VCNet: A Generative Model for Volume Completion

Jun Han, Chaoli Wang

Department of Computer Science & Engineering, University of Notre Dame, Notre Dame, IN 46556, United States

Abstract

We present VCNet, a new deep learning approach for volume completion by synthesizing missing subvolumes. Our solution leverages a generative adversarial network (GAN) that learns to complete volumes using the adversarial and volumetric losses. The core design of VCNet features dilated residual block and long-term connection. During training, VCNet first randomly masks basic subvolumes (e.g., cuboids, slices) from complete volumes and learns to recover them. Moreover, we design a two-stage algorithm for stabilizing and accelerating network optimization. Once trained, VCNet takes an incomplete volume as input and automatically identifies and fills in the missing subvolumes with high quality. We quantitatively and qualitatively test VCNet with volumetric data sets of various characteristics to demonstrate its effectiveness. We also compare VCNet against a diffusion-based solution and two GAN-based solutions.

Keywords: Volume visualization, Generative adversarial network, Data completion

1. Introduction

With the astounding advance of machine learning techniques, visualization researchers have proposed various deep learning-based data generation solutions for scientific visualization, such as super-resolution creation (in the spatial and temporal domains), ensemble generation, and variable translation. However, the task of volume completion is still unexplored. Volume completion aims to recover the damaged, deteriorating, or missing parts of a volume so that the complete volume can be presented. An example is shown in Figure 1. The potential applications of volume completion include recovering data when they are partially damaged and reducing data through only storing a part of voxels. For example, scientific simulations need to save data to disk for post-processing. However, such data may not be completely saved to local storage during transmission due to I/O suspension or network outage. Our approach can recover the incomplete data without rerunning the simulations if this scenario happens.

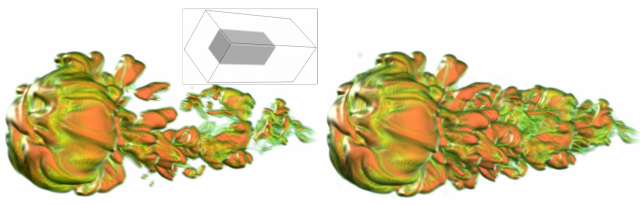


Figure 1: (a) shows the incomplete volume of the argon bubble data set where the cuboid missing subvolume is displayed on the side (same for other figures in the paper). (b) shows our VCNet completion results.

Recovering missing subvolumes poses four key challenges. First, unlike super-resolution and ensemble generation, where *full* information of volumes is provided (even at a low resolution), incomplete volumes only offer *partial* information. Using traditional convolutions (Convs) with a small receptive field will not complete the missing subvolumes, while a large receptive field will lead to high computational cost and memory demand. Second, only applying a series of Convs may not handle complex data sets whose distributions are *composited* (e.g., Gaussian+long-tail). This is because using only one gradient path will prevent the network from converging. Third, the *coherence* between the completed subvolume and its surroundings needs to be considered. Only discerning the completed subvolume can result in low visual quality, leading to pronounced boundary artifacts. Fourth, in *image* completion, the mask can be easily detected through visualization. However, in *volume* completion, due to the transfer function and viewpoint involved, it is difficult to generate such a mask via rendering. However, having such a mask is necessary for volume completion since it offers prior knowledge about which voxels are missing, making the completion task accurate.

To respond, we propose a novel deep learning solution, *volume completion network* (VCNet), to fill in missing subvolumes for volumetric data analysis and visualization. We leverage a *generative adversarial network* (GAN) consisting of a generator and a discriminator. The generator learns how to synthesize the missing subvolume via "seeing" the content from the ground-truth subvolume, and the discriminator scores the realness of the completed subvolume. The core of the generator lies in dilated Conv [1] (which provides a large receptive field without requiring additional computational cost) and long-term connection [2, 3] (which promotes loss into minimum and prevents the generator from falling into unexpected behaviors). The discriminator also judges the coherence between the completed subvolume and its surroundings, and the realness between the completed and ground-truth subvolumes. The training data are from volumes without missing voxels. During inference, given an incomplete volume as input, VCNet first generates a mask based on the Wasserstein distance between complete and incomplete subvolumes and then recovers the input.

We quantitatively and qualitatively test VCNet on several data sets with various characteristics to demonstrate its effectiveness. Furthermore, we compare VCNet against three baselines: gradient vector flow [4], context encoder [5], and global and local completion [6]. Our results show that VCNet achieves the best quality using the data-level metric *peak signal-to-noise ratio* (PSNR), image-level metric *mean opinion score* (MOS), and feature-level metric *isosurface similarity* (IS) [7]. Our contribution is three-fold. First, we propose VCNet, a new generative model that can synthesize missing subvolumes for volumetric data. Second, we design a mask detection algorithm to identify the missing voxels automatically. Third, we perform a comprehensive study to demonstrate the effectiveness of VCNet and investigate its impacting factors.

2. Related work

2.1. Deep Learning for Volume Visualization

Researchers have investigated deep learning techniques for solving volume visualization problems. Such examples include complex structure depiction [8], rendering pipeline replacement [9, 10], ambient occlusion [11], representative time step selection [12], and similarity prediction [13, 14]. Other researchers developed deep learning solutions for creating volumetric scalar and vector data or rendering images in the spatial [15, 16, 17, 18], temporal [19, 20, 21], spatiotemporal [22, 23], image [24, 25, 26], and variable [27, 28] domains. Our work differs from the above works. Instead of focusing on data generation [17, 19, 22, 27], we leverage deep learning solutions to solve the volume completion problem.

2.2. Data Completion

The data completion problem has been studied for more than two decades, which includes two directions: traditional and learning-based solutions. Traditional solutions can be separated into diffusion-based and patch-based approaches. For diffusion-based approaches, Xu and Prince [4] introduced gradient vector flow that estimates the missing voxels by minimizing the Laplacian over the whole data. Ballester et al. [29] proposed a data completion algorithm that jointly interpolates the image's gray levels and gradient directions, then smoothly extends the isophotelines to fill in missing data. Levin et al. [30] built an exponential family distribution over training images to complete image holes. For patch-based approaches, Drori et al. [31] iteratively approximated the unknown regions and composited adaptive image fragments into the image. Barnes et al. [32] proposed PathMatch, a randomized corresponding algorithm that randomly samples some good patch matches and propagates these matches to surrounding areas to keep natural coherence. Huang et al. [33] applied planar structure guidance to estimate planar projection parameters, softly segment the known region into planes, and discover translational regularity within these planes for image completion.

For learning-based solutions, Pathak et al. [5] proposed a context encoder for completing images only for the central regions. Iizuka et al. [6] built a globally and locally consistent image completion framework for arbitrary region completion, where two discriminators were used to guarantee local and global consistency. Liu et al. [34] established partial convolution (PConv) that incorporates a visibility mask into convolutional operation for irregular hole completion. Wang et al. [35] conducted a generative multi-column CNN (GMCNN), which simultaneously processes an incomplete image through three CNNs with different kernel sizes. Yu et al. [36] designed gated convolution (GConv), giving a learnable dynamic feature selection solution for free-form completion.

Our work belongs to the learning-based solution. Unlike the above works, which focus on image completion, we propose a generative model for volume completion and design a mask detection algorithm to discover the missing voxels for accurate inference.

3. VCNet

3.1. Notation

Let us denote $\mathbf{V}^C = \{\mathbf{V}^C_1, \cdots, \mathbf{V}^C_n\}$ and $\mathbf{V}^I = \{\mathbf{V}^I_1, \cdots, \mathbf{V}^I_m\}$ as the *complete* and *incomplete* volumetric data sets, respectively, where n and m are the respective numbers of data samples. For VCNet, $\mathbf{V}^C$ is the *training* set and $\mathbf{V}^I$ is the *inference* set. $\mathbf{V}^C_M = \{\mathbf{V}^C_{M,1}, \cdots, \mathbf{V}^C_{M,n}\}$ is an incomplete volumetric data set generated by $\mathbf{V}^C$ through random masking. $\mathbf{M}^C = \{\mathbf{M}^C_1, \cdots, \mathbf{M}^C_n\}$ is a binary volumetric mask set of $\mathbf{V}^C_M$, where $\mathbf{M}^C_j[v] = 1$ if $\mathbf{V}^C_{M,j}[v]$ is missing at voxel v; otherwise, $\mathbf{M}^C_j[v] = 0$. $\mathbf{M}^I = \{\mathbf{M}^I_1, \cdots, \mathbf{M}^I_m\}$ is a binary volumetric mask set of $\mathbf{V}^I$.

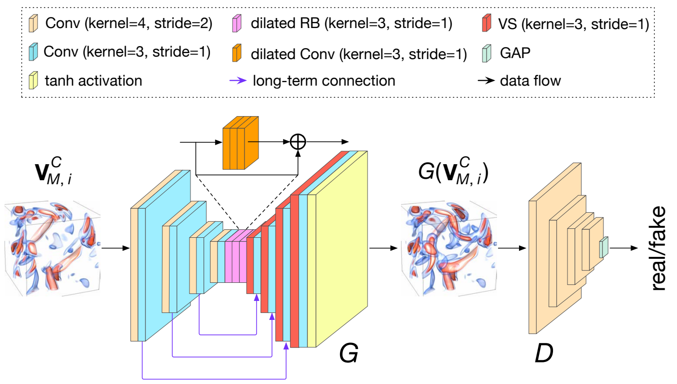


Figure 2: VCNet includes a generator G and a discriminator D. G takes incomplete volumes and synthesizes the missing subvolumes. D accepts the completed volumes as input and determines their realness. Note that D is used during training only.

3.2. Overview

Our VCNet design is adapted from 3D U-Net [37], a popular neural network for image generation and segmentation tasks. Given a volume sample $\mathbf{V}^C_i \in \mathbf{V}^C$, VCNet first randomly masks a subvolume to obtain an incomplete volume $\mathbf{V}^C_{M,i}$. Then taking

$\mathbf{V}^C_{M,i}$ as input, VCNet learns to synthesize the missing subvolume and calculates the error between the synthesized one and GT. To capture the coherence between the synthesized subvolume and its surroundings, we leverage a discriminator to score the volume's realness. During inference, VCNet accepts $\mathbf{V}^I$ as input, estimates the missing voxels, and fills them with high quality. In the following, we introduce the architecture of VCNet, including the generator, discriminator, and design criteria. Then, we provide optimization and inference details for VCNet.

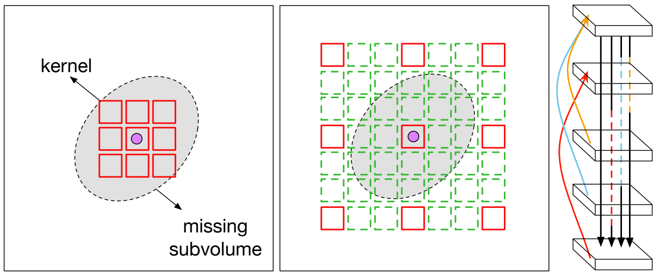
Table 1: Network architecture parameter details for G and D. "ker", "dil", "str", and "out chs" stand for the kernel, dilation, stride, and output channels, respectively.

G					D				
type	ker size	dil	str	out chs	type	ker size	dil	str	out chs
input	N/A	N/A	N/A	1	input	N/A	N/A	N/A	1
Conv+ReLU	4	1	2	32	Conv+ReLU	4	1	2	32
Conv+ReLU	3	1	1	32	Conv+ReLU	4	1	2	64
Conv+ReLU	4	1	2	64	Conv+ReLU	4	1	2	128
Conv+ReLU	3	1	1	64	Conv+ReLU	4	1	2	1
Conv+ReLU	4	1	2	128	GAP	N/A	N/A	N/A	1
Conv+ReLU	3	1	1	128					
Conv+ReLU	4	1	2	256					
Conv+ReLU	3	1	1	256					
dilated RB	3	2	1	256					
dilated RB	3	4	1	256					
dilated RB	3	8	1	256					
VS+Conv+ReLU	3	1	1	128					
VS+Conv+ReLU	3	1	1	64					
VS+Conv+ReLU	3	1	1	32					
VS+Conv+Tanh	3	1	1	1					

3.3. Network Architecture

Generator. The architecture of the generator (G) is sketched in Figure 2. The input to G is an incomplete volume, and the output is a complete one. The core of VCNet lies in applying dilated Conv [1] and long-term connection (LTC) [2, 3]. The design of G follows an encoder-decoder structure. The encoder decreases the input resolution several times to reduce memory storage and computational cost. The decoder restores the deep features to the original resolution of the input using voxel shuffle (VS) [19]. Followed by Iizuka et al. [6], Conv with a stride of two is applied to decrease the resolution in the encoder. We do not use max-pooling since it could lead to blurred texture in the missing subvolumes. We reduce the resolution four times in the encoder. After four rounds of downsizing, three residual blocks (RB) [38] with dilated Conv are applied to provide large receptive fields. Different dilations are utilized in these RBs. In the decoder, we apply four VS layers to upscale the features back to the original resolution. LTC is utilized to bridge the features from the encoder and the decoder. ReLU [39] is applied after each Conv in both the encoder and the decoder. The parameter setting of G is listed in Table 1.

Why dilated Conv? Dilated Conv is a variant of Conv operations, which has been used in image segmentation [1]. As shown in Figure 3, unlike traditional Conv, dilated Conv captures a larger receptive field by applying spread-out kernels with the same number of parameters. Providing a large receptive field is vital for our volume completion task because it allows the network to see a larger subvolume rather than only focusing on the missing subvolume's neighborhoods. Note that deformable Conv [40] can also support a large receptive field, but it requires additional parameters to determine the corresponding voxels involved in the Conv computation. We also use deformable Conv to replace dilated Conv, but no significant improvement is observed. Therefore, we decide to use dilated Conv for designing VCNet.



(a) traditional Conv (b) dilated Conv (c) LTC

Figure 3: (a) and (b) 2D illustrations of the receptive fields of different Conv operations. (c) adding three LTCs (i.e., the red, blue, and orange lines) increases the number of gradient paths to four. The dashed line means the corresponding Conv is not involved in backpropagation.

Why LTC? LTC is a popular technique used in image classification [3] and segmentation [2]. It bridges feature maps between two Conv layers to alleviate the gradient vanishing problem. Adding one LTC, we can rely on two independent paths for gradient computation: one with LTC and another without LTC. If the gradient on one path is zero during backpropagation, the network can still update its trainable parameters by propagating gradient on another path from the later to previous layers. An example is shown in Figure 3 (c). By adding three LTCs in a network with five Conv layers, we increase the gradient paths to four. Without LTC, there is only one computation path (i.e., the black one). Leveraging LTC in the volume completion task is essential since it can promote minimal loss and prevent the network from falling into unexpected behaviors [41].

Discriminator. The discriminator (D) is designed for discerning whether a volume has been completed. The network is based on a fully convolutional network that compresses the volume into a feature vector and predicts a value in $[0, 1]$ to indicate the input's realness. An overview of the network is shown in Figure 2. Specifically, D takes the completed volume as input and utilizes four Conv layers and one global average pooling (GAP) [42] layer to output a single 1D vector. All Conv layers employ a kernel size of four and a stride of two to downsize the volume resolution while increasing the number of feature maps. After four Conv operations, GAP transforms the input into a value, representing the realness probability of the input. The parameter setting of D is listed in Table 1.

Loss function. To guarantee the completed subvolume is realistic and coherent with its surroundings, we consider two loss functions: a *weighted mean squared error* (WMSE) *loss* for closeness to ground truth and an *adversarial loss* [43] for closeness to realism. These two loss functions have been used in image completion [6, 5], which can stabilize the training process and improve network performance.

The WMSE loss only takes into account the completed subvolume for loss computation. It is defined as

$$\mathcal{L}_{\mathrm{rec}}^{G} = \frac{1}{n}\sum_{j=1}^{n} \|\mathbf{M}_j^C \odot (G(\mathbf{V}_{M,j}^C) - \mathbf{V}_j^C)\|_2, \tag{1}$$

where $\odot$ is the voxel-wise multiplication, $\|\cdot\|_2$ is L^2 norm, and n is the number of training samples.

The adversarial losses of G and D are defined as

$$\mathcal{L}_{\mathrm{adv}}^{G} = \frac{1}{n}\sum_{j=1}^{n}[\log D(\mathbf{M}_j^C \odot G(\mathbf{V}_{M,j}^C) + (\mathbf{1} - \mathbf{M}_j^C) \odot \mathbf{V}_j^C)], \tag{2}$$

$$\begin{aligned}\mathcal{L}_{\mathrm{adv}}^{D} &= \frac{1}{n}\sum_{j=1}^{n}[\log D(\mathbf{V}_j^C)] \\ &+ \frac{1}{n}\sum_{j=1}^{n}[\log(1 - D(\mathbf{M}_j^C \odot G(\mathbf{V}_{M,j}^C) + (\mathbf{1} - \mathbf{M}_j^C) \odot \mathbf{V}_j^C))].\end{aligned} \tag{3}$$

Intuitively, D can only discern the completed subvolume; however, this ignores incoherence between the completed subvolume and its surrounding subvolumes. Therefore, in our design, D considers the coherence between the completed subvolume and its surroundings.

Overall, the total loss of G is defined by

$$\mathcal{L} = \lambda_{\mathrm{rec}}\mathcal{L}_{\mathrm{rec}}^{G} + \lambda_{\mathrm{adv}}\mathcal{L}_{\mathrm{adv}}^{G}, \tag{4}$$

where λ_{rec} and λ_{adv} control the importance of $\mathcal{L}_{\mathrm{rec}}^{G}$ and $\mathcal{L}_{\mathrm{adv}}^{G}$.

3.4. Optimization

Missing subvolumes. We consider four *basic* missing subvolumes as either an internal cuboid or a whole x-, y-, or z-stack of slices. VCNet learns to synthesize these missing subvolumes during training. In particular, at each training iteration, VCNet randomly chooses one missing subvolume type from the above four groups, then randomly masks the data as input. During inference, it can complete missing subvolumes with various sizes and shapes (e.g., cuboid, cylinder, hyperboloid, sphere, tetrahedron, and ring). Note that if we only consider an internal cuboid as a missing subvolume during training, VCNet will not complete missing subvolumes with different forms, e.g., a subvolume with a whole x-, y-, or z-stack of slices.

Training procedure. As reported in Iizuka et al. [6] and Han et al. [27], training a GAN model is expensive since the training process needs to go through two networks (G and D) and update gradients of G and D, respectively. Therefore, followed Wang et al. [44], we leverage a two-stage training algorithm (pre-train+fine-tune) to significantly reduce the training cost without sacrificing the performance. The algorithm is shown in Algorithm 1. At the first stage, we treat VCNet as an auto-encoder and only utilize $\mathcal{L}_{\mathrm{rec}}^{\mathrm{G}}$ to optimize VCNet for T_P epochs. At this pre-train stage, VCNet can learn to fill in the missing subvolume, which is close to ground truth but may lack realism. Then, at the second stage, D is added into the training process, and G and D are jointly optimized for T_F epochs.

Algorithm 1 VCNet training algorithm

Require: Initial parameters θ_G and θ_D; numbers of training epochs T_P and T_F for pre-train and fine-tune, respectively; and learning rates α_G and α_D for G and D, respectively.

for $j = 1 \cdots T_P$ **do**
  Sample a set of volumes $\mathbf{V}^C$ from training pool
  Randomly generate masks $\mathbf{M}^C$ and incomplete volumes $\mathbf{V}_M^C$
  Update θ_G using $\mathbf{M}^C$ and $\mathbf{V}^C$ (Equation 1)
end for
for $j = 1 \cdots T_F$ **do**
  Sample a set of volumes $\mathbf{V}^C$ from training pool
  Randomly generate masks $\mathbf{M}^C$ and incomplete volumes $\mathbf{V}_M^C$
  Freeze θ_G
  Update θ_D using $\mathbf{M}^C$, $\mathbf{V}_M^C$, and $\mathbf{V}^C$ (Equation 3)
  Freeze θ_D and activate θ_G
  Update θ_G using $\mathbf{M}^C$ and $\mathbf{V}^C$ (Equation 4)
  Activate θ_D
end for

At this fine-tune stage, with the judgment of D, G can refine the results produced from the pre-train stage toward realism. With the original GAN training algorithm [43], gradients of D can quickly explode because G cannot follow the evolution of D due to random initialization of G and D. This initialization could let G give up generating meaningful results if D evolves much faster than G after several training epochs. Such an imbalanced evolution is due to the disparity between the tasks of G and D (i.e., D is a *classification* task while G is a *generation* task). However, with this two-stage training algorithm, G already has a good initialization that can generate meaningful results through the first stage training. It can refine the results with the feedback from D rather than random initialization from scratch. Moreover, it reduces the training cost since the number of optimization of D is decreased.

Algorithm 2 Mask detection algorithm

Require: An incomplete volume $\mathbf{V}_j^I$; a complete volume $\mathbf{V}_j^C$; and a threshold ϵ.

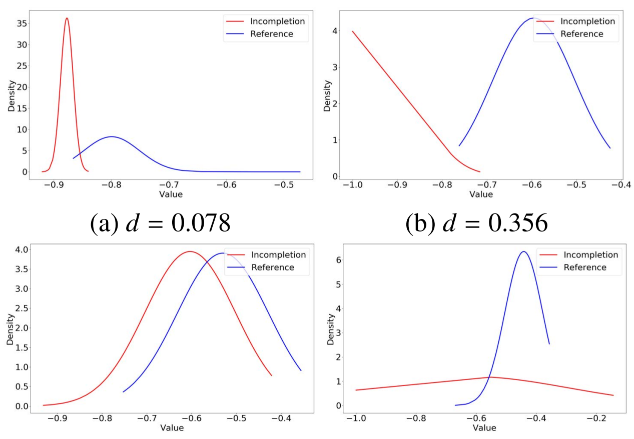
Initialize an empty mask $\mathbf{M}_j$
for each voxel v in $\mathbf{V}_j^I$ **do**
  Sample two $K \times K \times K$ subvolumes $\mathbf{V}_{j,v}^I$ and $\mathbf{V}_{j,v}^C$ where the centers are located at voxel v in $\mathbf{V}_j^I$ and $\mathbf{V}_j^C$, respectively
  Compute the Wasserstein distance d between $\mathbf{V}_{j,v}^I$ and $\mathbf{V}_{j,v}^C$
  if $d > \epsilon$ **then**
    $\mathbf{M}_j[v] \leftarrow 1$
  end if
end for
return $\mathbf{M}_j$

3.5. Inference

Once the training of VCNet converges, we can directly feed $\mathbf{V}^I$ to VCNet to synthesize the missing subvolumes following the equation

$$\mathbf{M}^I \odot G(\mathbf{V}^I) + (\mathbf{1} - \mathbf{M}^I) \odot \mathbf{V}^I. \tag{5}$$

Note that only $\mathbf{V}^I$ is given, and $\mathbf{M}^I$ is unknown. Therefore, we propose a mask detection algorithm to identify the missing voxels and produce the corresponding masks $\mathbf{M}^I$. The algorithm is based on the following assumption: given an incomplete volume $\mathbf{V}_j^I$ and a complete volume $\mathbf{V}_j^C$, the data distributions should exhibit a similar pattern at a voxel v's surrounding subvolume if both $\mathbf{V}_{j,v}^I$ and $\mathbf{V}_{j,v}^C$ are complete. If $\mathbf{V}_{j,v}^I$ is incomplete and $\mathbf{V}_{j,v}^C$ is complete, then the distributions should be different. To verify this assumption, we plot density maps with respect to a local subvolume around a selected voxel, as shown in Figure 4. As we can observe, both maps show a Gaussian distribution for the complete voxels; the only difference is that the mean and variance could vary. However, for the incomplete voxels, the distributions differ from the complete ones. For example, the maps could exhibit an almost straight pattern. The Wasserstein distance is computed to indicate whether the voxel is incomplete. We summarize the mask detection algorithm in Algorithm 2. For each voxel v, we sample two local subvolumes (we set K to 5) of v from $\mathbf{V}_j^I$ and $\mathbf{V}_j^C$, respectively, and compute the Wasserstein distance (d) between these two subvolumes to judge whether v is missing. Once looping through all voxels, the algorithm will return a binary mask $\mathbf{M}_j$, indicating which voxels need to be completed.



(a) $d = 0.078$ (b) $d = 0.356$

(c) $d = 0.073$ (d) $d = 0.254$

Figure 4: The density maps with respect to a local subvolume around a complete voxel (left) and an incomplete voxel (right) for the solar plume (top row) and vortex (bottom row) data sets.

Table 2: The data set, variable, dimension, and training epochs.

data set	variable	dimension ($x \times y \times z \times n$)	T_P	T_F
argon bubble	intensity	$320 \times 128 \times 128 \times 100$	200	50
five jets	intensity	$128 \times 128 \times 128 \times 100$	400	50
solar plume	velocity magnitude	$128 \times 128 \times 512 \times 28$	200	50
supernova	entropy	$128 \times 128 \times 128 \times 60$	800	100
vortex	vorticity magnitude	$128 \times 128 \times 128 \times 90$	400	50

4. Results and Discussion

4.1. Data Sets and Network Training

We tested VCNet using the time-varying data sets given in Table 2. The volume samples were randomly drawn from the sequence. We used 35% of data for training. The remaining 65% of data are for inference. We trained and inferred VCNet using an NVIDIA TESLA V100 GPU with 32GB video memory. PyTorch was used for implementation. In terms of optimization, we initialized VCNet parameters following He et al. [45] and leveraged the Adam optimizer [46] to update parameters. We used one training sample for each mini-batch. The learning rates for G and D are 10^{-4} with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\lambda_{\text{adv}} = 10^{-3}$, and $\lambda_{\text{rec}} = 1$. All these parameters are empirically decided through experiments.

4.2. Results

Baselines. To evaluate VCNet, we implement three baseline solutions for comparison:

- Gradient vector flow (GVF) [4]: As a diffusion-based method, GVF completes missing subvolumes by minimizing the Laplacian over the whole data.

- Context encoder (CE) [5]: CE is a deep learning solution for image completion. Its architecture includes an encoder and a decoder. The encoder includes five Conv layers followed by leaky ReLU and one Conv layer to yield a feature representation with 4,000 neurons. The decoder includes several deconvolutional (DeConv) layers, followed by ReLU for upscaling. WMSE and adversarial losses are leveraged for optimization.

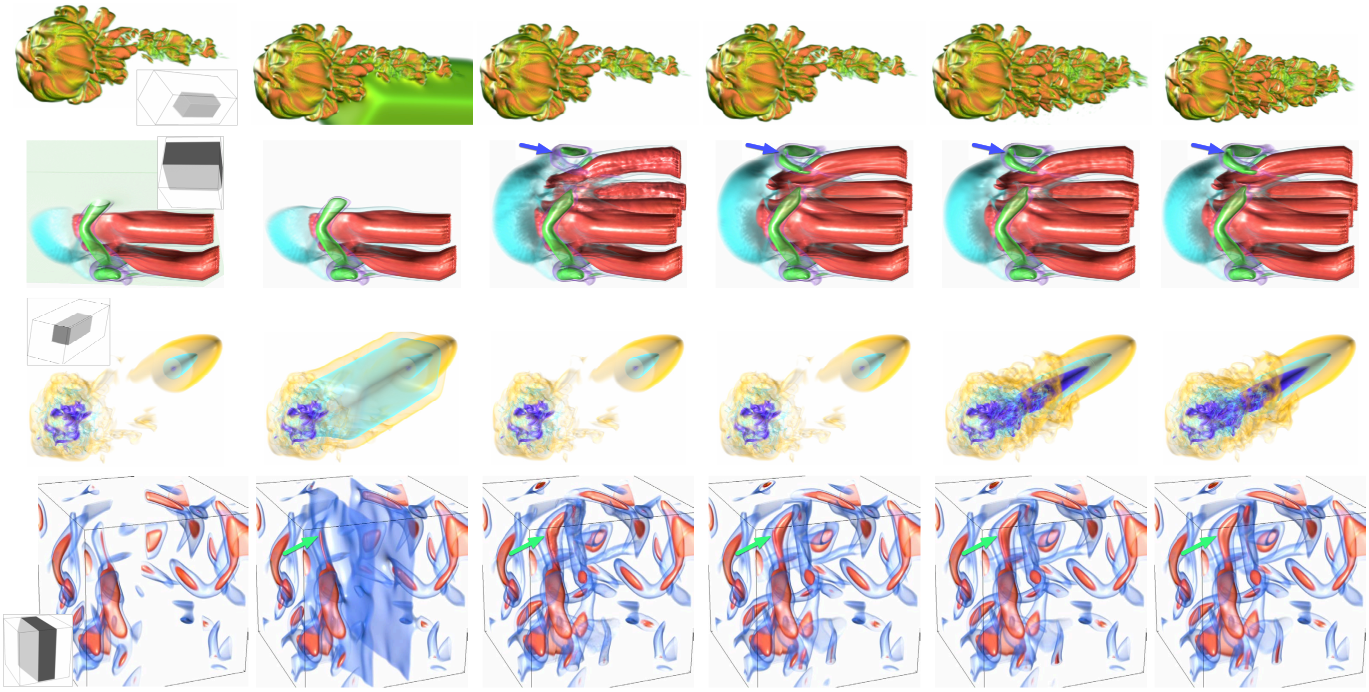
- Global and local completion (GLC) [6]: GLC is a fully convolutional network that includes 11 Conv, four dilated Conv, and two DeConv layers. In addition, it has two discriminators to guarantee local and global consistency, respectively.

We used the same training settings for CE, GLC, and VCNet, namely, the training epochs, optimizer, learning rate, and loss functions (i.e., WMSE and adversarial losses). The only difference between these three deep learning solutions is architecture design.

We also tried PConv [34], GConv [36], and GMCNN [35] as the baselines. However, these solutions are rather deep (PConv), multi-stage (GConv), or multi-column (GMCNN). Applying them to 3D volumetric data sets is difficult due to the limited GPU memory. We tried to reduce the depths, stages, or columns to adapt them into 3D data sets, but the performance was unsatisfactory. Therefore, we only chose CE and GLC as our deep learning baselines.

Unless otherwise mentioned, all visualization results presented for volumes synthesized by VCNet are the inferred results, which are not seen by the network during training. For the same data set, all visualizations follow the same setting for lighting, viewing, transfer function (for volume rendering), and isovalue (for isosurface rendering). In reference to the ground truth (GT) results, we compare our VCNet results against GVF, CE, and GLC. The supplementary video provides the frame-to-frame comparison results.

Evaluation metrics. We compute the data-level PSNR, image-level MOS, and feature-level IS, between the recovered data and GT for quantitative evaluation. We do not use SSIM for image quality assessment because this metric may not differentiate well different methods when the missing subvolumes



(a) input (b) GVF (c) CE (d) GLC (e) VCNet (f) GT

Figure 5: Comparison of volume rendering results. Top to bottom: argon bubble, five jets, solar plume, and vortex.

are small (in this case, all methods will achieve similarly high SSIM values). Note that only the missing subvolumes are involved in the PSNR and IS computation.

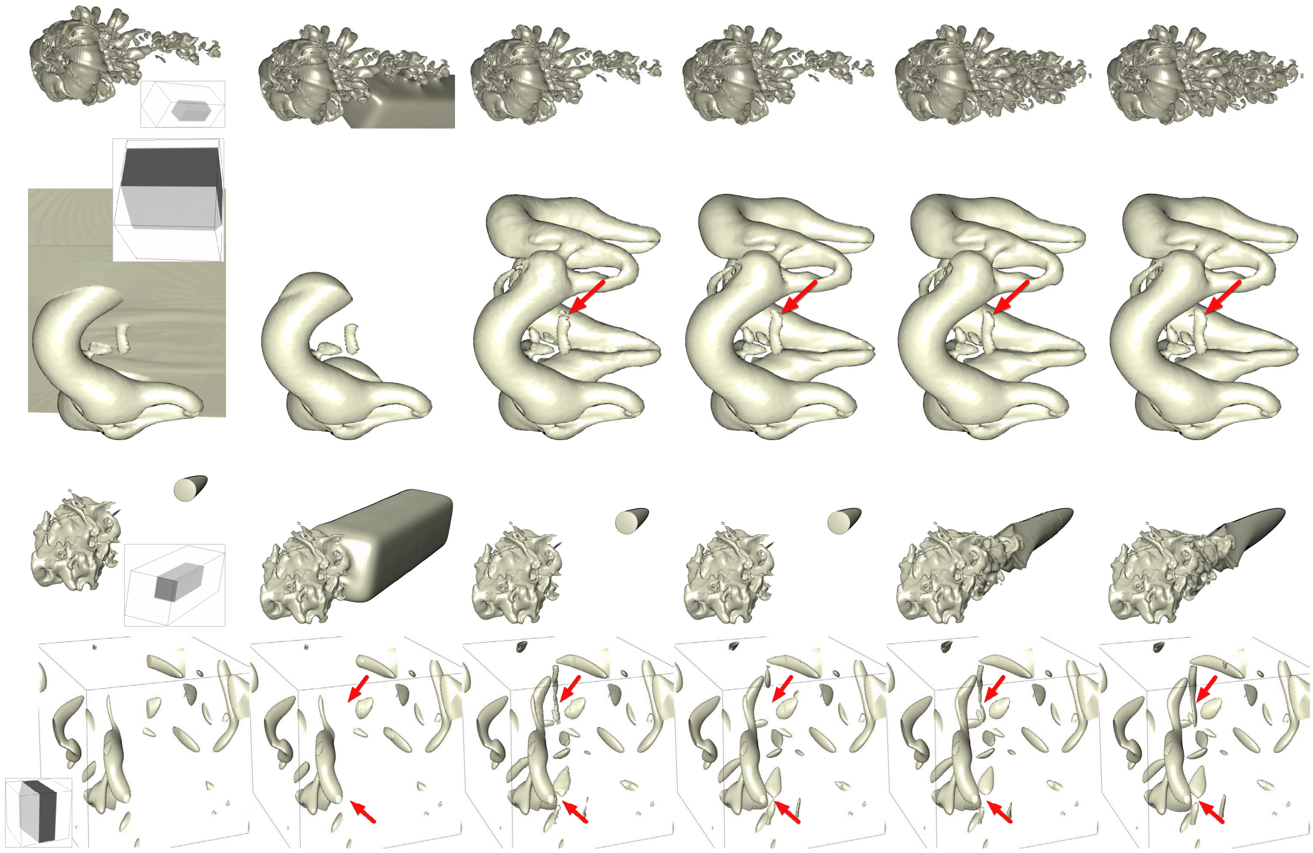
Quantitative analysis. Table 3 reports the average PSNR values for GVF, CE, GLC, and VCNet. VCNet leads to the best PSNR values except for the vortex data set (where the gap between VCNet and CE is only 0.11). Table 3 also gives the average training time per epoch and model size for CE, GLC, and VCNet. It is clear that GLC takes the longest training time since it includes three networks (i.e., one generator and two discriminators) and only downsamples the input twice, while there is no significant difference in the inference time. VCNet requires 120MB to store the model. Although CE is a fully convolutional network, the model size depends on the data set's resolution. It needs to compress the data into a $4,000$-dimensional vector and upscale to the original resolution, which requires a different number of DeConv layers in the decoder based on the input's resolution. Table 4 reports the average IS values for GVF, CE, GLC, and VCNet. Again, VCNet achieves the highest IS value for all data sets.

Qualitative analysis. Figure 5 shows volume rendering results from the volumes completed by GVF, CE, GLC, and VCNet. For the argon bubble and solar plume data sets, VCNet achieves the best completion quality. For example, VCNet completes the argon bubble and solar plume's missing subvolumes. GVF fills in nearly constant values. In contrast, both CE and GLC do not fill in any missing subvolumes (i.e., the volume rendering results are identical to those of the incomplete input volumes). For the five jets data set, GVF cannot repair the missing subvolume, and CE does not synthesize the subvolume with sufficient details. Both GLC and VCNet produce similar results, but taking a close comparison, VCNet synthesizes finer details for the green part (refer to the blue arrows), compared with GT. For the vortex data set, GVF does not complete the missing subvolume, while CE, GLC, and VCNet recover all the missing voxels. However, taking a close comparison, we observe that the result produced by CE includes noises and artifacts, and the result synthesized by GLC lacks coherence with its surrounding subvolumes (refer to the green arrows).

Table 3: Average PSNR (dB), training time per epoch (in seconds), and model size (MB). The best ones are highlighted in bold (same for other tables in the paper).

data set	method	PSNR	train	model size
argon bubble	GVF	13.88	—	—
	CE	23.45	211.61	1, 392.64
	GLC	23.45	2, 291.34	71.9
	VCNet	**37.98**	166.88	120
five jets	GVF	19.71	—	—
	CE	39.55	71.04	1, 146.88
	GLC	43.77	927.68	71.9
	VCNet	**44.64**	34.32	120
solar plume	GVF	13.96	—	—
	CE	20.35	215.34	2, 140.16
	GLC	20.37	3, 072.68	71.9
	VCNet	**41.80**	206.73	120
vortex	GVF	12.46	—	—
	CE	**33.85**	62.02	1, 146.88
	GLC	31.98	817.58	71.9
	VCNet	33.74	30.54	120

Figure 6 shows isosurface rendering results from the volumes completed by GVF, CE, GLC, and VCNet. For each data set, we pick one data sample and one isovalue to genereate the isosurface. VCNet performs the best for the argon bubble and solar plume data sets. For the five jets data set, VCNet and GLC



(a) input (b) GVF (c) CE (d) GLC (e) VCNet (f) GT

Figure 6: Comparison of isosurface rendering results. Top to bottom: argon bubble, five jets, solar plume, and vortex. The chosen isovalues are −0.25, −0.1, −0.4, and 0.1, respectively.

Table 4: Average IS values at selected isovalues.

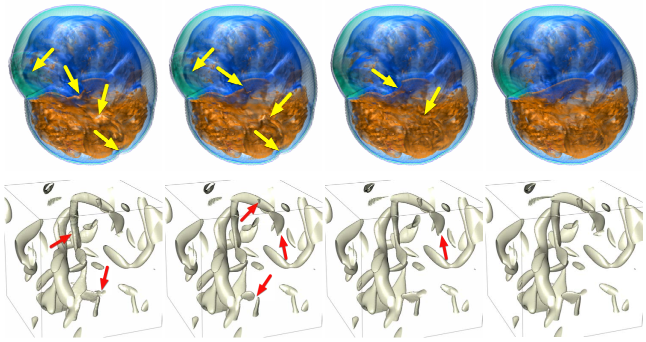
data set (isovalue)	GVF	CE	GLC	VCNet
argon bubble ($v = -0.25$)	0.03	0	0	**0.82**
five jets ($v = -0.1$)	0.05	0.83	0.89	**0.92**
solar plume ($v = -0.4$)	0.02	0	0	**0.88**
supernova ($v = 0$)	0.01	0.58	0.64	**0.67**
vortex ($v = 0.1$)	0.06	0.85	0.83	**0.90**

produce similar results while CE completes the isosurface with some noises and artifacts (see the specular highlights), and GVF only recovers a partial subvolume. As for the vortex data set, VCNet generates more details and preserves better coherence between the incomplete subvolume and its surrounding.

Table 5: Average MOS given by the ten participants.

	volume rendering			isosurface rendering		
data set	CE	GLC	VCNet	CE	GLC	VCNet
five jets	0.50	0.71	**0.76**	0.66	0.73	**0.76**
supernova	0.63	0.69	**0.80**	0.44	0.53	**0.56**
vortex	0.54	0.60	**0.71**	0.56	0.74	**0.79**

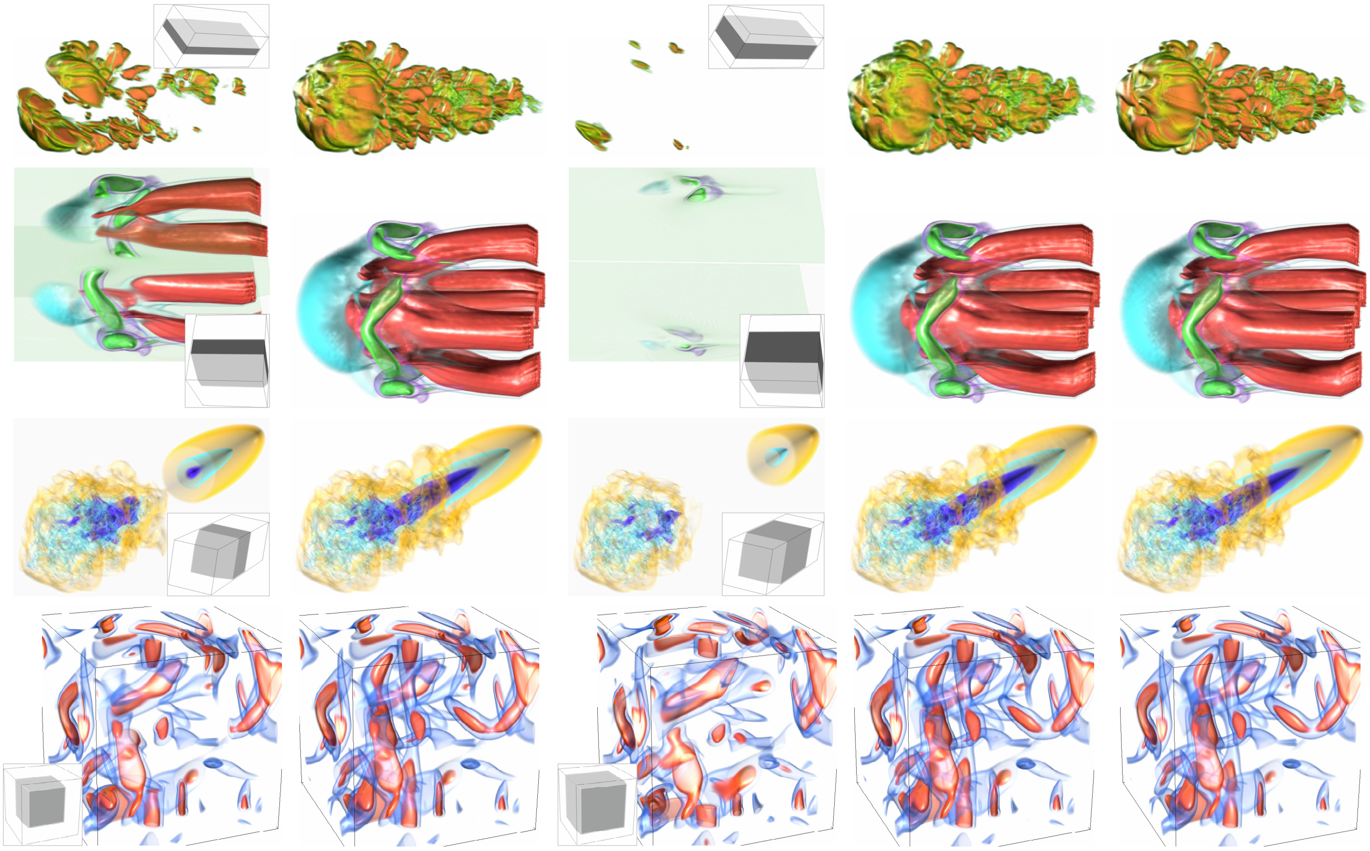
User evaluation. To evaluate the perceptual quality of synthesized volumes, we conducted a user study with volume and isosurface rendering images generated by CE, GLC, and VCNet, compared with GT images. For each rendering option, we chose three data sets for comparison. For each data set, we selected six different volume samples. In total, we collected 108



(a) CE (b) GLC (c) VCNet (d) GT

Figure 7: Highlighted differences from the participants. Top: volume rendering for supernova. Bottom: isosurface rendering for vortex.

$(3 \times 2 \times 3 \times 6)$ image tuples for comparison. For each tuple, we set the left image as rendered from incomplete data, the middle image as synthesized by one of the three methods (CE, GLC, or VCNet with the order randomly shuffled), and the right image as rendered from the GT data. Ten Ph.D. students were recruited to complete the study. All of them major in computer science and have visualization-related backgrounds. These participants were asked to compare the middle image's completion quality with that of the right image by giving a score ranging



(a) input (b) VCNet (c) input (d) VCNet (c) GT

Figure 8: Volume rendering results under different missing ratios. (a) and (c) show 25% and 50% missing ratios, respectively. Top to bottom: argon bubble, five jets, solar plume, and vortex.

Table 6: Average number of highlights given by participants.

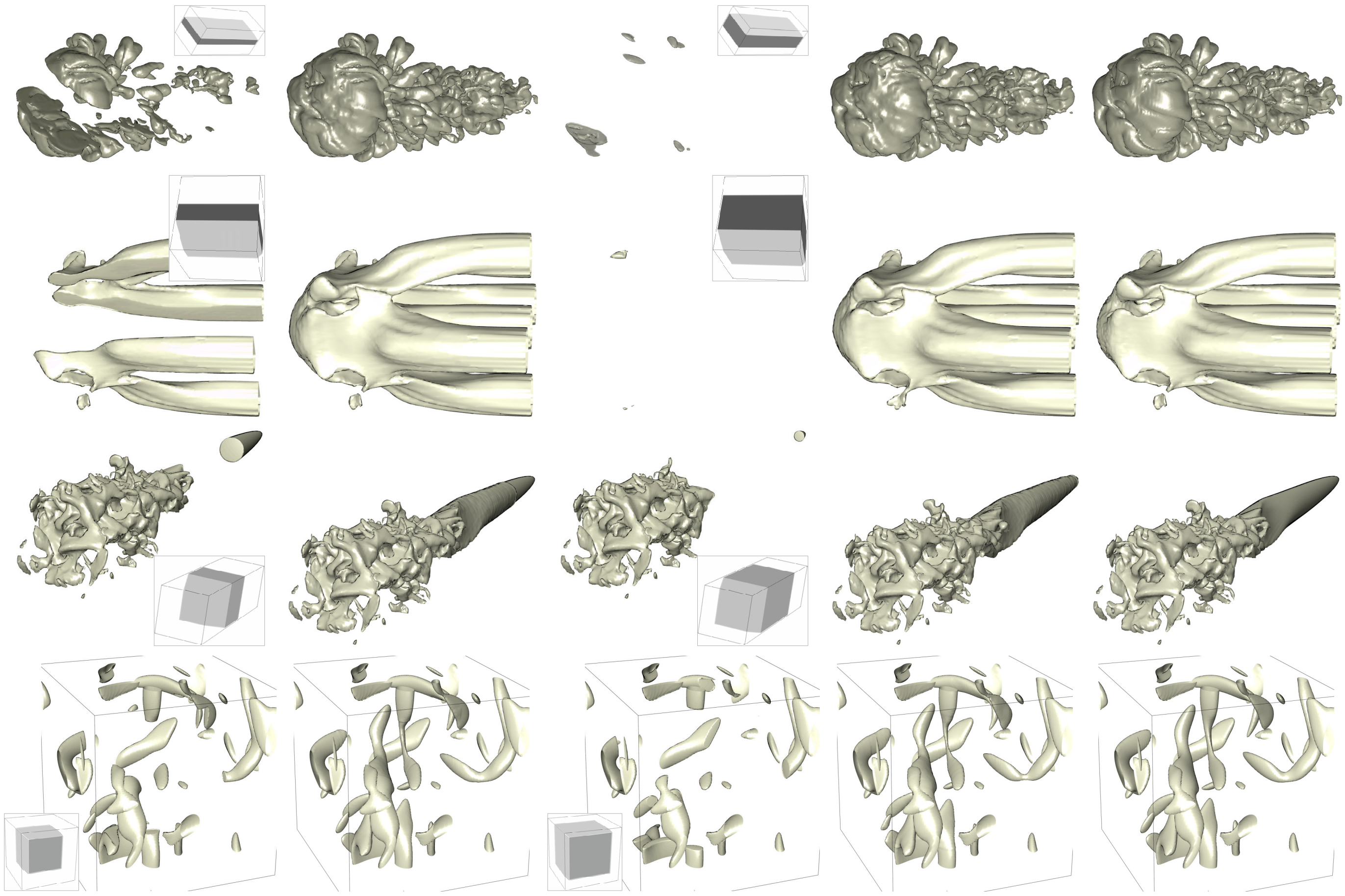
	volume rendering			isosurface rendering		
data set	CE	GLC	VCNet	CE	GLC	VCNet
five jets	2.71	1.96	**1.83**	2.42	2.17	**2.03**
supernova	2.08	1.88	**1.08**	2.92	2.88	**2.67**
vortex	3.04	2.79	**2.54**	2.92	2.13	**1.71**

from 0.0 (most dissimilar) to 1.0 (most similar). Furthermore, they were also asked to highlight, in the middle image, the differences between the middle and right images. We requested up to five differences for each tuple. Sample highlighting results from the participants are shown in Figure 7. Participants were allowed to update the scores during the evaluation, especially at the beginning, when the score calibration is needed. We reminded them that various factors, such as the overall impression, visible content shift, local color consistency, shape preservation, noise level, and coherence between the completed subvolume and its surroundings, should be considered in the evaluation. It took a participant around two hours to complete the study, and each received $20 as compensation. We report the average MOS in Table 5 and average number of highlights in Table 6. As we can see, VCNet achieves the highest MOS and lowest number of highlights for all these three data sets.

Evaluation of missing ratio. To investigate the capability of VCNet in completing different missing ratios, we evaluate VCNet on four different ratios: 12.5%, 25%, 37.5%, and 50%. As shown in Figures 8 and 9, under the missing ratio of 25%, the completed subvolumes are close to the GT for each data set. However, under the missing ratio of 50%, we can observe the differences clearly. For example, the argon bubble's head is inconsistent with the GT. The texture of the five jets' cap is not preserved well. The tail of the solar plume contains some artifacts. The sizes of several red components of the vortex are not consistent with those of GT. Furthermore, in Figure 10, we compare average PSNR values under different missing ratios with different methods. VCNet outperforms CE and GLC for most cases. In addition, when the missing ratio gets larger, the more benefit VCNet can bring. Therefore, depending on the quality need, the maximum missing ratio that VCNet can handle could range from 25% to 50%.

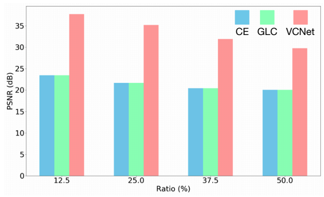
Baseline analysis. As shown in Figures 5 and 6, we observe that (1) GVF does not recover the missing subvolumes for all data sets; (2) the rendering results generated by CE contain noticeable noises and artifacts, while those produced by GLC and VCNet are not that evident; (3) CE and GLC work well for the vortex and five jets data sets but fail for the argon bubble and solar plume data sets. The explanations for these three observations are as follows.

GVF does not complete volumetric data sets with large incomplete subvolumes because it only linearly interpolates the

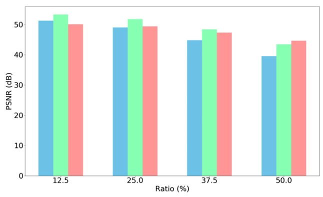


(a) input (b) VCNet (c) input (d) VCNet (c) GT

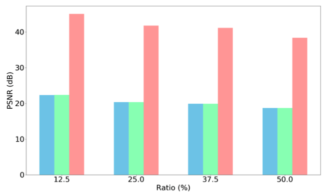
Figure 9: Isosurface rendering results under different missing ratios. (a) and (c) show 25% and 50% missing ratios, respectively. Top to bottom: argon bubble, five jets, solar plume, and vortex. The chosen isovalues are −0.5, 0.4, −0.2, and −0.05, respectively.



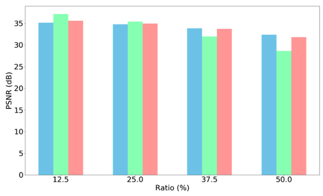
(a) argon bubble



(b) five jets

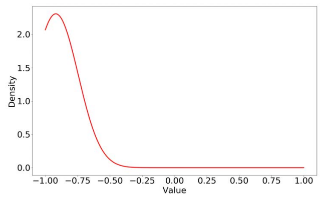


(c) solar plume

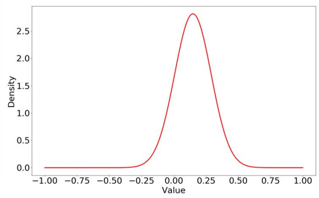


(d) vortex

Figure 10: Average PSNR values under different missing ratios.



(a) argon bubble



(b) five jets

Figure 11: Density maps of different volumetric data sets.

missing voxels by aggregating their neighborhoods. When the missing subvolume becomes large, the neighborhoods can no longer provide enough information for GVF to recover.

The noises and artifacts generated by CE are due to the use of DeConv layers [17]. In CE, it upscales deep features through several DeConv layers but not subsequent Conv layers after each DeConv layer. Without these subsequent Conv layers, the upscaled features are not refined and denoised since the DeConv operation will introduce the checkerboard-like artifact.

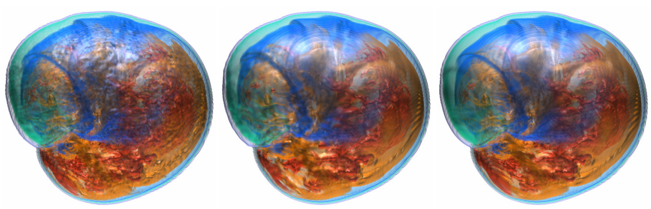
As for CE and GLC's failures on the argon bubble and solar plume data sets, we speculate that it is due to gradient vanishing. To verify this, we compute the average gradient values at different Conv layers in CE, GLC, and VCNet. The average gradients are given in Table 7. For the argon bubble data set, the gradients from Conv 3 to Conv 5 are always 0 for CE and GLC, while VCNet still preserves a small gradient at each Conv layer. As for the five jets data set, all three methods have a non-zero gradient at each Conv layer. These gradient values

Table 7: Average gradient values at different Conv layers under different architectures.

layer	method	gradient (argon bubble)	gradient (five jets)
Conv 3	CE	0	-7.27×10^{-8}
	GLC	0	1.15×10^{-6}
	VCNet	1.24×10^{-6}	-3.72×10^{-8}
Conv 4	CE	0	3.72×10^{-9}
	GLC	0	-1.64×10^{-8}
	VCNet	1.31×10^{-6}	-4.84×10^{-8}
Conv 5	CE	0	7.25×10^{-9}
	GLC	0	1.15×10^{-6}
	VCNet	2.58×10^{-7}	8.19×10^{-9}

confirm our speculation since the learnable parameters in CE and GLC are no longer updated for the argon bubble data set, which leads to the failure. Still, we wonder about the difference between these four data sets. To understand this, we plot their density maps, as shown in Figure 11. It is clear that both five jets exhibit a nearly symmetric distribution, which means if one subvolume is missing, the network can quickly learn to fill in through searching the symmetric counterpart. However, this is not the case for argon bubble. It shows a composed distribution: a Gaussian distribution plus a long-tail distribution. That is, using a forward path in the network is not enough to capture such distributions. Adding multiple forward paths can help the network see more "globally" and merge the results to synthesize the missing subvolume, which is the exact role LTC is playing in VCNet.

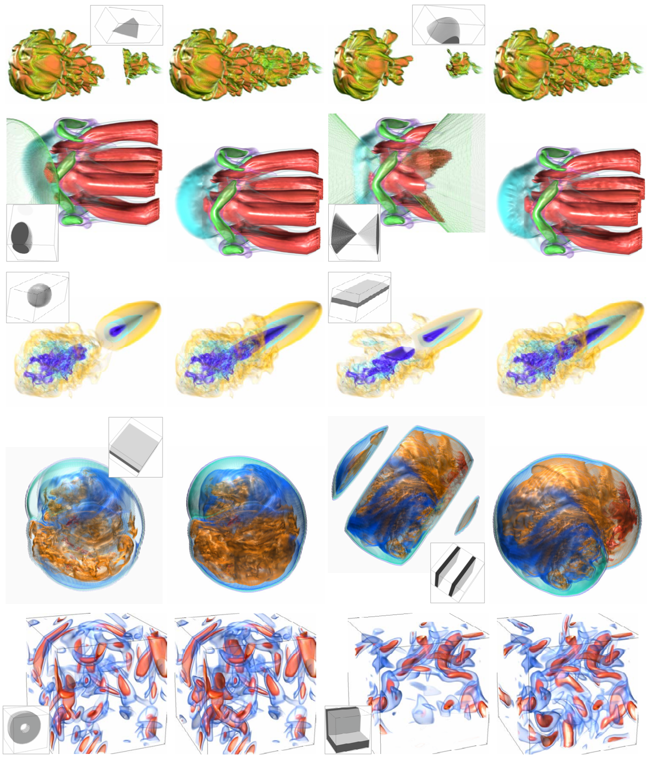
Comparison with lossy compression. One potential application of VCNet is volumetric data reduction. Therefore, we compare our solution against a lossy compression (LC) algorithm [47]. We cull away half of the original volume and utilize VCNet to fill the culled part. We set the same PSNR value (i.e., 44 dB) for both methods for comparison. As displayed in Figure 12, both approaches can recover the overall shape of the supernova, while LC produces more artifacts and noises.



(a) LC (b) VCNet (c) GT

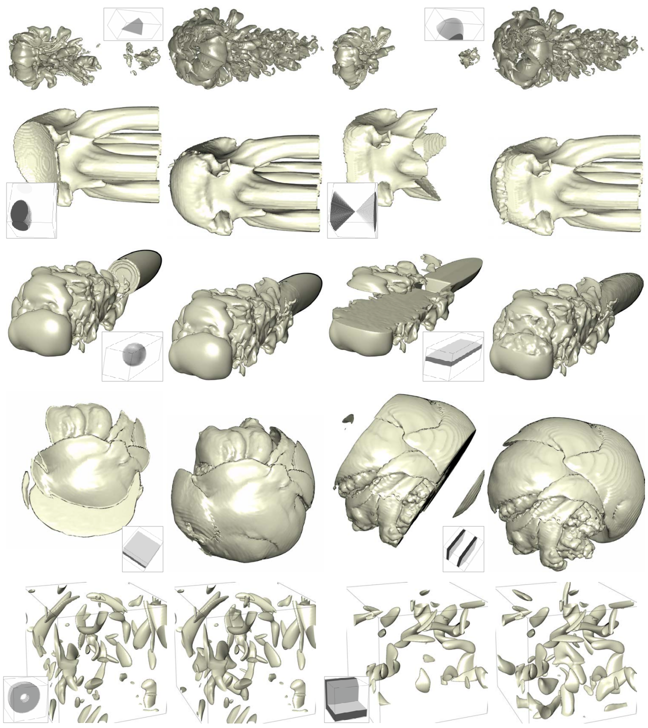
Figure 12: Comparison of volume rendering results with VCNet and LC using the supernova data set.

Robustness evaluation. To study VCNet's robustness in completing different missing subvolumes, we test VCNet for various missing subvolumes (e.g., cuboid, cylinder, hyperboloid, sphere, tetrahedron, and ring) using different data sets. Volume and isosurface rendering results are shown in Figures 13 and 14. The results show that VCNet can handle different missing subvolumes. It can also work well when the input volumes have multiple missing subvolumes.



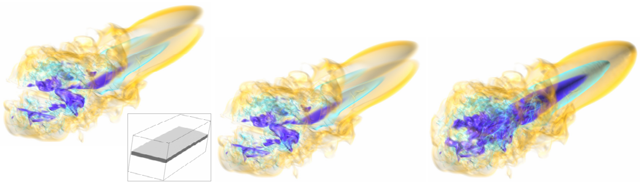
(a) input (b) VCNet (c) input (d) VCNet

Figure 13: Volume rendering results under various missing subvolumes. Top to bottom: argon bubble, five jets, solar plume, supernova, and vortex.



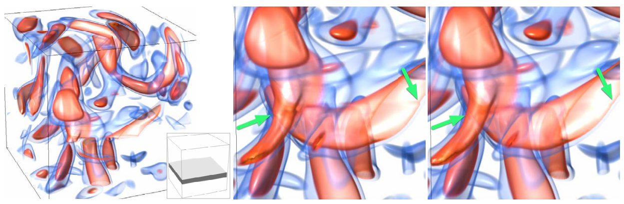
(a) input (b) VCNet (c) input (d) VCNet

Figure 14: Isosurface rendering results under various missing subvolumes. Top to bottom: argon bubble, five jets, solar plume, supernova, and vortex. The chosen isovalues are -0.2, 0.25, -0.8, 0.0, and -0.1, respectively.



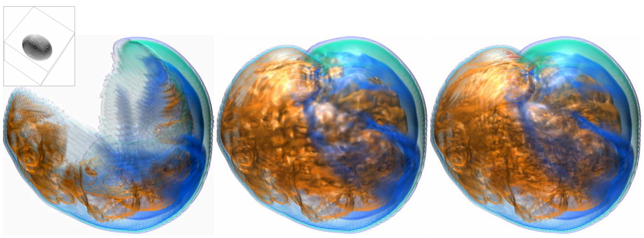
(a) input (b) without (c) with

Figure 15: Volume rendering results with and without LTC using the solar plume data set.



(a) input (b) without (c) with

Figure 16: Zoom-in volume rendering results with and without dilated Conv using the vortex data set.



(a) input (b) VCNet (c) GT

Figure 17: A subpar case of VCNet with the supernova data set.

4.3. Ablation Study

For the ablation study, we investigate the impact of long-term connection and dilated Conv. To investigate the impact of LTC in VCNet, we train VCNet with and without LTC. As shown in Figure 15, without LTC, VCNet cannot recover the missing subvolume for the solar plume data set. These results confirm the effectiveness of LTC in VCNet. To study the usefulness of dilated Conv, we apply traditional Conv to replace dilated Conv in VCNet. As shown in Figure 16, with dilated Conv, the recovered volume of the vortex data set can preserve a better coherence with its surroundings (refer to the green arrows).

4.4. Discussion

While VCNet can complete volumes with various missing subvolumes, it may not satisfactorily synthesize fine details on some specific subvolumes. One example with the supernova data set is shown in Figure 17 where the missing subvolume corresponds to the supernova's center. We can see that VCNet does not generate high-fidelity rendering results, even though the overall shape is well recovered. This is because the surrounding subvolumes may exhibit different structures than the center. Thus, leveraging the surroundings' information does not help fill in the supernova's center seamlessly.

5. Conclusions and Future Work

We have presented VCNet, a novel deep learning framework that synthesizes missing subvolumes for analyzing and visualizing 3D volumetric data sets. Leveraging GAN, VCNet completes different missing subvolumes with varying missing ratios. In terms of volume rendering and isosurface rendering, VCNet achieves better visual quality than GVF and two other solutions based on deep learning (i.e., CE and GLC). In addition to qualitative comparison, quantitative evaluation results using PSNR, MOS, and IS also confirm the effectiveness of VCNet. In the future, we will consider the information of neighboring time steps for preserving temporal coherence. We will also use VCNet to complete large volumetric data sets through multiple GPUs and model parallel.

6. Acknowledgements

This work was supported in part by the U.S. National Science Foundation through grants IIS-1455886, 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] F. Yu, V. Koltun, Multi-scale context aggregation by dilated convolutions, in: Proceedings of International Conference on Learning Representations, 2016.

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

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

[4] C. Xu, J. L. Prince, Gradient vector flow: A new external force for snakes, in: Proceedings of IEEE International Conference on Computer Vision, 1997, pp. 66–71.

[5] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, A. A. Efros, Context encoders: Feature learning by inpainting, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2536–2544.

[6] S. Iizuka, E. Simo-Serra, H. Ishikawa, Globally and locally consistent image completion, ACM Transactions on Graphics 36 (4) (2017) 1–14.

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

[8] H.-C. Cheng, A. Cardone, S. Jain, E. Krokos, K. Narayan, S. Subramaniam, A. Varshney, Deep-learning-assisted volume visualization, IEEE Transactions on Visualization and Computer Graphics 25 (2) (2019) 1378–1391.

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

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

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

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

[13] G. Tkachev, S. Frey, T. Ertl, Local prediction models for spatiotemporal volume visualization, IEEE Transactions on Visualization and Computer Graphics 27 (7) (2021) 3091–3108.

[14] G. Tkachev, S. Frey, T. Ertl, S4: Self-supervised learning of spatiotemporal similarity, IEEE Transactions on Visualization and Computer GraphicsAccepted.

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

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

[17] J. Han, C. Wang, SSR-TVD: Spatial super-resolution for time-varying data analysis and visualization, IEEE Transactions on Visualization and Computer GraphicsAccepted.

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

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

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

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

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

[23] Y. An, H.-W. Shen, G. Shan, G. Li, J. Liu, STSRNet: Deep joint space-time super-resolution for vector field visualization, IEEE Computer Graphics and Applications 41 (6) (2021) 122–132.

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

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

[26] S. Weiss, M. Işik, J. Thies, R. Westermann, Learning adaptive sampling and reconstruction for volume visualization, IEEE Transactions on Visualization and Computer GraphicsAccepted.

[27] J. Han, H. Zheng, Y. Xing, D. Z. Chen, 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) (2021) 1290–1300.

[28] P. Gu, J. Han, D. Z. Chen, C. Wang, Scalar2Vec: Translating scalar fields to vector fields via deep learning, in: Proceedings of IEEE Pacific Visualization Symposium, 2022, accepted.

[29] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, J. Verdera, Filling-in by joint interpolation of vector fields and gray levels, IEEE Transactions on Image Processing 10 (8) (2001) 1200–1211.

[30] A. Levin, A. Zomet, Y. Weiss, Learning how to inpaint from global image statistics, in: Proceedings of IEEE International Conference on Computer Vision, 2003, pp. 305–313.

[31] I. Drori, D. Cohen-Or, H. Yeshurun, Fragment-based image completion, ACM Transactions on Graphics (2003) 303–312.

[32] C. Barnes, E. Shechtman, A. Finkelstein, D. B. Goldman, PatchMatch: A randomized correspondence algorithm for structural image editing, ACM Transactions on Graphics 28 (3) (2009) 24–33.

[33] J.-B. Huang, S. B. Kang, N. Ahuja, J. Kopf, Image completion using planar structure guidance, ACM Transactions on Graphics 33 (4) (2014) 1–10.

[34] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, B. Catanzaro, Image inpainting for irregular holes using partial convolutions, in: Proceedings of European Conference on Computer Vision, 2018, pp. 85–100.

[35] Y. Wang, X. Tao, X. Qi, X. Shen, J. Jia, Image inpainting via generative multi-column convolutional neural networks, in: Proceedings of Advances in Neural Information Processing Systems, 2018, pp. 331–340.

[36] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, T. S. Huang, Free-form image inpainting with gated convolution, in: Proceedings of IEEE International Conference on Computer Vision, 2019, pp. 4471–4480.

[37] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, O. Ronneberger, 3D U-Net: Learning dense volumetric segmentation from sparse annotation, in: Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention, 2016, pp. 424–432.

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

[39] V. Nair, G. E. Hinton, Rectified linear units improve restricted Boltzmann machines, in: Proceedings of International Conference on Machine Learning, 2010, pp. 807–814.

[40] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, Y. Wei, Deformable convolutional networks, in: Proceedings of IEEE International Conference on Computer Vision, 2017, pp. 764–773.

[41] H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein, Visualizing the loss landscape of neural nets, in: Proceedings of Advances in Neural Information Processing Systems, 2018, pp. 6389–6399.

[42] M. Lin, Q. Chen, S. Yan, Network in network, in: Proceedings of International Conference on Learning Representations, 2014.

[43] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Proceedings of Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.

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

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

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

[47] X. Liang, S. Di, D. Tao, S. Li, S. Li, H. Guo, Z. Chen, F. Cappello, Error-controlled lossy compression optimized for high compression ratios of scientific datasets, in: Proceedings of IEEE International Conference on Big Data, 2018, pp. 438–447.