STNet: An End-to-End Generative Framework for Synthesizing Spatiotemporal Super-Resolution Volumes

Jun Han, Hao Zheng, Danny Z. Chen *Fellow, IEEE* and Chaoli Wang, *Senior Member, IEEE*

Abstract— We present STNet, an end-to-end generative framework that synthesizes spatiotemporal super-resolution volumes with high fidelity for time-varying data. STNet includes two modules: a generator and a spatiotemporal discriminator. The input to the generator is two low-resolution volumes at both ends, and the output is the intermediate and the two-ending spatiotemporal super-resolution volumes. The spatiotemporal discriminator, leveraging convolutional long short-term memory, accepts a spatiotemporal super-resolution sequence as input and predicts a conditional score for each volume based on its spatial (the volume itself) and temporal (the previous volumes) information. We propose an unsupervised pre-training stage using cycle loss to improve the generalization of STNet. Once trained, STNet can generate spatiotemporal super-resolution volumes from low-resolution ones, offering scientists an option to save data storage (i.e., sparsely sampling the simulation output in both spatial and temporal dimensions). We compare STNet with the baseline bicubic+linear interpolation, two deep learning solutions (SSR+TSR, STD), and a state-of-the-art tensor compression solution (TTHRESH) to show the effectiveness of STNet.

Index Terms—Time-varying data, generative adversarial network, spatiotemporal super-resolution.

1 Introduction

We have witnessed the success of deep learning in scientific visualization tasks, such as volume upscaling [61, 58, 21, 17, 20, 59], data reconstruction [19, 16], variable and ensemble generation [25, 22], rendering image synthesis [27, 2, 11], and representative selection [6, 47, 18]. However, the end-to-end *spatiotemporal super-resolution* (STSR) task for time-varying data is unexplored. That is, given a set of sparsely sampled low-resolution volumes (e.g., $128\times128\times128\times50$), we aim to generate STSR volumes (e.g., $512\times512\times512\times200$). STSR is meaningful for scientific visualization due to its potential applications in data reduction and data recovery.

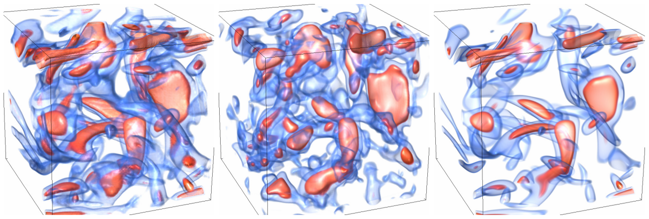

(a) SSR+TSR (b) TSR+SSR (c) GT

Fig. 1: STSR using two solutions: SSR+TSR and TSR+SSR.

A straightforward solution for generating spatiotemporal solution is to train a *spatial super-resolution* (SSR) network (e.g., [20]) and a *temporal super-resolution* (TSR) network (e.g., [21]) in sequence, i.e., SSR+TSR or TSR+SSR. An example is shown in Figure 1. The results show that both solutions do not yield high-quality STSR volumes compared with the ground-truth (GT). The is because the errors in the first stage (e.g., SSR) could accumulate and amplify in the second stage (e.g., TSR). Thus, such sequential solutions may not guarantee to synthesize STSR volumes with high fidelity.

Three challenges remain for the STSR task. First, although SSR and TSR have been studied independently, merely concatenating the two solutions cannot guarantee satisfactory STSR volumes since the rendering quality is far away from GT. Designing an *end-to-end* STSR architecture is critical for avoiding error accumulation and amplification. Second, deep learning's success depends heavily on large input data, which is often challenging to acquire in scientific visualization. The limited training data will prevent the network from a better generalization during inference. How to utilize inadequate data samples to improve the generalization power should be considered in the optimization. Third, the computational cost is high as it usually requires days to train a *generative adversarial network* (GAN) on 3D data. In addition, the synthesized volumes should maintain similar spatial (e.g., structure and texture) and temporal (e.g., close similarity among neighboring time steps) coherence compared with GTs.

To respond, we design **STNet**, an *end-to-end* **s**patio**t**emporal generative **net**work for STSR. STNet encompasses two stages: *pre-training* and *fine-tuning*. During pre-training, STNet accepts all available low-resolution data as input, generates synthesized low-resolution volumes, and applies cycle loss for optimization. During fine-tuning, STNet takes low-resolution data at early time steps as input, produces STSR volumes, and leverages volumetric and adversarial losses for training. Specifically, we first investigate popular framework designs in both SSR and TSR tasks and design an end-to-end spatiotemporal model with post-upsampling for spatial upscaling and feature interpolation for temporal upscaling. That is, STNet interpolates the feature of each low-resolution volume and upscales the features into the data (super-resolution) space. Second, we customize a *pre-training* task for STSR by only leveraging the information from low-resolution volumes. The goal is to explicitly promote a better generalization for producing spatiotemporal volumes for time-varying data. Third, we design a spatiotemporal discriminator to guarantee spatial and temporal coherence of the synthesized spatiotemporal volumes. We apply a two-stage optimization procedure to cut the computational cost and boost the stability of GAN training.

For evaluation, we apply STNet to different time-varying data sets with various characteristics. Volume and isosurface rendering results show that STNet achieves better visual quality than bicubic+linear interpolation, SSR+TSR, STD (a variant of STNet), and TTHRESH [1] (a state-of-the-art tensor compression algorithm). Moreover, qualitative analysis results also confirm the effectiveness of STNet using three metrics at the data, image, and feature levels.

The contributions of STNet are as follows. First, we design STNet, a novel end-to-end deep learning model that applies GANs to simultaneously upscales volumes at both spatial and temporal dimensions. Second, we establish a pre-training algorithm that can improve the network's generalization ability. Third, we perform a comprehensive study to investigate the potential impact factors for STSR.

- *The authors are with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556. E-mail: {jhan5, hzheng3, dchen, chaoli.wang}@nd.edu.*

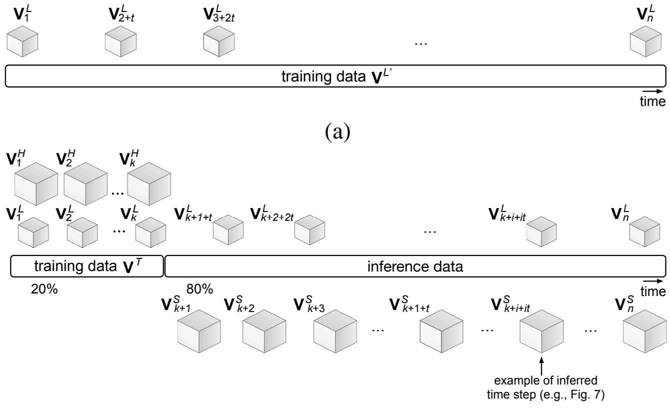
Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx/

2 Related Work

Deep learning for volume visualization. With the noticeable success of deep learning in computer vision, robotics, and NLP, researchers have explored neural networks' possibility in solving volume visualization problems. Such examples include SSR for volume [61, 20] and isosurface [58], TSR for volume [21], variable selection and translation [22], ensemble generation [25], volume rendering [27, 2, 11], and viewpoint estimation [52]. Our work is closely related to SSR-TVD [20] and TSR-TVD [21]. Instead of only focusing on either SSR or TSR, we upscale volumes at both spatial and temporal dimensions *simultaneously* in an *end-to-end* style.

Deep learning for super-resolution. Deep neural networks have been widely applied in SSR,TSR, and STSR tasks. The examples of SSR include SRCNN [10], SRGAN [32], ZSSR [53], SRFBN [33], SRNTT [64], and NatSR [54]. As for TSR, examples are phase-based interpolation [38], DVF [36], SepConv [42], DeepLLE [41], and SloMo [29]. The works of STSR include Xiang et al. [60], Shechtman et al. [51], Mudenagudi et al. [39], Takeda et al. [55], and Shahar et al. [50]. Our work is different from these works in three ways. First, compared with SSR and TSR, we propose a framework with end-to-end training for STSR. Second, unlike these computationally expensive STSR solutions with limited capacity to capture complex spatiotemporal patterns, we build a fast and accurate STSR framework. Third, instead of training from scratch in these works, we design a pre-training algorithm to improve the network's generalization ability.

Network pre-training. Pre-training in deep learning models aims to provide a good parameter initialization for better generalization in a particular task (e.g., classification). Based on different tasks, pre-training examples include inpainting [45], colorization [31], synthesis [9], feature agreement [15], rotation prediction [13], context prediction [8], and feature contrast [23, 5]. Unlike these pre-training approaches, which are tailored for classification, detection, or segmentation, we propose a novel unsupervised pre-training method for STSR using cycle loss [65, 62].



(a)

(b)

Fig. 2: Illustration of STNet's training and inference data at (a) pre-training and (b) fine-tuning stages.

3 STNet

3.1 Notation

Let $\mathbf{V}^L = \{\mathbf{V}_1^L, \mathbf{V}_2^L, \mathbf{V}_3^L, \cdots, \mathbf{V}_n^L\}$ and $\mathbf{V}^H = \{\mathbf{V}_1^H, \mathbf{V}_2^H, \mathbf{V}_3^H, \cdots, \mathbf{V}_n^H\}$ be low-resolution and high-resolution time-varying volumetric sequences, respectively, where n denotes the number of time steps. $\mathbf{V}^{L'} = \{\mathbf{V}_1^L, \mathbf{V}_{2+t}^L, \mathbf{V}_{3+2t}^L, \cdots, \mathbf{V}_n^L\}$ is a sparsely sampled low-resolution time-varying volumetric sequence, where t denotes the number of intermediate time steps and $t' = t+1$ is the *temporal upscaling factor* (i.e., the sequence is upscaled t' times at the temporal dimension). $\mathbf{V}^{L'}$ is also the *pre-training* data in STNet (Figure 2 (a)). $\mathbf{V}^T = \{(\mathbf{V}_1^L, \mathbf{V}_1^H), (\mathbf{V}_2^L, \mathbf{V}_2^H), \cdots, (\mathbf{V}_k^L, \mathbf{V}_k^H)\}$ is a sequence used for *fine-tuning* STNet (Figure 2 (b)), where k is the total training samples. In this paper, we set $k = 0.2n$. $\mathbf{V}^S = \{\mathbf{V}_1^S, \mathbf{V}_2^S, \mathbf{V}_3^S, \cdots, \mathbf{V}_n^S\}$ is a super-resolution time-varying volumetric sequence that we aim to generate via STNet. Namely, $\mathbf{V}^H \approx \mathbf{V}^S = \text{STNet}(\mathbf{V}^{L'})$. s is the *spatial upscaling factor* (i.e., each volume is upscaled s times at each spatial dimension). Note that we purposefully refer to the *synthesized* data as *super-resolution* data and the *original* data as *high-resolution* data for differentiation.

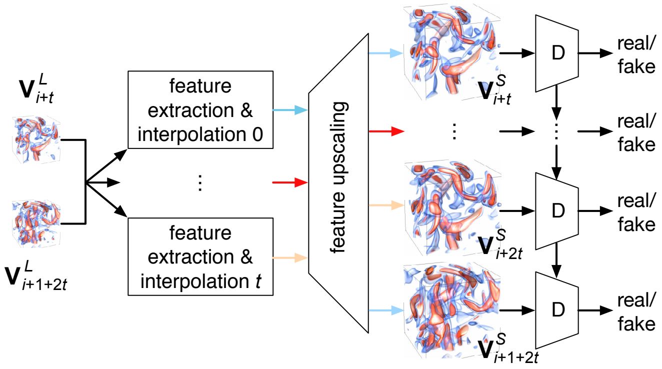


Fig. 3: Overview of STNet. The network consists of several feature extraction and interpolation (FEI) modules for representing spatiotemporal features and one feature upscaling (FU) module for generating super-resolution volumes. After that, a spatiotemporal discriminator is utilized to discern the spatial and temporal realness.

3.2 Overview

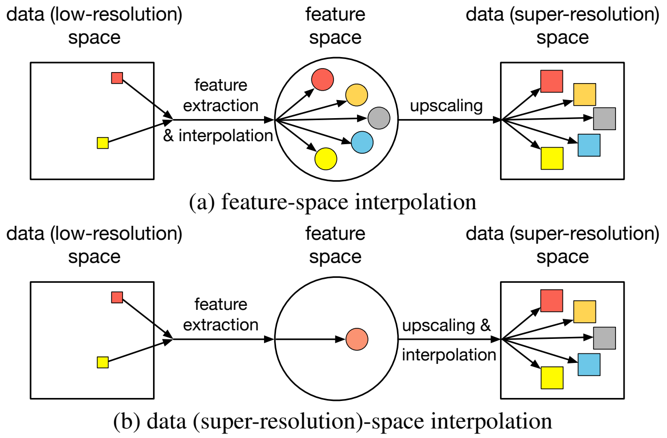
As shown in Figure 3, given two-ending low-resolution volumes $\mathbf{V}_{i+t}^L$ and $\mathbf{V}_{i+1+2t}^L$, STNet first leverages $t+1$ FEI modules to learn features of the intermediate and the two-ending volumes. One FEI module is tailored for representing the two-ending volumes, and additional t modules are for learning the t intermediate volumes. Once the features are learned, a FU module transforms the spatiotemporal features into a high-dimensional space for generating high-fidelity super-resolution volumes. To discern the spatial and temporal realness of these synthesized volumes, we apply a spatiotemporal discriminator (D) based on *convolutional long short-term memory* (ConvLSTM) [21]. D accepts a volume sequence as input and scores each volume's realness through its spatial (the volume itself) and temporal (its previous time steps) information. To optimize STNet, we propose a two-stage *pre-training* and *fine-tuning* algorithm. During pre-training, we only utilize the low-resolution volumes (i.e., $\mathbf{V}^{L'}$) to optimize STNet using cycle loss. This stage aims to furnish a proper parameter initialization for STNet, which can boost its generalization ability. During fine-tuning, we use $\mathbf{V}^T$ as training samples to fine-tune STNet for performance improvement. In the following, we discuss the criteria and rationales for designing spatial and temporal modules. Then, we provide optimization details for the pre-training and fine-tuning stages.

3.3 Framework

STNet follows a post-upsampling architecture for spatial upscaling and performs interpolation in the feature space for temporal upscaling. The rationales are provided as follows.

Why choose post-upsampling for SSR? Considering SSR architectures, the most widely used ones are *pre-sampling* and *post-upsampling* frameworks [57]. Pre-sampling applies common upscaling approaches (e.g., bicubic and trilinear) for upscaling and follows a series of Conv layers to refine the upscaled data. In contrast, post-sampling leverages Convs to represent low-resolution data and upscales the representations to super-resolution data using deconvolutional or shuffle layers. Compared with pre-sampling, post-sampling brings two benefits: *speed* and *performance*. First, since most operations perform in the low-dimensional space and only a few operations occur in the high-dimensional space in post-sampling, the computational cost is low. Second, Convs cannot completely eliminate the noises and artifacts introduced by common upscaling approaches in pre-upsampling, while post-upsampling has no such issue in upscaling because Convs already distill data in the low-dimensional space.

Why perform feature-space temporal interpolation? For TSR architectures, two common options are performing interpolation in the *feature* or *data* (super-resolution) space. The feature space refers to the hidden representations of low-resolution volumes, which CNNs usually extract. The data space refers to the space composed of the original volumes. The examples are sketched in Figure 4. For feature-space interpolation, give two time steps at both ends, we leverage feature extraction and interpolation to generate the feature of each intermediate time step and the two-ending time steps individually. We then use a FU module to generate the super-resolution volumes from these features. For data-space interpolation, a unified representation is learned from all intermediate and the two-ending time steps. Then the feature is upscaled and interpolated in the data (super-resolution) space. Taking into account the involvement of SSR, feature interpolation is a more suitable solution due to the following reason. Applying data-space interpolation requires a powerful FEI module to learn a representation with rich spatiotemporal information for all intermediate and the two-ending volumes. It also demands a powerful FU module to transform one feature into $t+2$ time steps. This would be difficult, especially in a low-dimensional space, since the information in low-resolution data is limited. For feature-space interpolation, the difficulties of extracting spatiotemporal information and the FU module's demanding capability are mitigated through interpolating multiple features.


data (low-resolution) space
feature space
data (super-resolution) space
feature extraction & interpolation
upscaling
data (low-resolution) space
feature space
data (super-resolution) space
feature extraction
upscaling & interpolation

(a) feature-space interpolation

(b) data (super-resolution)-space interpolation

Fig. 4: Illustration of two different temporal interpolation options.

Generator. The core of generator lies in the *feature extraction and interpolation* (FEI) and *feature upscaling* (FU) modules. The FEI module comprises four dense blocks (DBs) [28]. As shown in Figure 5 (a), in each DB, it includes three Conv layers. Each Conv accepts all previous outputs stacked together as input. In particular, we utilize $t+1$ FEI modules to interpolate features of the intermediate and the two-ending volumes. One module accepts the low-resolution volumes as input and produces the corresponding features. The rest of the t modules take the two ending volumes as input and interpolate t features of the intermediate volumes. As sketched in Figure 5 (b), in the FU module, we first separate the input into two branches. In each branch, one *voxel shuffle* (VS) layer [21] is used to upscale the input. Then in the second branch, after VS, a Conv and a sigmoid activation function follow. This result is multiplied by the output from the first branch. After merging, two Conv layers and skip connection [28, 49] are utilized to produce the final output [7, 43]. The motivation of using this two-branch-based FU module is that the network can estimate the importance of each neuron in the feature maps, and the more important neurons will offer a larger weight in the following convolution computation. This design forces the network to pay more attention to interesting volumetric regions instead of treating interesting (e.g., features) and uninteresting (e.g., background) regions equally. We have only a FU module in the generator, which upscales the intermediate features of all time steps. The architecture detail is given in the Appendix. Note that for generating STSR volumes, we need $t+1$ FEI modules: one for representing the two-ending volumes and t for the t intermediate volumes. Rectified linear unit (ReLU) [40] is applied after each Conv layer except the final output layer. No activation function follows after the output layer. Adding tanh or sigmoid will significantly hurt the performance for some specific data sets (e.g., supercurrent) since tanh will saturate at the tails of -1 and 1 and sigmoid will saturate at the tails of 0 and 1, which could kill the gradient and prevent the network from continuous learning.

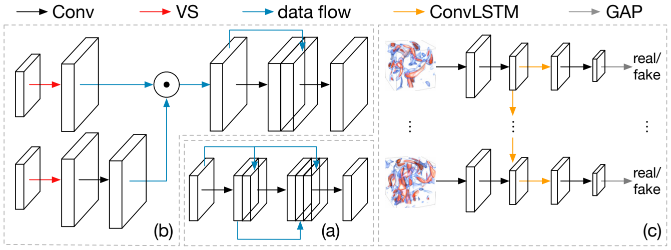
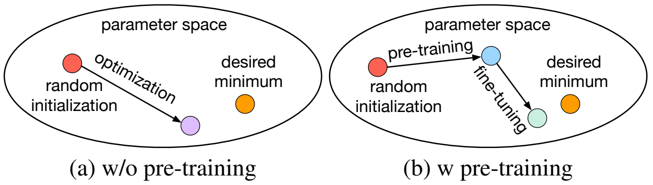


Fig. 5: Architectures of (a) dense block, (b) FU module, and (c) spatiotemporal discriminator.

Spatiotemporal discriminator. We build a spatiotemporal discriminator to judge the spatial and temporal realness of the volumes generated by STNet. As displayed in Figure 5 (c), two Conv layers are utilized to extract spatial information from the input volumes. Each Conv decreases the dimension by half while doubling the channels. Then, temporal coherence is evaluated by incorporating ConvLSTM that accepts the features of the previous and current time steps as inputs. Finally, a Conv and *global average pooling* (GAP) [34] layer compresses the feature into a single value, which scores the realness of the input volume. ReLU is picked as the activation function, excluding the ConvLSTM and GAP layers.

3.4 Optimization

To optimize STNet, we design a two-stage training algorithm: *pre-training* and *fine-tuning*. Pre-training offers an appropriate starting point for training STNet and provides the network with better generalization ability during inference. Fine-tuning fits the network in the downstream STSR task.



(a) w/o pre-training

(b) w pre-training

Fig. 6: Illustration of how learnable parameters change (a) without and (b) with pre-training.

Why pre-training? A straightforward optimization is to randomly initialize the learnable parameters in STNet and directly use low-resolution and high-resolution pairs to train STNet, as sketched in Figure 6 (a). Although random initialization can promote the network to a local minimum, it could not improve further to the desired one since it would overfit the training data. However, adding pre-training can promote the randomly initialized parameters to a local minimum based on the pre-training task, which can be utilized in the downstream task. Then, based on this new starting point, the network can better fit the data, as shown in Figure 6 (b). The pre-training stage can guide the learning process towards the minima to support better generalization from the training data [12]. In the context of STSR, pre-training can help STNet see the inference data in the low-dimensional space, preventing the network from overfitting in the training data and enhancing the network's generalization ability in the inference data.

Pre-training. To pre-train in an unsupervised fashion, we leverage *cycle* loss to optimize STNet. The cycle loss for $\mathbf{V}_i^S$ is defined as

$$\mathscr{L}_{\text{cyc}} = ||\mathscr{D}(\mathbf{V}_i^S) - \mathbf{V}_i^L||_2, \tag{1}$$

where $\mathscr{D}$ denotes a downsizing operation (e.g., trilinear) and $||\cdot||_2$ is L^2 norm. The rationale for designing this loss is that once the super-resolution volumes are generated and if we downsize them again to the low-dimensional space, the downsized version (i.e., $\mathscr{D}(\mathbf{V}_i^S)$) should be consistent with the original low-resolution volumes (i.e., $\mathbf{V}_i^L$).

Fine-tuning. To fine-tune STNet in the STSR task, we leverage *volumetric* loss for the closeness to high-resolution volumes, and *adversarial* loss for the realness, to train STNet. The volumetric loss for $\mathbf{V}_i^S$ is defined as

$$\mathscr{L}_{\text{vol}}^{\text{G}} = ||\mathbf{V}_i^S - \mathbf{V}_i^H||_2. \tag{2}$$

The adversarial losses of generator G and discriminator D for $\{\mathbf{V}_i^S, \cdots, \mathbf{V}_{i+t+1}^S\}$ are defined as

$$\mathscr{L}_{\text{adv}}^{\text{G}} = \sum_{j=0}^{t+1} 1 - \text{D}\left(\mathbf{V}_{i+j}^S | \mathbf{V}_{i+j-1}^S, \cdots, \mathbf{V}_i^S\right), \tag{3}$$

$$\begin{aligned} \mathscr{L}_{\text{adv}}^{\text{D}} = & \sum_{j=0}^{t+1} \text{D}\left(\mathbf{V}_{i+j}^S | \mathbf{V}_{i+j-1}^S, \cdots, \mathbf{V}_i^S\right) \\ & + \sum_{j=0}^{t+1} 1 - \text{D}\left(\mathbf{V}_{i+j}^H | \mathbf{V}_{i+j-1}^H, \cdots, \mathbf{V}_i^H\right). \end{aligned} \tag{4}$$

Considering both volumetric and adversarial losses, we define the total loss of G as

$$\mathscr{L}^{\text{G}} = \lambda_{\text{vol}} \mathscr{L}_{\text{vol}}^{\text{G}} + \lambda_{\text{adv}} \mathscr{L}_{\text{adv}}^{\text{G}}, \tag{5}$$

where λ_{vol} and λ_{adv} control the weights of these two losses.

The training algorithm of STNet is listed in Algorithm 1. In the pre-training stage, we first use the sparsely sampled low-resolution sequence $\mathbf{V}^{L'}$, as sketched in Figure 2 (a), to train STNet. After training T_P epochs, we begin to fine-tune STNet using the low-resolution and high-resolution pairs at the early time steps $\mathbf{V}^T$, as shown in Figure 2 (b). Following Wang et al. [56], the fine-tuning stage contains two steps. First, only volumetric loss is applied to optimize STNet for training stabilization and computational cost reduction. Second, the spatiotemporal discriminator D is involved in the training procedure for enhancing spatial and temporal coherence.

Table 1: Variables and dimension of each data set.

data set	variables	dimension ($x \times y \times z \times t$)
five jets	intensity	$128 \times 128 \times 128 \times 100$
ionization	H, H+, He, He+	$600 \times 248 \times 248 \times 100$
half-cylinder [48]	velocity magnitude	$640 \times 240 \times 80 \times 100$
supercurrent	rho	$256 \times 128 \times 32 \times 200$
Tangaroa [46]	velocity magnitude	$300 \times 180 \times 120 \times 150$
vortex	vorticity magnitude	$128 \times 128 \times 128 \times 90$

4 Results and Discussion

4.1 Data Sets and Network Training

We tested STNet using the data sets reported in Table 1. Note that half-cylinder is an ensemble data set with different Reynolds numbers (i.e., 320, 640, and 6,400). PyTorch [44] was used for implementation. Training and inference were performed on an NVIDIA TESLA V100 GPU. The low-resolution data were obtained by applying bicubic kernel with reflection padding. We scaled the range of $\mathbf{V}^L$ and $\mathbf{V}^H$ to $[-1, 1]$. We initialized parameters following He et al. [24] for optimization and utilized the Adam optimizer [30] for parameter update. In each mini-batch, one training sample is used. The learning rates for G and D are 10^{-4} with $\beta_1 = 0.9$, $\beta_2 = 0.999$. $\lambda_{\text{vol}} = 1$ and $\lambda_{\text{adv}} = 10^{-3}$. T_P, T_{F_1}, and T_{F_2} are set to 200, 400, and 50 epochs, respectively, for all data sets. All these hyperparameter settings are determined based on experiments.

Algorithm 1 STNet training algorithm.

```
Require: initial parameters $\theta_{\text{G}}$ and $\theta_{\text{D}}$ for G and D, number of training epochs
  in pre-training and fine-tuning stages: $T_P$, $T_{F_1}$, and $T_{F_2}$, learning rates $\alpha_{\text{G}}$ and
  $\alpha_{\text{D}}$ for G and D, respectively.
  /* pre-training stage */
  for $j = 1 \cdots T_P$ do
      sample low-resolution data from $\mathbf{V}^{L'}$;
      compute $\mathscr{L}_{\text{cyc}}$ according to Equation 1;
      update $\theta_{\text{G}}$;
  end for
  /* fine-tuning stage 1: only using volumetric loss to optimize STNet */
  for $j = 1 \cdots T_{F_1}$ do
      sample low-resolution and high-resolution data pairs from $\mathbf{V}^T$;
      compute $\mathscr{L}_{\text{vol}}^{\text{G}}$ according to Equation 2;
      update $\theta_{\text{G}}$;
  end for
  /* fine-tuning stage 2: taking temporal coherence into consideration */
  for $j = 1 \cdots T_{F_2}$ do
      sample low-resolution and high-resolution data pairs from $\mathbf{V}^T$;
      compute $\mathscr{L}_{\text{adv}}^{\text{D}}$ according to Equation 4;
      update $\theta_{\text{D}}$;
      compute $\mathscr{L}^{\text{G}}$ according to Equation 5;
      update $\theta_{\text{G}}$;
  end for
```

4.2 Results

Baselines. We compare STNet with three baseline solutions:

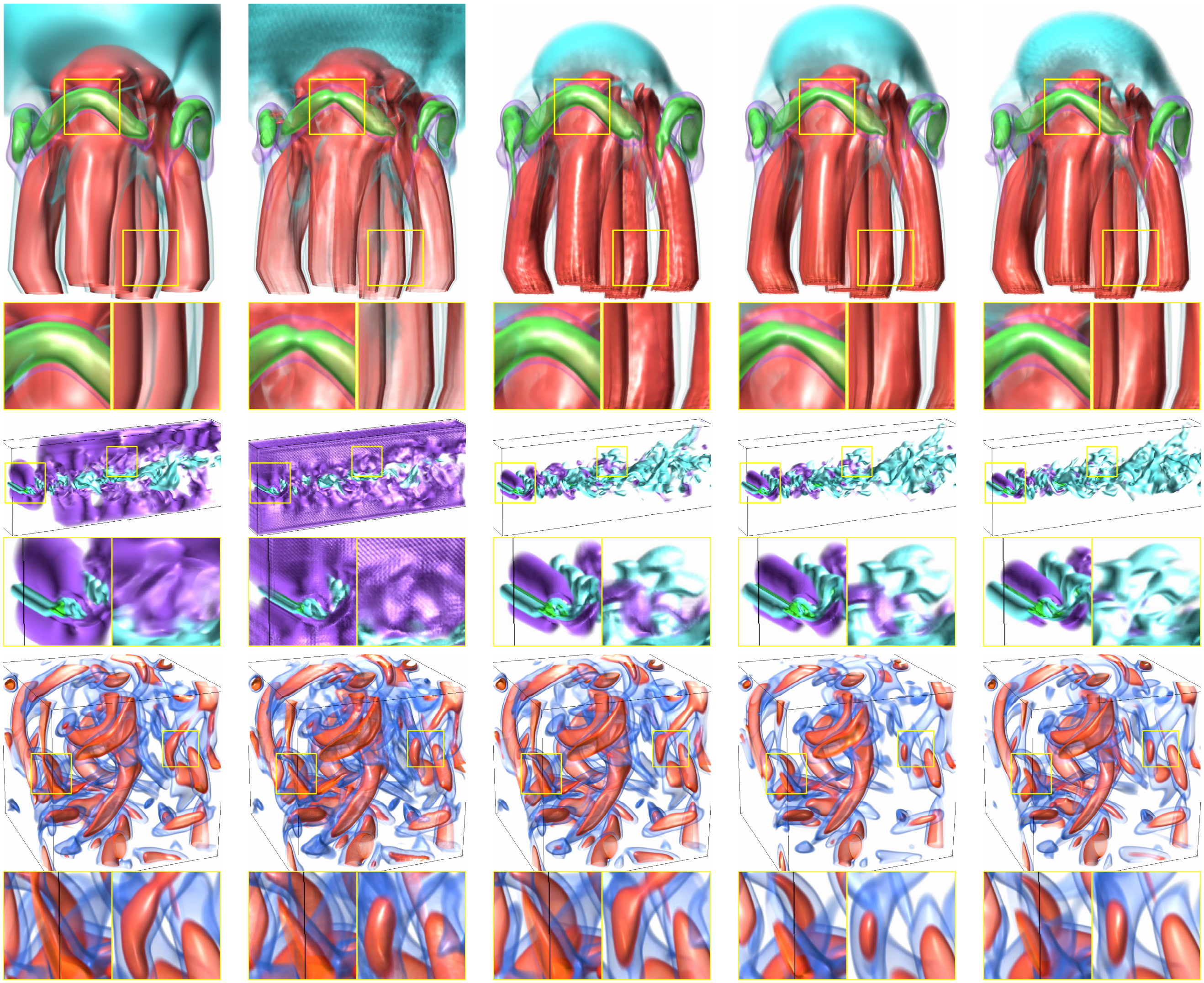
- BL: Bicubic interpolation is used for SSR and linear interpolation for TSR. BL stands for bicubic+linear interpolation.
- SSR+TSR: SSR [20] is a GAN solution for time-varying data SSR, and TSR [21] is a recurrent generative solution for TSR. We train SSR for 400 epochs and TSR for 400 epochs.
- STD: STD is a variant of STNet. Instead of performing temporal interpolation in the feature space, STD directly interpolates the volumes in the data space. Namely, given two volumes at both ends, STD leverages a FEI module to simultaneously learn spatiotemporal features of all intermediate and the two-ending time steps and applies a FU module to generate super-resolution volumes.

Note that existing STSR works [51, 39, 55, 50] are not suitable for 3D data since they could not capture complicated spatiotemporal patterns. As for Xiang et al. [60], it leverages deformable Conv for spatiotemporal super-resolution. However, it is difficult to extend this architecture to handle 3D data sets for two reasons. First, deformable Conv needs to learn offsets to perform Convs, which requires additional parameters and memories. Second, the offsets are learned from the whole data, not a subregion. Thus, the computational cost is extremely high when the volume is large.

Section 1 in the Appendix provides a detailed discussion of the two deep learning baselines. The accompanying video shows the frame-to-frame comparison results. For the same data set, all visualization results follow the same rendering parameters for lighting, viewpoint, transfer function (used in volume rendering), and isovalue (used in isosurface rendering). Except for the GT results, all results from STNet and baseline solutions are rendered using inferred data from a later time step (refer to Figure 2 (b)).

Evaluation metrics. We utilize three metrics, including data-level *peak signal-to-noise* (PSNR), image-level *structural similarity index* (SSIM), and feature-level *isosurface similarity* (IS) [3], for quantitative evaluation.

Quantitative and qualitative analysis. Figure 7 shows volume rendering results produced from BL, SSR+TSR, STD, STNet, and GT using the five jets, half-cylinder (640), and vortex data sets. For the five jets data set, both BL and SSR+TSR produce more cyan parts at the cap, and the rendering results are overly smooth at the legs. STD and STNet generate similar results, but taking a close comparison, STNet synthesizes finer details at the green part (refer to the zoom-ins on the



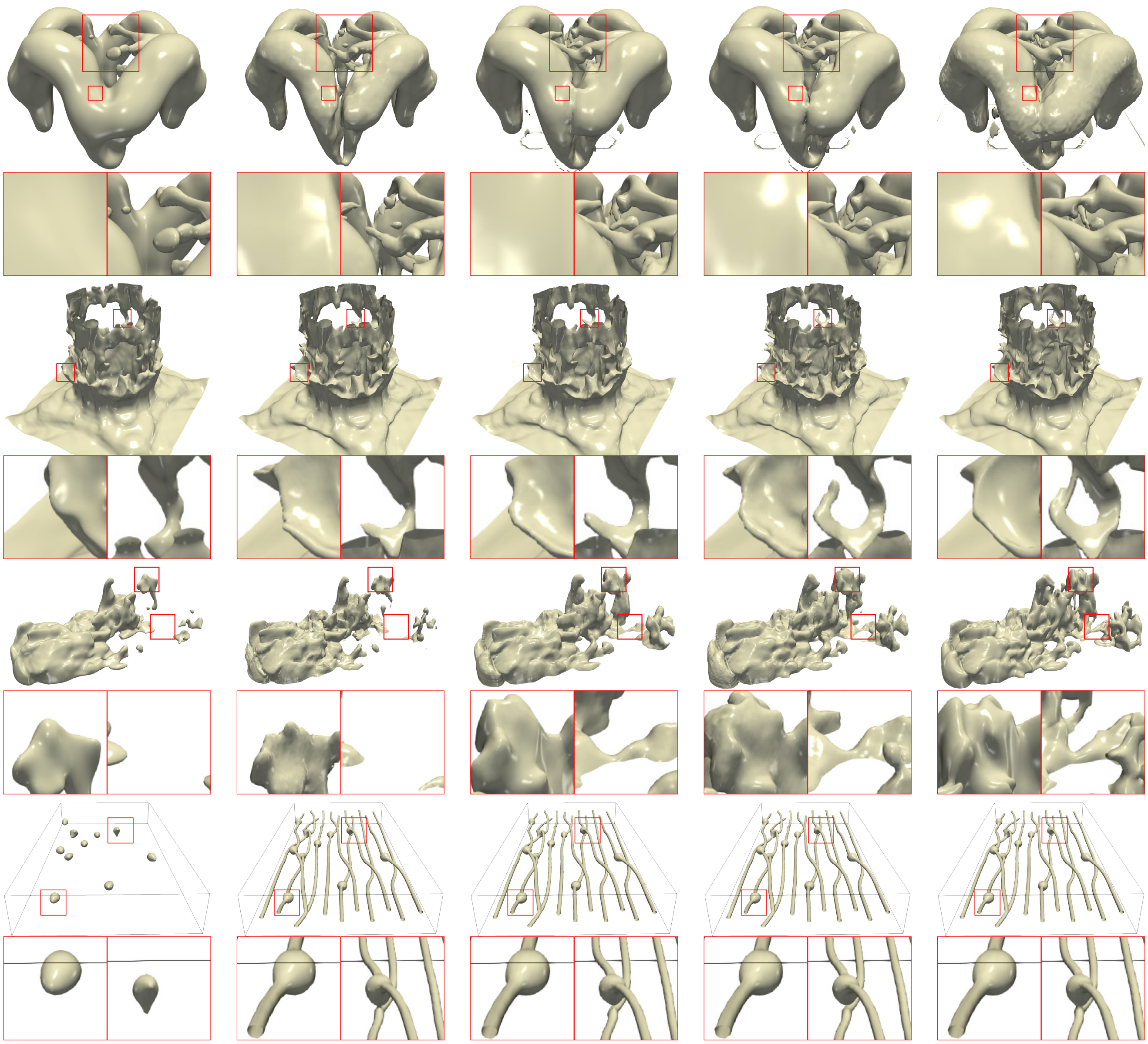
(a) BL (b) SSR+TSR (c) STD (d) STNet (e) GT

Fig. 7: Comparison of volume rendering results. Top to bottom: five jets, half-cylinder (640), and vortex.

Table 2: Average PSNR (dB), SSIM, and training time (in seconds). The best ones are highlighted in bold (same for other tables in the paper).

data set	method	PSNR	SSIM	training time	data set	method	PSNR	SSIM	training time
five jets	BL	27.96	0.751	—	supercurrent	BL	26.92	0.812	—
	SSR+TSR	27.30	0.685	64.024		SSR+TSR	44.54	0.995	87.258
	STD	**40.00**	0.892	24.662		STD	**44.74**	0.995	31.443
	STNet	39.63	**0.901**	43.702		STNet	44.71	**0.995**	66.285
half-cylinder (640)	BL	28.12	0.864	—	Tangaroa	BL	21.85	0.853	—
	SSR+TSR	24.15	0.792	66.187		SSR+TSR	26.96	0.858	96.037
	STD	35.60	0.907	27.884		STD	30.07	0.879	37.964
	STNet	**36.84**	**0.944**	45.580		STNet	**33.26**	**0.892**	65.568
ionization (H)	BL	33.52	0.862	—	vortex	BL	29.78	**0.749**	—
	SSR+TSR	43.05	0.908	68.123		SSR+TSR	23.89	0.576	57.622
	STD	40.67	0.892	28.556		STD	31.12	0.693	24.573
	STNet	**43.19**	**0.913**	44.781		STNet	**32.73**	0.720	39.329

left) and the legs (refer to the zoom-ins on the right). For the half-cylinder (640) data set, both BL and SSR+TSR do not produce high-quality rendering results. STD generates the result with tiny noises and artifacts at the front (refer to the zoom-ins on the left) and more purple parts (refer to the zoom-ins on the right). As for the vortex data set, BL, SSR+TSR, and STD produce more blue and red parts over the whole volume. STNet generates closer results compared with GT. For the quantitative results, we report average PSNR and SSIM values in Table 2. In general, STNet achieves the best performance among these four solutions, except for the average SSIM for the vortex data set and the average PSNR for the five jets and supercurrent data sets. The model sizes of SSR+TSR, STD, and STNet are 74.0MB, 138MB, and 62.5MB, respectively. As for the training time, STD requires the shortest time for optimization, and SSR+TSR needs the longest time. This is because SSR has one generator and two discriminators, and TSR has a recurrent generator and one discriminator. SSR+TSR needs to optimize two generators and three discriminators to go through one training data sample, incurring an expensive computational cost. Since



(a) BL (b) SSR+TSR (c) STD (d) STNet (e) GT

Fig. 8: Comparison of isosurface rendering results. Top to bottom: five jets, ionization (H), Tangaroa, and supercurrent. The chosen isovalues are 0, 0.5, 0, and -0.2 , respectively.

STNet has more FU modules than STD ($t+1$ vs. 1), it demands more time to compute gradients and optimize. However, there is no significant difference for the inference time.

Figure 8 shows isosurface rendering results among BL, SSR+TSR, STD, STNet, and GT using the five jets, ionization (H), Tangaroa, and supercurrent data sets. For each data set, we pick one isovalue for comparison. For the five jets data set, both BL and SSR+TSR do not capture the isosurface details (refer to the zoom-ins on the right), and the lighting on the isosurface generated by STD shifts too much compared with GT (refer to the zoom-ins on the left). For the ionization (H) data set, STNet can capture finer structures (refer to the zooms-in on the right) compared with other solutions. For the Tangaroa data set, both BL and SSR+TSR do not produce isosurfaces with fine details. In addition, the isosurfaces generated by SSR+TSR contain noticeable noises and artifacts. For STD and STNet, both can synthesize similar isosurfaces compared with GT. However, STNet can extract more details. For example, it produces close isosurfaces at two corners (refer to the zoom-ins). For the supercurrent data set, BL does not extract close isosurfaces compared with GT, while SST+TSR, STD, and STNet produce similar results, and all of them are comparable to GT. In terms of quantitative comparison, Table 3 reports the average IS score for BL, SSR+TSR, STD, and STNet. STNet achieves the best performance for all data sets. Note that for the five jets and supercurrent data sets, STD and STNet achieve similar performance. This is because the overall content does not change too much over different time steps for these two data sets (i.e., the training and inference data are similar), which means data-space interpolation could lead to satisfactory results. But to achieve similar performance, STD needs around 36 million parameters while STNet requires about 16 million.

Comparison with baselines. As shown in Figures 7 and 8, STNet outperforms SSR+TSR and STD in terms of visual quality and achieves better quantitative scores for most data sets compared with STD. The potential reasons are as follows. SSR+TSR directly uses two networks to perform SSR and TSR, respectively. It forces the latter network (i.e., TSR) to complete two tasks, i.e., TSR and denoising, since the results generated from the former network (i.e., SSR) are not GT, and they contain unobservable noises. These noises could be sensitive [63, 26] when synthesizing high-quality volumes. This explains

Table 3: Average IS values at chosen isovalues.

data set (isovalue)	BL	SSR+TSR	STD	STNet
five jets ($v = 0$)	0.78	0.80	**0.88**	**0.88**
half-cylinder (640) ($v = 0.2$)	0.62	0.60	0.74	**0.79**
ionization (H) (v =0.5)	0.71	0.78	0.79	**0.81**
supercurrent ($v = -0.2$)	0.23	**0.96**	**0.96**	**0.96**
Tangaroa ($v = 0$)	0.57	0.59	0.73	**0.75**
vortex ($v = -0.2$)	0.82	0.79	0.84	**0.86**

why the rendering results produced by SSR+TSR contain noises and artifacts. We try to add a denoising module into the TSR framework to clean up these noises, but the results are not satisfactory. For STD, it achieves comparable PSNR values for simple data sets (e.g., supercurrent) but cannot generate high fidelity results for complex data sets (e.g., half-cylinder). This is because extracting a global spatiotemporal representation for all intermediate and the two-ending time steps is extremely difficult for these volumes whose patterns change dynamically. We point out that the number of parameters in STD is twice of those in STNet. The reason is as follows. Ideally, to achieve a fair comparison between STD and STNet, we need to set the same number of parameters in both STD and STNet. However, under the model size of 62.5MB, STD cannot generate satisfactory STSR volumes. Therefore, we expand the width (i.e., the number of channels) of STD.

Table 4: Comparison of TTHRESH and STNet. Top: average SSIM under the same PSNR of 36.84 dB. Bottom: average PSNR (dB) and SSIM under the same compression ratio of 206.74×.

data set	method	compression ratio	SSIM
half-cylinder (640)	TTHRESH	**1745.33 ×**	0.902
	STNet	162.62 ×	**0.944**

data set	method	PSNR	SSIM
ionization (H)	TTHRESH	**49.37**	0.906
	STNet	43.19	**0.913**

Comparison with state-of-the-art compression. Figure 9 shows volume rendering results obtained from the upscaled volumes generated by STNet and the volumes compressed then decompressed using TTHRESH [1]. We choose TTHRESH, a tensor compression solution, because it smoothly degrades the data, leads to errors smaller than other state-of-the-art algorithms, and requires a low cost for compression and decompression. We consider two scenarios: (1) keeping the same PSNR (i.e., 36.84 dB) for the half-cylinder (640) data set and (2) controlling the same compression ratio (i.e., 206.74×) for the ionization (H) data set. To achieve a fair comparison against TTHRESH, we include the model size in the computation of compression ratio. We utilize a lossless compression algorithm [35] to further reduce the storage of the saved model and data. For the first scenario, as shown in Figure 9 (a), the image rendered by TTHRESH contains fewer cyan parts. It also produces noticeable noises in the rendering image. The top part of Table 4 reports the compression ratios and average SSIM values for both methods. Under the same PSNR, although TTHRESH achieves a higher compression ratio, which is about 10 times compared with that of STNet, STNet achieves a higher SSIM value for the synthesized volumes than those recovered from TTHRESH. For the second scenario, as shown in Figure 9 (b), TTHRESH generates more red parts. The bottom part of Table 4 gives average PSNR and SSIM values for both methods. Under the same compression ratio, although TTHRESH produces a higher PSNR value, STNet achieves a higher SSIM value and better visual quality.

Evaluation of ensemble and multivariate data sets. In Figures 10 and 11, we compare volume and isosurface rendering results from the synthesized volumes given by BL and STNet on ensemble and multivariate data sets to evaluate the generalization ability. We use an ensemble parameter (variable) X_S of a data set for training, while another ensemble parameter (variable) X_T of the same data set is used for inference (i.e., $X_S \rightarrow X_T$). For the half-cylinder data set, we test two cases: $640 \rightarrow 320$ and $640 \rightarrow 6,400$. The complexity increases as the Reynolds number gets large. For $640 \rightarrow 320$, compared with the result generated by BL, STNet produces better visual results in the purple and cyan parts of volume rendering results and extracts finer details of isosurfaces as shown in isosurface rendering results. For $640 \rightarrow 6,400$, STNet synthesizes more detailed rendering results. For example, the cyan part's lighting and the isosurfaces at the middle and right corners are closer to GT results. As for H → H+ of the ionization data set, BL produces fewer details at the bottom of the ionization for both rendering results. For instance, for volume rendering, the image generated by BL shows more purple color. For isosurface rendering, the lighting at the bottom generated by BL is inconsistent with that of GT. As for quantitative results, STNet also outperforms BL in terms of PSNR and SSIM, as shown in Table 5.

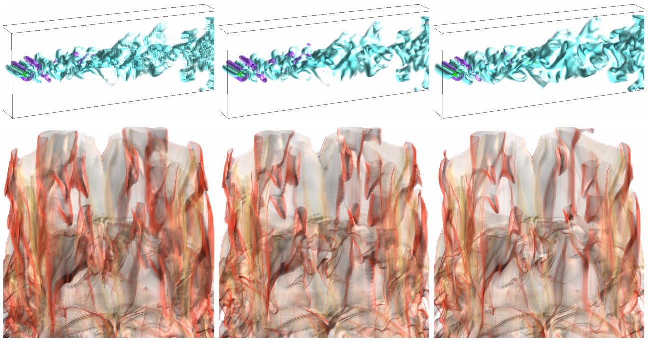

(a) TTHRESH (b) STNet (c) GT

Fig. 9: Volume rendering results. Top and bottom: half-cylinder (640) and ionization (H).

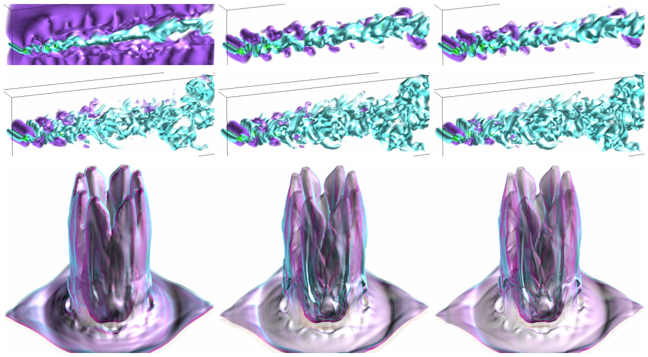
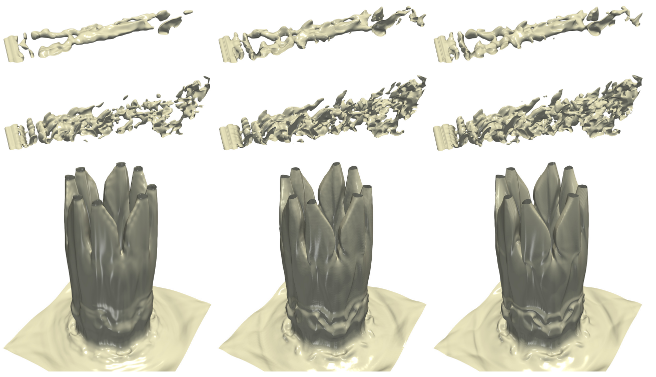

(a) BL (b) STNet (c) GT

Fig. 10: Variable and ensemble volume rendering results. Top to bottom: half-cylinder (320), half-cylinder (6,400), and ionization (H+).

Table 5: Average PSNR (dB) and SSIM for ensemble and multivariate data sets.

data set ($X_S \rightarrow X_T$)	method	PSNR	SSIM
half-cylinder ($640 \rightarrow 320$)	BL	30.01	0.886
	STNet	**35.26**	**0.951**
half-cylinder ($640 \rightarrow 6,400$)	BL	26.47	0.857
	STNet	**33.86**	**0.926**
ionization (H→H+)	BL	33.53	0.867
	STNet	**42.99**	**0.910**

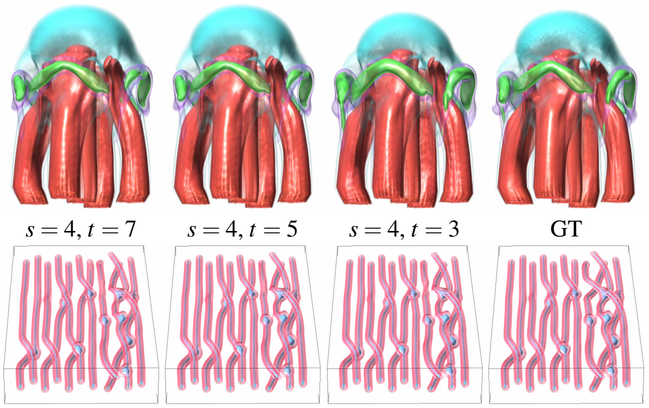
Evaluation of s and t. To analyze the performance of STNet with different s and t, we set $s = 4$ and $s = 8$ with different t using the five jets and supercurrent data sets, respectively. As shown in the top rows of Figures 12 and 13, $t = 3$ achieves the best quality. However, all of them can capture the overall shape and details of the five jets, and the main difference is the size of the cyan cap. For the supercurrent data set, the rendering results are shown in the bottom rows of Figures 12 and 13. All produce similar results compared to GT for



(a) BL (b) STNet (c) GT

Fig. 11: Variable and ensemble isosurface rendering results. Top to bottom: half-cylinder (320), half-cylinder (6,400), and ionization (H+). The chosen isovalues are −0.2, 0, and −0.2, respectively.

volume and isosurface rendering. But taking a close comparison, under $t = 10$, the isosurface is broken into two parts at the top-left corner (refer to the red arrow). Besides, average PSNR and SSIM values are shown in Figure 14. STNet significantly outperforms BL for the five jets and supercurrent data sets under different settings of s and t.



$s = 4, t = 7$ $s = 4, t = 5$ $s = 4, t = 3$ GT

$s = 8, t = 10$ $s = 8, t = 6$ $s = 8, t = 2$ GT

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

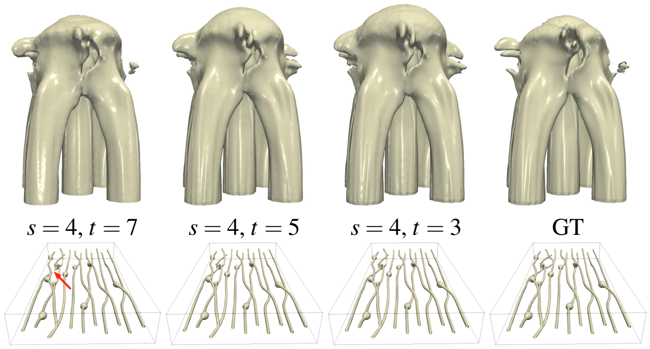
Fig. 12: Volume rendering results under different s and t. Top and bottom: five jets and supercurrent.

Based on the above results, our suggestions for choosing s and t for different data sets are as follows.

- With $s = 4$, the appropriate value for t could be large for simple data sets (e.g., five jets and supercurrent), where the patterns change slowly over time. The suitable value is determined by the total sample time steps. With sufficient samples, t could be 9 for the supercurrent data set (200 time steps), while with limited samples, t could be 5 for the five jets data set (100 time steps).
- With $s = 4$, for complex data sets (e.g., half-cylinder and vortex) where the patterns could evolve rapidly in neighborhood time steps, 3 is a proper value for t.
- With $s = 8$, it is almost infeasible with the current architecture for upscaling these data sets (e.g., half-cylinder and vortex), where the spatial structures are complex.
- For data sets (e.g., five jets and supercurrent) where the shapes are simple and less complicated, $s = 8$ could still work.

Refer to Section 2 in the Appendix for additional results.

Temporal coherence. To compare how well temporal coherence is preserved using BL and STNet, we show five consecutive time steps using the half-cylinder (320) data set. As shown in Figure 15, STNet can better capture temporal coherence compared with BL as BL does not produce meaningful intermediate time steps. This is because BL only assumes that features evolve linearly, which is not the case for most data sets.

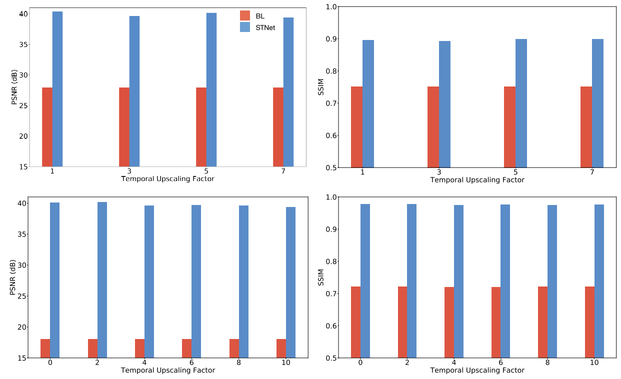


$s = 4, t = 7$ $s = 4, t = 5$ $s = 4, t = 3$ GT

$s = 8, t = 10$ $s = 8, t = 6$ $s = 8, t = 2$ GT

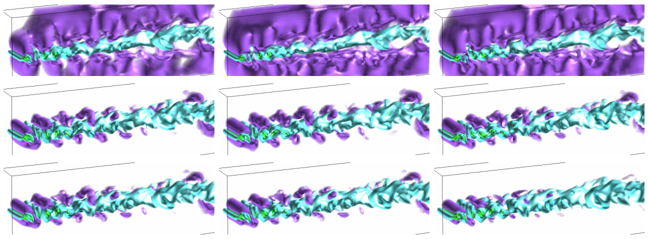
(a) (b) (c) (d)

Fig. 13: Isosurface rendering results under different s and t. Top and bottom: five jets and supercurrent. The chosen isovalues are 0.4 and −0.2, respectively.



(a) PSNR (b) SSIM

Fig. 14: Average PSNR and SSIM under different s and t. Top and bottom: five jets with $s = 4$ and supercurrent with $s = 8$.



(a) 94 (b) 95 (c) 96

Fig. 15: Volume rendering results of the half-cylinder (320) data set with five time steps (94 to 96). Top to bottom: BL, STNet, and GT.

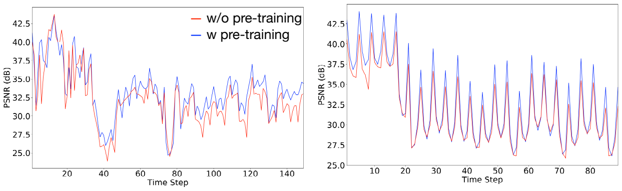
4.3 Network Analysis

To analyze STNet, we study the impact of pre-training and loss function. A detailed discussion is as follows.

Evaluation of pre-training. To investigate the effectiveness of adding pre-training, we train STNet with and without pre-training. Table 6 gives the average PSNR and SSIM under these two training schemes. As we can see, the pre-training can improve about 1dB and 0.01 for the average PSNR and SSIM values, respectively. Moreover, we plot the PSNR curves over the whole sequence, as shown in Figure 16. The curves indicate that pre-training can boost the PSNR value at almost every time step. As for visual quality, volume rendering results are shown in the top row of Figure 17. Clearly, using pre-training

Table 6: Average PSNR (dB) and SSIM under different settings.

data set	method	PSNR	SSIM
Tangaroa	w/o pre-training	32.26	0.883
	w pre-training	**33.26**	**0.892**
vortex	w/o pre-training	31.75	0.709
	w pre-training	**32.73**	**0.720**
five jets	w/o volumetric loss	22.11	0.621
	w volumetric loss	**39.63**	**0.892**
half-cylinder (640)	w/o volumetric loss	20.62	0.888
	w volumetric loss	**36.84**	**0.944**
ionization (H)	w/o adversarial loss	**43.80**	0.904
	w adversarial loss	43.19	**0.913**



(a) Tangaroa (b) vortex

Fig. 16: PSNR curves with and without pre-training.

can generate results closer to GT (refer to the yellow ellipse). These quantitative and qualitative analysis results confirm the usefulness of the pre-training algorithm.

Evaluation of volumetric loss. Cycle and volumetric losses serve a similar role in optimization while constraining the volumes in the low-dimensional and high-dimensional spaces, respectively. So, would it still work if we only leverage one loss to train the network? To answer this question, we optimize the network with and without considering volumetric loss. Note that training without cycle loss means removing pre-training, which has been discussed. As shown in Table 6, without volumetric loss, average PSNR and SSIM values drop significantly. As for rendering quality, we display volume rendering results in Figure 17. Using cycle loss only captures the overall shape of the five jets but could not preserve fine details. This is because different data samples in the high-dimensional space can be downsized to the same data in the low-dimensional space with the same downsizing function (e.g., bicubic) [37]. Constrained only in the low-dimensional space, the network could jump into an undesired local minimum in the high-dimensional space.

Evaluation of adversarial loss. To study the impact of adversarial loss, we optimize STNet with and without adversarial loss. Table 6 reports the average PSNR and SSIM under these two optimizations. Although we can achieve a higher PSNR value without adversarial loss, adding adversarial loss improves SSIM values (i.e., the image-level metric). Moreover, the rendering images also confirm that adversarial loss can boost perceptual quality, as shown in Figure 17. For example, without adversarial loss, the rendering image produces more red parts at the ionization's head. Therefore, these results demonstrate the usefulness of adversarial loss in improving visual quality.

5 Conclusions and Future Work

We have presented STNet, a novel generative solution for producing STSR volumes for time-varying data analysis and visualization. Leveraging post-upsampling and feature interpolation, STNet can synthesize high fidelity super-resolution sequence given two low-resolution volumes at both ends as input. Compared to BL, SSR+TSR, and STD, STNet produces time-varying sequences of better visual quality, both qualitatively and quantitatively. We also compare STNet with TTHRESH to verify its effectiveness.

STNet can be applied to the in-situ scenario: at simulation time, scientists can store the early time steps for STNet training while saving the later time steps sparsely for storage saving. For example, they can keep one time step for every ten time steps simulated and downsize these time steps by 4 at each spatial dimension. During postprocessing, the network is trained with the early time steps only. Once trained, they can recover the super-resolution intermediate time steps with high fidelity, given the sparsely output low-resolution time steps.

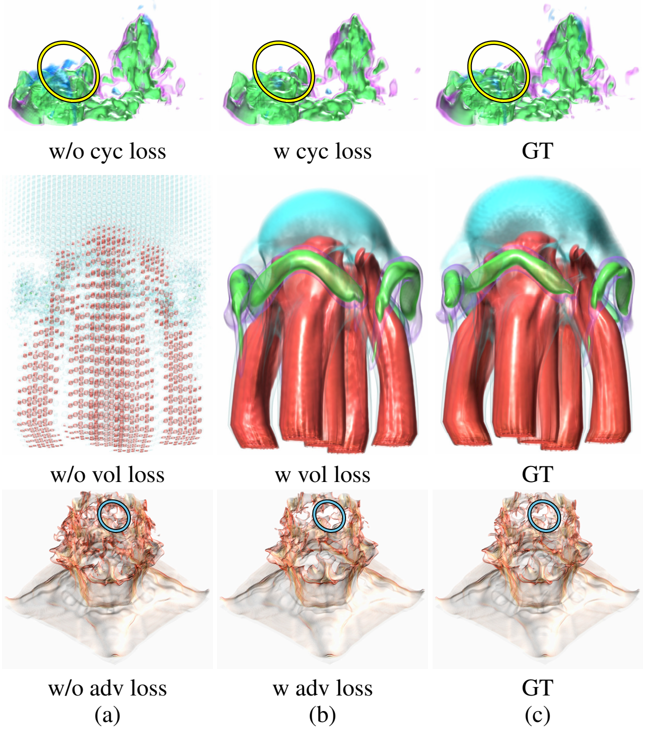


Fig. 17: Volume rendering results under different loss settings. From top to bottom: Tangaroa, five jets, and ionization (H).

Our future work aims to improve STNet in three aspects. (1) **Adaptive sampling.** The current temporal sampling strategy is based on uniform sampling, which may not capture the dynamic pattern well. Ideally, we need to densely store time steps when the pattern evolves rapidly and sparsely save time steps when the pattern changes slowly. Based on adaptive sampling, STNet can better learn the data structures in both spatial and temporal dimensions, and the performance can be improved further. (2) **Unsupervised super-resolution.** So far, STNet still relies on the high-resolution data as a reference to learn the mapping from low-resolution to super-resolution. However, this requirement impedes its implementation in the in-situ scenario. We want to explore the possibility of generating super-resolution in an unsupervised manner by incorporating knowledge distillation [14] with cycle loss. (3) **Larger spatial upscaling factor.** For now, given complex data sets, STNet can only upscale the volumes 4 times along each spatial dimension (leading to 64 times reduction) due to the network's limited capability. We will design a more powerful framework that can handle a larger spatial upscaling factor, such as 8 or 16 [4].

Acknowledgments

This research was supported in part by the U.S. National Science Foundation through grants IIS-1455886, CCF-1617735, CNS-1629914, DUE-1833129, IIS-1955395, IIS-2101696, and OAC-2104158. The authors would like to thank the anonymous reviewers for their insightful comments.

References

[1] R. Ballester-Ripoll, P. Lindstrom, and R. Pajarola. TTHRESH: Tensor compression for multidimensional visual data. *IEEE Transactions on Visualization and Computer Graphics*, pages 7324–7334, 2019.

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

[3] S. Bruckner and T. Möller. Isosurface similarity maps. *Computer Graphics Forum*, 29(3):773–782, 2010.

[4] K. C. Chan, X. Wang, X. Xu, J. Gu, and C. C. Loy. GLEAN: Generative latent bank for large-factor image super-resolution. *arXiv preprint arXiv:2012.00739*, 2020.

[5] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of International Conference on Machine Learning*, 2020.

[6] Z. Chen, W. Zeng, Z. Yang, L. Yu, C.-W. Yu, M. Raj, and H. Qu. LassoNet: Deep lasso-selection of 3D point clouds. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):195–204, 2020.

[7] T. Dai, J. Cai, Y. Zhang, S.-T. Xia, and L. Zhang. Second-order attention network for single image super-resolution. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 11065–11074, 2019.

[8] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of IEEE International Conference on Computer Vision*, pages 1422–1430, 2015.

[9] J. Donahue and K. Simonyan. Large scale adversarial representation learning. In *Proceedings of Advances in Neural Information Processing Systems*, pages 10542–10552, 2019.

[10] 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.

[11] D. Engel and T. Ropinski. Deep volumetric ambient occlusion. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1268–1278, 2021.

[12] D. Erhan, A. Courville, Y. Bengio, and P. Vincent. Why does unsupervised pre-training help deep learning? In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 201–208, 2010.

[13] S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. In *Proceedings of International Conference on Learning Representations*, 2018.

[14] J. Gou, B. Yu, S. J. Maybank, and D. Tao. Knowledge distillation: A survey. *arXiv preprint arXiv:2006.05525*, 2020.

[15] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko. Bootstrap your own latent: A new approach to self-supervised learning. In *Proceedings of Advances in Neural Information Processing Systems*, 2020.

[16] P. Gu, J. Han, D. Z. Chen, and C. Wang. Reconstructing unsteady flow data from representative streamlines via diffusion and deep learning based denoising. *IEEE Computer Graphics and Applications*, 41(5), 2021. In Press.

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

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

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

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

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

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

[23] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

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

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

[26] D. Hendrycks, K. Lee, and M. Mazeika. Using pre-training can improve model robustness and uncertainty. In *Proceedings of International Conference for Learning Representations*, 2019.

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

[28] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 4700–4708, 2017.

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

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

[31] G. Larsson, M. Maire, and G. Shakhnarovich. Learning representations for automatic colorization. In *Proceedings of the European Conference on Computer Vision*, pages 577–593, 2016.

[32] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. P. 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.

[33] Z. Li, J. Yang, Z. Liu, X. Yang, G. Jeon, and W. Wu. Feedback network for image super-resolution. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 3867–3876, 2019.

[34] M. Lin, Q. Chen, and S. Yan. Network in network. In *Proceedings of International Conference for Learning Representations*, 2014.

[35] P. Lindstrom and M. Isenburg. Fast and efficient compression of floating-point data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1245–1250, 2006.

[36] Z. Liu, R. A. Yeh, X. Tang, Y. Liu, and A. Agarwala. Video frame synthesis using deep voxel flow. In *Proceedings of IEEE International Conference on Computer Vision*, pages 4463–4471, 2017.

[37] S. Menon, A. Damian, S. Hu, N. Ravi, and C. Rudin. PULSE: Self-supervised photo upsampling via latent space exploration of generative models. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 2437–2445, 2020.

[38] S. Meyer, O. Wang, H. Zimmer, M. Grosse, and A. Sorkine-Hornung. Phase-based frame interpolation for video. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1410–1418, 2015.

[39] U. Mudenagudi, S. Banerjee, and P. K. Kalra. Space-time super-resolution using graph-cut optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):995–1008, 2010.

[40] 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.

[41] A.-D. Nguyen, W. Kim, J. Kim, and S. Lee. Video frame interpolation by plug-and-play deep locally linear embedding. *arXiv preprint arXiv:1807.01462*, 2018.

[42] S. Niklaus, L. Mai, and F. Liu. Video frame interpolation via adaptive separable convolution. In *Proceedings of IEEE International Conference on Computer Vision*, pages 261–270, 2017.

[43] B. Niu, W. Wen, W. Ren, X. Zhang, L. Yang, S. Wang, K. Zhang, X. Cao, and H. Shen. Single image super-resolution via a holistic attention network. In *Proceedings of European Conference on Computer Vision*, pages 191–207, 2020.

[44] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Proceedings of Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

[45] 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.

[46] S. Popinet, M. Smith, and C. Stevens. Experimental and numerical study of the turbulence characteristics of airflow around a research vessel. *Jour-*

nal of Atmospheric and Oceanic Technology, 21(10):1575–1589, 2004.

[47] W. P. Porter, Y. Xing, B. R. von Ohlen, J. Han, and C. Wang. A deep learning approach to selecting representative time steps for time-varying multivariate data. In *Proceedings of IEEE VIS Conference (Short Papers)*, pages 131–135, 2019.

[48] I. B. Rojo and T. Günther. Vector field topology of time-dependent flows in a steady reference frame. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):280–290, 2019.

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

[50] O. Shahar, A. Faktor, and M. Irani. Space-time super-resolution from a single video. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 3353–3360, 2011.

[51] E. Shechtman, Y. Caspi, and M. Irani. Increasing space-time resolution in video. In *Proceedings of European Conference on Computer Vision*, pages 753–768, 2002.

[52] N. Shi and Y. Tao. CNNs based viewpoint estimation for volume visualization. *ACM Transactions on Intelligent Systems and Technology*, 10(3):27:1–27:22, 2019.

[53] A. Shocher, N. Cohen, and M. Irani. Zero-shot super-resolution using deep internal learning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 3118–3126, 2018.

[54] J. W. Soh, G. Y. Park, J. Jo, and N. I. Cho. Natural and realistic single image super-resolution with explicit natural manifold discrimination. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 8122–8131, 2019.

[55] H. Takeda, P. Van Beek, and P. Milanfar. Spatiotemporal video upscaling using motion-assisted steering kernel (mask) regression. In M. Mrak, M. Grgic, and M. Kunt, editors, *High-Quality Visual Experience*, pages 245–274. 2010.

[56] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy. ESGAN: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European Conference on Computer Vision Workshops*, 2018.

[57] Z. Wang, J. Chen, and S. C. H. Hoi. Deep learning for image super-resolution: A survey. *arXiv preprint arXiv:1902.06068*, 2019.

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

[59] S. Weiss, J. Han, C. Wang, and R. Westermann. Deep learning-based upscaling for in situ volume visualization. In H. Childs, J. Bennett, and C. Garth, editors, *In Situ Visualization for Computational Science*. Springer, 2021. Accepted.

[60] X. Xiang, Y. Tian, Y. Zhang, Y. Fu, J. P. Allebach, and C. Xu. Zooming Slow-Mo: Fast and accurate one-stage space-time video super-resolution. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 3370–3379, 2020.

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

[62] Y. Yuan, S. Liu, J. Zhang, Y. Zhang, C. Dong, and L. Lin. Unsupervised image super-resolution using cycle-in-cycle generative adversarial networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 701–710, 2018.

[63] R. Zhang. Making convolutional networks shift-invariant again. In *Proceedings of IEEE International Conference on Machine Learning*, pages 7324–7334, 2019.

[64] Z. Zhang, Z. Wang, Z. Lin, and H. Qi. Image super-resolution by neural texture transfer. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 7982–7991, 2019.

[65] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of IEEE International Conference on Computer Vision*, pages 2223–2232, 2017.

Appendix

1 Implementation Details and Baselines

For easy reproduction of STNet, we list the pseudocode of the DB and FU modules in Algorithm 1. In addition, the architecture details of STNet with $s = 4$ and $s = 8$ are listed in Tables 1 and 2, respectively.

Algorithm 1 The DB and FU modules in the PyTorch-like format.

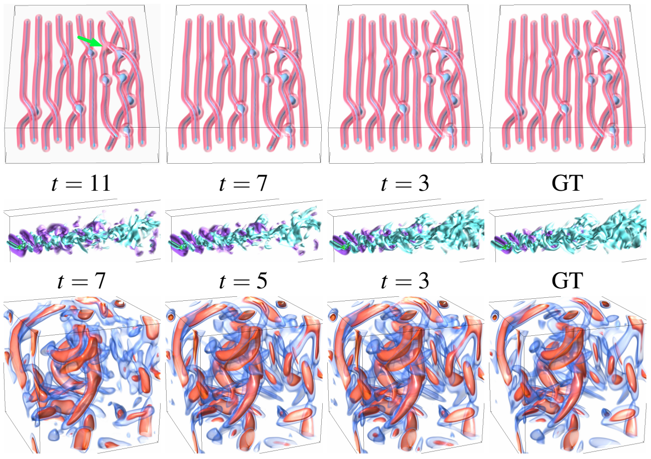
```
/* inc: input channels; outc: output channels */
class DenseBlock():
    def init (self, inc, outc, ac): // ac: activation function
        self.conv1 = Conv(inc, 2×inc, 3, 1, 1)
        self.conv2 = Conv(3×inc, 4×inc, 3, 1, 1)
        self.conv3 = Conv(7×inc, outc, 3, 1 ,1)
        self.ac = ac

    def forward(self, x):
        x1 = self.ac(self.conv1(x))
        x2 = self.ac(self.conv2(torch.cat((x,x1), dim=1))
        x3 = self.ac(self.conv3(torch.cat((x,x1,x2), dim=1)))
        return x3

class FeatureUpscaling():
    def init (self, inc, outc, s, ac): // s: spatial upscaling factor
        self.up1 = VS(inc, outc, s)
        self.up2 = VS(inc, outc, s)
        self.conv = Conv(outc, outc, 3, 1, 1)
        self.conv1 = Conv(outc, outc, 3, 1, 1)
        self.conv2 = Conv(2×outc, outc, 3, 1, 1)
        self.ac = ac

    def forward(self, x):
        x1 = self.ac(self.up1(x))
        /* multiply a weighted mask to the upscaled features */
        x1 = torch.sigmoid(self.conv(self.up2(x))) ⊙ x1
        x2 = self.ac(self.conv1(x1))
        x3 = self.ac(self.conv2(torch.cat((x1, x2), dim=1)))
        return x3
```

We provide the implementation details of the two deep learning baselines (SSR+TSR, STD) in the following.

- SSR+TSR: SSR uses a post-upsampling architecture for upscaling volumes in the spatial dimensions (SSR-TVD), and TSR is a recurrent generative framework for upscaling volumes in the temporal dimension (TSR-TVD). We first leverage SSR for spatial upscaling, then TSR for temporal upscaling. We also attempt the reverse order; however, SSR could not converge. We speculate that TSR+SSR makes SSR struggle in learning the true and synthesized data distributions, which fails spatial upscaling. SSR+TSR only requires TSR to learn the synthesized distribution, which is an easier task. In addition, we modify SSR and TSR in the following ways. First, Tanh is removed in the last Conv layer in both SSR and TSR because we find this activation function will compress the data distribution of the supercurrent data set, leading to poor quality of the rendering results. Second, we add skip connections in TSR to bridge the feature learning and upscaling modules. Third, we only blend forward and backward prediction results in the blending module.
- STD: STD is a variant of STNet. Instead of performing temporal interpolation in the *feature* space like STNet, STD directly interpolates the volumes in the *data* space. That is, given two ending volumes, STD leverages a feature extraction module to simultaneously learn spatiotemporal features of all intermediate and the two ending time steps. It then applies several FU modules to generate super-resolution volumes. Note that the number of output channels in the final Conv layer is $t+2$, where t is the number of intermediate time steps, and 2 represents the two ending time steps. The architecture details are listed in Table 3. We use the same loss functions and training settings in STNet to optimize STD.

Table 1: Architecture parameter details of STNet for $s = 4$. The two parameters in DB are input channels and output channels. The five parameters in Conv are input channels, output channels, kernel size, stride, and padding. The three parameters in VS are input channels, output channels, and spatial upscaling factor. The output size is a five-dimensional tensor (batch size, channels, length, width, height). L, W, and H are the dimensions of the super-resolution volumes.

layer	operator	output size
input	—	1 × 2 × L/4 × W/4 × H/4
FEI-0	$\left\{\begin{array}{c}\text{DB}(1,16)\\ \text{DB}(16,32)\\ \text{DB}(32,64)\\ \text{DB}(64,64)\end{array}\right\}$	2 × 64 × L/4 × W/4 × H/4
FEI-$\{1,2,\cdots,t\}$	$\left\{\begin{array}{c}\text{DB}(2,16)\\ \text{DB}(16,32)\\ \text{DB}(32,64)\\ \text{DB}(64,64)\end{array}\right\}$	1 × 64 × L/4 × W/4 × H/4
FU-1	$\left\{\begin{array}{c}\text{VS}(64,64,2)\\ \text{Conv}(64,64,3,1,1)\\ \text{VS}(64,64,2)\\ \text{Conv}(64,64,3,1,1)\\ \text{Conv}(128,64,3,1,1)\end{array}\right\}$	$(t+2)$ × 64 × L/2 × W/2 × H/2
FU-2	$\left\{\begin{array}{c}\text{VS}(64,32,2)\\ \text{Conv}(32,32,3,1,1)\\ \text{VS}(64,32,2)\\ \text{Conv}(32,32,3,1,1)\\ \text{Conv}(64,32,3,1,1)\end{array}\right\}$	$(t+2)$ × 32 × L × W × H
output	Conv (32,1,3,1,1)	$(t+2)$ × 1 × L × W × H


$t = 11$ $t = 7$ $t = 3$ GT

$t = 7$ $t = 5$ $t = 3$ GT

$t = 5$ (a) $t = 3$ (b) $t = 1$ (c) GT (d)

Fig. 1: Volume rendering results under different t with $s = 4$. Top to bottom: supercurrent, half-cylinder (640), and vortex.

2 Evaluation of s and t

To investigate the performance of STNet on different data sets with various s and t, we test STNet on four data sets (five jets, half-cylinder, supercurrent, and vortex) with $s \in \{4,8\}$ and $t \in \{0,1,\cdots,11\}$.

With $s = 4$, volume and isosurface rendering results are shown in Figures 1 and 2 for the supercurrent, half-cylinder (640), and vortex data sets. In terms of volume rendering results (Figure 1), there is no clear difference under different t for the supercurrent data set. But taking a closer comparison, under $t = 11$, one blue line is broken into two separate lines at the top-right corner (refer to the green arrow). As for the half-cylinder and vortex data sets, STNet can only achieve high-fidelity rendering results under $t = 3$. Furthermore, the results are far away from GT. In terms of isosurface rendering results (Figure 2), again, we can see that using a smaller t leads to a better rendering result. While the results for the supercurrent data set are indistinguishable, we can observe clear quality degradation as t goes beyond 3 for both half-cylinder and vortex data sets. Besides, we show the quantitative results in Figure 3. STNet outperforms BL for all data sets in terms of PSNR and SSIM, and the only exception is SSIM for the vor-

Table 2: Architecture parameter details of STNet for $s = 8$.

layer	operator	output size
input	—	1 × 2 × L/8 × W/8 × H/8

FEI-0	DB(1,16)
	DB(16,32)
	DB(32,64)
	DB(64,64)
		2 × 64 × L/8 × W/8 × H/8

FEI-$\{1,2,\cdots,t\}$	DB(2,16)
	DB(16,32)
	DB(32,64)
	DB(64,64)
		1 × 64 × L/8 × W/8 × H/8

FU-1	VS(64,64,2)
	Conv(64,64,3,1,1)
	VS(64,64,2)
	Conv(64,64,3,1,1)
	Conv(128,64,3,1,1)
		$(t+2)$ × 64 × L/4 × W/4 × H/4

FU-2	VS(64,32,2)
	Conv(32,32,3,1,1)
	VS(64,32,2)
	Conv(32,32,3,1,1)
	Conv(64,32,3,1,1)
		$(t+2)$ × 32 × L/2 × W/2 × H/2

FU-3	VS(64,32,2)
	Conv(32,32,3,1,1)
	VS(64,32,2)
	Conv(32,32,3,1,1)
	Conv(64,32,3,1,1)
		$(t+2)$ × 32 × L × W × H

output	Conv (32,1,3,1,1)	$(t+2)$ × 1 × L × W × H

Table 3: Architecture parameter details of STD with $s = 4$.

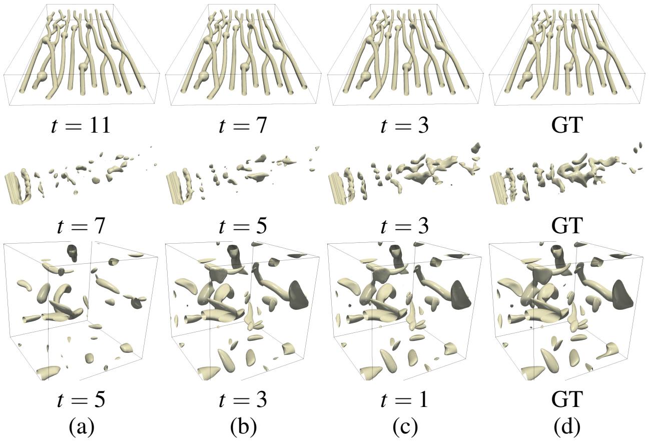
layer	operator	output size
input	—	1 × 2× L/4 × W/4 × H/4

FE	DB(2,32)
	DB(32,64)
	DB(64,128,3,1,1)
	DB(128,256,3,1,1)
		1 × 256× L/4 × W/4 × H/4

FU-1	VS(256,128,2)
	Conv(128,128,3,1,1)
	VS(256,128,2)
	Conv(128,128,3,1,1)
	Conv(256,128,3,1,1)
		1 × 128× L/2 × W/2 × H/2

FU-2	VS(64,64,2)
	Conv(64,64,3,1,1)
	VS(64,64,2)
	Conv(64,64,3,1,1)
	Conv(128,64,3,1,1)
		1 × 64× L × W × H

output	Conv(64,$t+2$,3,1,1)	1 × $(t+2)$× L × W × H


$t = 11$ $t = 7$ $t = 3$ GT

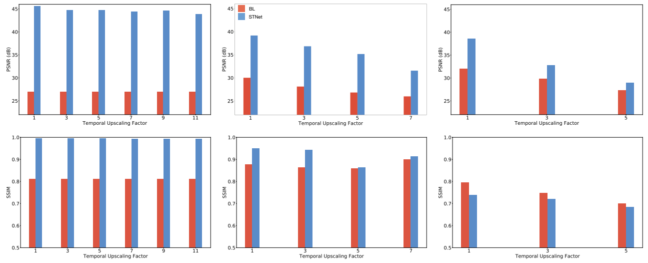
$t = 7$ $t = 5$ $t = 3$ GT

$t = 5$ $t = 3$ $t = 1$ GT

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

Fig. 2: Isosurface rendering results under different t with $s = 4$. Top to bottom: supercurrent, half-cylinder (640), and vortex. The chosen isovalues are 0.2, −0.4, and 0, respectively.

tex data set. However, STNet can produce better visual quality than BL, as indicated by the results in the main paper. We cannot interpolate more time steps for the half-cylinder and vortex data sets since there are not sufficient training samples available. Hence, with $s = 4$, the appropriate value for t is 9 for simple data sets (e.g., supercurrent), where the pattern changes slowly over time. For complex data sets (e.g., half-cylinder and vortex) where the pattern could evolve rapidly in neighboring time steps, 3 is a proper value for t.



(a) supercurrent (b) half-cylinder (640) (b) vortex

Fig. 3: Average PSNR and SSIM under different t with $s = 4$. Top and bottom: PSNR and SSIM.

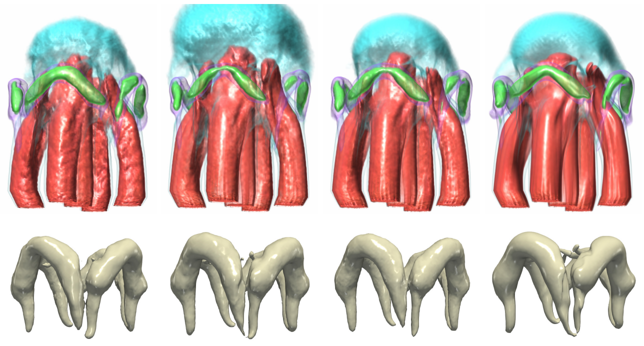

(a) $t = 4$ (b) $t = 2$ (c) $t = 0$ (d) GT

Fig. 4: Volume and isosurface rendering results under different t with $s = 8$ using the five jets data set. The chosen isovalue is −0.2.

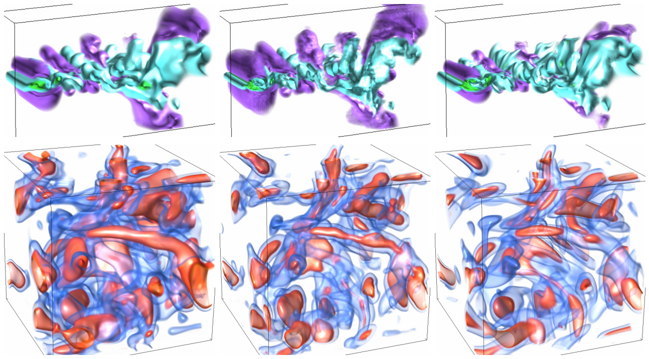
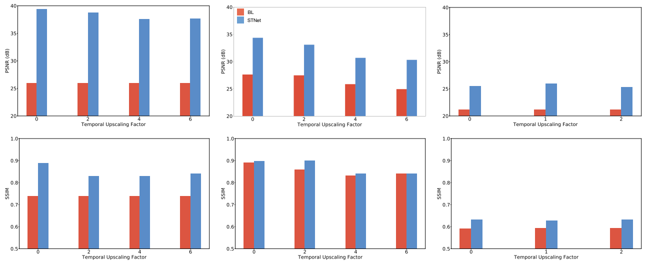

(a) BL (b) STNet (c) GT

Fig. 5: Volume rendering results with $s = 8$ and $t = 0$. Top and bottom: half-cylinder (640) and vortex.

With $s = 8$, the rendering results are displayed in Figures 4 and 5. As we can see, STNet can reach a satisfactory quality with $t = 0$ for the five jets data set. Beyond that, volume rendering results reveal obvious noises and artifacts. As for the half-cylinder (640) and vortex data sets, both STNet and BL cannot produce high-fidelity volume rendering results. Moreover, the quantitative results are shown in Figure 6. STNet outperforms BL in terms of PSNR and SSIM under different t. We also try to increase the training samples and expand the model's depth and width for performance improvement; however, none of these attempts work for the half-cylinder and vortex data sets. To conclude, under the current architecture, upscaling volumes with $s = 8$ is almost infeasible for these data sets (e.g., half-cylinder and vortex) where the spatial structures are complex. For other data sets (e.g., five jets and supercurrent) where the structures are simple and less complicated,



(a) five jets (b) half-cylinder (640) (b) vortex

Fig. 6: Average PSNR and SSIM under different t with $s = 8$. Top and bottom: PSNR and SSIM.

$s = 8$ could still work. In terms of model size, it exhibits a linear relationship with t, from around 25MB for $t = 0$ to around 165MB for $t = 11$, when $s = 4$. There is almost no additional cost on the model size when s switches from 4 to 8 with the same setting for t.