlines are traced from the reconstructed field for supercurrent and supernova, respectively.

where $\mathbf{V}_x, \mathbf{V}_y, \mathbf{V}_z$ are the velocities at components x, y, and z in $\mathbf{V}$; and $\text{MSE}(\mathbf{V}, \mathbf{V}')$ is the mean squared error between $\mathbf{V}$ and $\mathbf{V}'$, namely

$$\text{MSE}(\mathbf{V}, \mathbf{V}') = \frac{1}{3 \times L \times W \times H} \sum_{i=1}^{L \times W \times H} \sum_{j \in x,y,z} (\mathbf{V}[i]_j - \mathbf{V}'[i]_j)^2, \tag{10}$$

where $\mathbf{V}[i]$ and $\mathbf{V}'[i]$ are, respectively, the velocities of $\mathbf{V}$ and $\mathbf{V}'$ at voxel i. The error between $\mathbf{V}[i]$ and $\mathbf{V}'[i]$ is defined as

$$\xi(\mathbf{V}[i], \mathbf{V}'[i]]) = \sqrt{\sum_{j \in x,y,z} (\mathbf{V}[i]_j - \mathbf{V}'[i]_j)^2}. \tag{11}$$

Let s_V be the space for storing the original, uncompressed vector field, and s_L and $s_{L'}$ be the space for storing the uncompressed and SZ-compressed streamlines and their corresponding velocities, respectively. We define the reduction rate as $\text{RR} = s_V/s_L$ and compression rate as $\text{CR} = s_V/s_{L'}$.

Qualitative and quantitative analysis. We qualitatively and quantitatively compare our method against GVF [16]. The baseline method of GVF is a linear model for vector field reconstruction. First, we compare visual results in terms of streamlines traced from reconstructed vector fields and errors introduced in vector field reconstruction. In Figure 6, we can see that our method achieves a better reconstruction quality than GVF: it generates streamlines that are closer to the input streamlines (obvious reconstruction errors made by GVF are highlighted in black ellipses) while yields a much higher PSNR. In Figure 7, we can see that the vector field reconstruction errors (as defined in Equation 11) introduced by our method are smaller than those introduced by GVF. The visual differences for the supercurrent and supernova data sets are greater than that of the cloud data set as the PSNR differences get larger.

Second, we compare the performance with different numbers of selected streamlines as input. Our approach outperforms GVF in all cases, except for the cloud data set using 100 streamlines, as shown in the left part of Table 2. We find that GVF achieves a similar PSNR when the flow field is simple (e.g., for the synthesized five critical points data set). However, our approach shows significant improvement when the flow is relatively complex. The PSNR difference could reach 10 in the supercurrent data set using 300 samples and in the supernova data set using 800 samples. In these data sets, GVF does not benefit from extra streamlines, but our approach is able to incorporate additional information. In the supernova data set, when the number of samples increases from 100 to 800, the PSNR of our approach improves from 19.58 to 30.05, but that of GVF only raises from 16.25 to 19.44. We believe that GVF fails to estimate the complex non-linear flow patterns using a linear technique (minimizing the Laplacian). In terms of reduction rate (without considering compression for vector fields and selected streamlines),

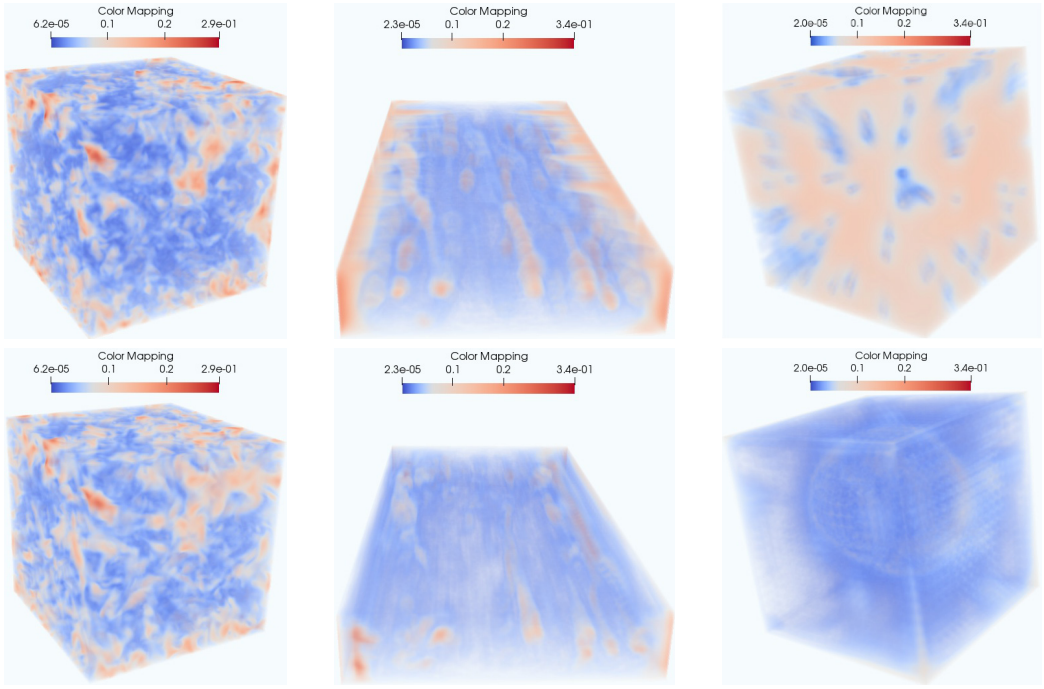


Figure 7: Visualization of errors introduced in vector field reconstruction using (a) 800 streamlines for cloud, (b) 300 streamlines for supercurrent, and (c) 800 streamlines for supernova. Top row: GVF (cloud: PSNR = 26.96 db, supercurrent: PSNR = 23.58 db, supernova: PSNR = 19.44 db). Bottom row: Our method (cloud: PSNR = 30.26 db, supercurrent: PSNR = 34.20 db, supernova: PSNR = 30.05 db).

storing uncompressed streamlines instead of the original vector field reduces the data by 3.16× to 82.76×. Among the four data sets, the lowest and highest reduction rates are achieved for the supercurrent and cloud data sets, respectively.

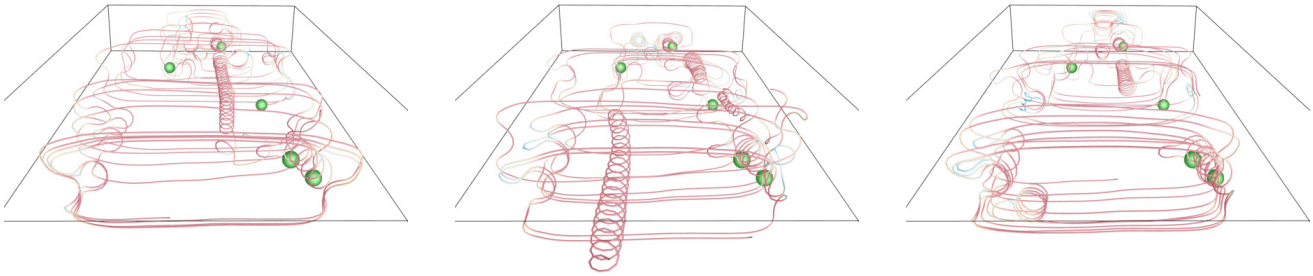
Third, we compare the performance under different error bounds. From the right part of Table 2, we can see that adding SZ compression would bring us an impressive compression rate ranging from 25.17× to 572×. Among the four data sets, the lowest and highest reduction rates are achieved for the five critical points and supernova data sets, respectively. Clearly, the larger the error bound, the higher the compression rate and the lower the PSNR. We also observe that the PSNR difference under error bounds 0.001 and 0.01 is small, while that under error bounds 0.01 and 0.1 is much larger. This is because under $\varepsilon = 0.1$, SZ aggressively compresses several points and their velocities into a single one, which leads to a much lower PSNR. As an exception, when applying GVF to the supernova data set, the PSNR under $\varepsilon = 0.01$ is larger than that under $\varepsilon = 0.001$. This is possibly because that GVF is sensitive to the distribution of samples. Still, our method beats GVF in terms of PSNR under different error bounds, by a large margin for the supernova data set and a small margin for the five critical points data set. This may due to the fact that, unlike the five critical points data set, there exist abrupt changes of velocities among different regions in the supernova data set. This causes unreliable estimation of these changes through minimizing the Laplacian by GVF.

4.4 Discussion

Why not compress vector field directly? We use SZ compression to compress selected streamlines to representing the underlying flow field in a reduced form. An alternative is to compress the original vector field directly using SZ which compresses each vector component separately. In Figure 8, we compare (a) the ground truth streamlines traced from the original vector field, (b) the streamlines traced from the SZ compressed then decompressed vector field, and (c) the streamlines traced from the reconstructed vector field using our method. Note that for a fair comparison, both (b) and (c) take similar compression rates (even though (b) achieves a higher PSNR than (c)). The five seeding points are randomly placed, not on the selected streamlines as used in our approach to reconstruct the vector field. The result shows that compared to our method, the streamlines traced from the directly compressed then decompressed vector field are less similar to the ground truth streamlines. This renders direct SZ compression of vector field less reliable than our method. The reason is that direct compression of vector field only controls the error of vector field, not the resulting streamlines. Moreover, small changes in the vector field during the direct compression-decompression process could lead to large differences in the streamlines.

Table 2: Comparison of reduction/compression rate and reconstruction PSNR for each data set. Left: comparison across different numbers of streamlines. Right: comparison across different error bounds.

					PSNR (db)						PSNR (db)	
Data Set	s_V (MB)	# Lines	s_L (MB)	RR	GVF	Ours	# Lines	ε	$s_{L'}$ (MB)	CR	GVF	Ours
cloud	24	100	0.29	82.76	22.07	22.04	800	0.001	0.37	64.86	26.95	30.02
		500	1.4	17.14	25.78	27.56		0.01	0.17	141.18	26.42	29.44
		800	2.2	10.91	26.96	30.26		0.1	0.05	480	19.65	21.02
five critical pts	1.51	100	0.13	11.62	23.03	25.19	300	0.001	0.06	25.17	30.62	30.94
		200	0.23	6.57	28.02	29.19		0.01	0.02	75.5	30.60	30.71
		300	0.39	3.87	30.62	31.33		0.1	0.01	151	17.10	18.50
supercurrent	12	100	1.3	9.23	20.86	26.41	300	0.001	0.37	32.43	23.56	32.57
		200	2.6	4.62	22.39	30.01		0.01	0.15	80	20.83	30.14
		300	3.8	3.16	23.58	34.20		0.1	0.04	300	17.38	23.46
supernova	11.44	100	0.15	76.27	16.25	19.58	800	0.001	0.18	63.55	19.44	29.31
		500	0.67	17.07	18.34	26.51		0.01	0.08	143	19.54	27.82
		800	1.1	10.4	19.44	30.05		0.1	0.02	572	15.59	18.34



(a) (b) (c)

Figure 8: Streamline visualization of the supercurrent data set. (a) Ground truth streamlines. (b) Streamlines traced from the SZ compressed then decompressed vector field ($\varepsilon = 0.03$, CR $= 32.17$, PSNR $= 41.14$ db). (c) Streamlines traced from the reconstructed vector field using our method with 300 input streamlines ($\varepsilon = 0.001$, CR $= 32.43$, PSNR $= 32.57$ db). All show five streamlines traced from the same seeding points (denoted as green spheres).

Toward in situ flow field reduction. We need to consider the following in order to make our current solution practical for in situ flow field reduction. First, there are several key parameters to set (e.g., the number of input streamlines, the low-resolution flow field size, and the error bound for compression). Based on our results, we recommend setting these parameters according to the complexity exhibited by the flow field or the selected streamlines. More streamlines and lower error bounds should be used for more complex data sets. Second, deep learning requires long training time (in our cases, hours on a single GPU) even though this happens during post hoc analysis. We need to accelerate the training speed by speeding up low-resolution initialization using multiple devices and high-resolution refinement using model parallelism. Third, nontrivial extensions are needed for handling large unsteady flow fields (e.g., 1000^3) with hundreds of time steps. Efficient out-of-core computation for large data sets and learning across time steps should be considered.

5 Conclusion

As an emerging research direction, applying deep learning techniques to solve challenging scientific visualization problems is on the rise. We have presented a new approach for streamline-based flow field representation and reduction. The main algorithm lies in the deep-learning-based algorithm for vector field reconstruction. Inspired by recent advances in image super-resolution and completion, our approach enables high-quality reconstruction of vector fields from streamlines, which outperforms the de facto method of GVF for vector field reconstruction. Along with SZ compression, we achieve impressive data compression rates while keeping PSNRs higher than those obtained using GVF. This work is our first step toward deep learning for in situ flow field reduction and visualization. Our next step is to leverage multiple GPUs and model parallelism for acceleration, and to implement out-of-core computation and efficient learning for large unsteady flow fields.

Acknowledgements

This research was supported in part by the U.S. National Science Foundation through grants IIS-1455886, CCF-1617735, CNS-1629914, and DUE-1833129, U.S. Department of Energy under contract number DE-AC02-06CH11357,

and the NVIDIA GPU Grant Program. The authors would like to thank the anonymous reviewers for their insightful comments.

References

[1] Y. Chen, J. Cohen, and J. Krolik. Similarity-guided streamline placement with error evaluation. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1448–1455, 2007.
[2] S. Di and F. Cappello. Fast error-bounded lossy HPC data compression with SZ. In *Proceedings of IEEE International Symposium on Parallel and Distributed Processing*, pages 730–739, 2016.
[3] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, 2016.
[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.
[5] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics*, 36(4):107:1–107:14, 2017.
[6] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of International Conference for Learning Representations*, 2015.
[7] M. Lage, F. Petronetto, A. Paiva, H. Lopes, T. Lewiner, and G. Tavares. Vector field reconstruction from sparse samples with applications. In *Proceedings of IEEE Conference on Computer Graphics and Image Processing*, pages 297–306, 2006.
[8] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 4681–4690, 2017.
[9] S. Li, N. Marsaglia, C. Garth, J. Woodring, J. Clyne, and H. Childs. Data reduction techniques for simulation, visualization and data analysis. *Computer Graphics Forum*, 2018.
[10] P. Lindstrom. Fixed-rate compressed floating-point arrays. *IEEE Transactions on Visualization and Computer and Graphics*, 20(12):2674–2683, 2014.
[11] F. A. Mussa-Ivaldi. From basis functions to basis fields: Vector field approximation from sparse data. *Biological Cybernetics*, 67(6):479–489, 1992.
[12] V. Nair and G. E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of International Conference on Machine Learning*, pages 807–814, 2010.
[13] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016.
[14] J. Tao, J. Ma, C. Wang, and C.-K. Shene. A unified approach to streamline selection and viewpoint selection for 3D flow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 19(3):393–406, 2013.
[15] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang. Deep networks for image super-resolution with sparse prior. In *Proceedings of International Conference on Computer Vision*, pages 370–378, 2015.
[16] C. Xu and J. L. Prince. Gradient vector flow: A new external force for snakes. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 66–71, 1997.
[17] L. Xu, T.-Y. Lee, and H.-W. Shen. An information-theoretic framework for flow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1216–1224, 2010.
[18] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li. High-resolution image inpainting using multi-scale neural patch synthesis. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 6721–6729, 2017.

Jun Han is a PhD student in the Department of Computer Science & Engineering at University of Notre Dame. Contact him at jhan5@nd.edu.
Jun Tao is a postdoctoral researcher in the Department of Computer Science & Engineering at University of Notre Dame. Contact him at jtao1@nd.edu.
Hao Zheng is a PhD student in the Department of Computer Science & Engineering at University of Notre Dame. Contact him at hzheng3@nd.edu.
Hanqi Guo is an assistant computer scientist in the Mathematics and Computer Science Division at Argonne National Laboratory. Contact him at hguo@anl.gov.
Danny Z. Chen is a professor in the Department of Computer Science & Engineering at University of Notre Dame. Contact him at dchen@nd.edu.
Chaoli Wang is an associate professor in the Department of Computer Science & Engineering at University of Notre Dame. Contact him at chaoli.wang@nd.edu.