Computers & Graphics (2023)

Contents lists available at ScienceDirect

Computers & Graphics

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

GMT: A Deep Learning Approach to Generalized Multivariate Translation for Scientific Data Analysis and Visualization

Siyuan Yao[a], Jun Han[b], Chaoli Wang[a]

[a]*Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, United States*
[b]*School of Data Science, Chinese University of Hong Kong, Shenzhen, Shenzhen, Guangdong, China*

ARTICLE INFO

Article history:
Received March 27, 2023

Keywords: Scientific visualization, multivariate time-varying data, deep learning, variable-to-variable translation

ABSTRACT

In scientific visualization, despite the significant advances of deep learning for data generation, researchers have not thoroughly investigated the issue of data translation. We present a new deep learning approach called generalized multivariate translation (GMT) for multivariate time-varying data analysis and visualization. Like V2V, GMT assumes a preprocessing step that selects suitable variables for translation. However, unlike V2V, which only handles one-to-one variable translation during training and inference, GMT enables one-to-many and many-to-many variable translation in the same framework. We leverage the recent StarGAN design from multi-domain image-to-image translation to achieve this generalization capability. We experiment with different loss functions and injection strategies to explore the best choices and leverage pre-training for performance improvement. We compare GMT with other state-of-the-art methods (i.e., Pix2Pix, V2V, StarGAN). The results demonstrate the overall advantage of GMT in translation quality and generalization ability.

1. Introduction

As a branch of AI for VIS research, deep learning for scientific visualization has quickly become a hot topic. Within this topic, data generation aims to produce or reconstruct scientific data from their feature representations, low-resolution versions, or lower-dimensional counterparts. Generally speaking, there are four tasks under the umbrella of data generation: *super-resolution*, *compression and reconstruction*, *translation*, and *extrapolation* [1]. In scientific visualization, translation refers to mapping one variable or ensemble sequence to another or one field to another (e.g., a scalar field to a vector field). Compared with the first two tasks, translation is underexplored. Scientists studying various physical phenomena often generate large-scale *multivariate time-varying data* (MTVD) for post hoc analysis and visualization. Due to I/O bandwidth and storage constraints, the output variable sequences from large-scale simulations can only be partially stored in most cases. As the number of variable sequences increases, only a smaller portion of each sequence (i.e., sparsely sampled time steps) can be saved. Translation allows scientists to save only one or a few variable sequences, and the remaining variable sequences can be restored from the trained neural network. This alternative implies direct data reduction (only one or a few variable sequences are needed). It also provides an advantageous point to compress the variable sequence that must be saved for further reduction (fewer sequences must be compressed).

Existing translation works include translating between scalar variables [2, 3], translating from scalar fields to vector fields [4, 5], and translating from an input field to another that satisfies specific properties or constraints (e.g., generating divergence-free vector fields [6]). These works have successfully produced faithful translation results allowing scientists to thoroughly investigate the underlying MTVD. These deep learning models can also disentangle complex variable relationships from the same simulation for further analysis.

However, all these works tackle variable pairs one at a time,

ignoring the critical *scalability* or *generalizability* challenge. Taking the ionization data set, for example, the neural network only takes one pair of source (e.g., H) and target (e.g., H+) variables in the training or inference stage for translation (i.e., H→H+). Looping over different variable pairs would require network retraining (i.e., the network trained over H→H+ would not work for He→He+) or suffer from the downgrade of translation quality (i.e., the network trained over H→H+ would work less effectively for H→He). One can employ the brute-force solution to separately train the network $m(m-1)$ times to learn all translations among m variables to obtain the best results. Unfortunately, this practice becomes increasingly inefficient and ineffective as variables increase. It is typical for a scientific simulation to produce many scalar and vector variable or ensemble sequences. For example, a turbulent combustion simulation can generate dozen chemical species reacting to the combustion process. Therefore, a practical deep learning solution for analyzing MTVD should enable *generalized* variable-to-variable translation (e.g., going beyond *one-to-one* to support *one-to-many* or *many-to-many*).

We present a novel deep learning approach to *generalized multivariate translation* (GMT) for scientific visualization. Our work is inspired by StarGAN [7], which unifies generative adversarial networks (GANs) for multi-domain image-to-image translation. In the context of MTVD, we aim to take in training data of multiple variables and learn the translation between all variables using the same neural network. We adopt one-hot vectors as labels to represent variable information and explore different injection strategies to design a flexible network for learning variable translation.

Like V2V [2] and Scalar2Vec [5], we assume that the same *variable selection* step precedes the *variable translation* step. Variable selection first utilizes U-Net [8] to extract features from variables and then projects the learned features to a 2D space via t-SNE [9] for estimation of variable similarity. Variables with the same footprint or share proximity in the t-SNE projection are deemed suitable for the subsequent translation. The generalization ability of GMT enables domain scientists to process multiple variable pairs under the same framework simultaneously. Moreover, handling multiple pairs in one architecture will likely boost generalization performance. To achieve this, we conjecture that GMT can learn information from different domains (i.e., variables in our context) and utilize the knowledge from other variables to enhance the synthesized quality of one variable.

We summarize the contributions of our work as follows. First, we propose GMT, the first of its kind in scientific visualization, to solve the generalized variable translation problem. As deep learning for scientific visualization research is in full swing, addressing the fundamental and yet often overlooked challenge of *generalization* is imperative for further technique advance and solution adoption. Second, our GMT improves generalizability by leveraging adaptive instance normalization to control the translation outputs. We investigate different loss functions and injection strategies to achieve the best overall variable-to-variable translation quality. Third, we employ a pre-training and fine-tuning scheme to improve translation performance. Fourth, we thoroughly evaluate GMT and demonstrate its advantages over state-of-the-art solutions (i.e., Pix2Pix, StarGAN, V2V). The results show that GMT outperforms existing solutions while taking much less overall training time than StarGAN and V2V.

2. Related Work

Deep learning for scientific data generation. Thanks to the great advances of deep learning solutions in image and language generation tasks, the visualization community has started to explore this direction for scientific data generation. Examples include spatial [10, 11, 12] and temporal [13, 14] super-resolution generation for scalar and vector field data. Han et al. [15] proposed STNet, an end-to-end generative solution for upscaling low-resolution scalar volumetric data into high-resolution in both spatial and temporal spaces. An et al. [16] developed STSRNet, a deep learning framework that generates space-time super-resolution for vector field visualization. Engel and Ropinski [17] designed DVAO, a U-Net [8] based solution for predicting the corresponding ambient occlusion volume given the opacity volume. Lu et al. [18] presented neurocomp, a coordinate-based network [19] that uses a set of fully-connected layers to reconstruct volumetric data from its coordinates via implicit neural representation. Jakob et al. [20] studied convolutional neural networks (CNNs) for spatially upscaling flow maps. Han et al. [2] designed V2V, a three-stage pipeline for variable translation using representation learning and GAN. Gu et al. [5] developed Scalar2Vec, a framework that utilizes k-complete bipartite translation network (kCBT-Net) [21] to translate scalar fields to velocity vector fields. Shi et al. [22] proposed GNN-Surrogate that utilizes a hierarchical and adaptive graph neural network (GNN) to predict simulation outputs on irregular grids given a set of simulation parameters. Farokhmanesh et al. [3] investigated the applicability of V2V to meteorological reanalysis data and presented an algorithm for efficient V2V transfer by considering groups of parameter fields in the transfer. We aim to synthesize MTVD using deep learning techniques like the above works. However, unlike closely related works V2V [2] and Scalar2Vec [5], the proposed GMT solution can *simultaneously* handle *multiple pairs* of variable translation without network retraining and fine-tuning.

Multi-domain translation. Multi-domain translation aims to translate data from different domains through a unified framework. For instance, Liu and Tuzel [23] introduced coupled GAN (CoGAN) that learns a joint distribution of multi-domain images. Liu et al. [24] built a joint distribution in cross-domains by leveraging images from the marginal distributions in individual domains and combining variational autoencoder and CoGAN. Choi et al. [7] proposed StarGAN, a generative solution for synthesizing images with different facial expressions. Huang et al. [25] proposed multimodal unsupervised image-to-image translation (MUNIT) that disentangles images into content and style for multimodal translation. Brock et al. [26] controlled the image generation process by injecting class embedding into the network architecture. Hao et al. [27] manipulated video generation through sparse motion trajectories provided by users.

Instead of focusing on the *image* or *video* domain, our GMT work aims to provide a single framework for multiple variable translations in the *scientific data* domain. The proposed GMT network borrows the StarGAN architecture by treating variables as domains. Besides, we designed a new mechanism that allows users to control the generated variable.

3. GMT

This section first provides an overview of GMT, then discusses the design and experiment of different loss functions and injection strategies, and finally offers architecture and pre-training details.

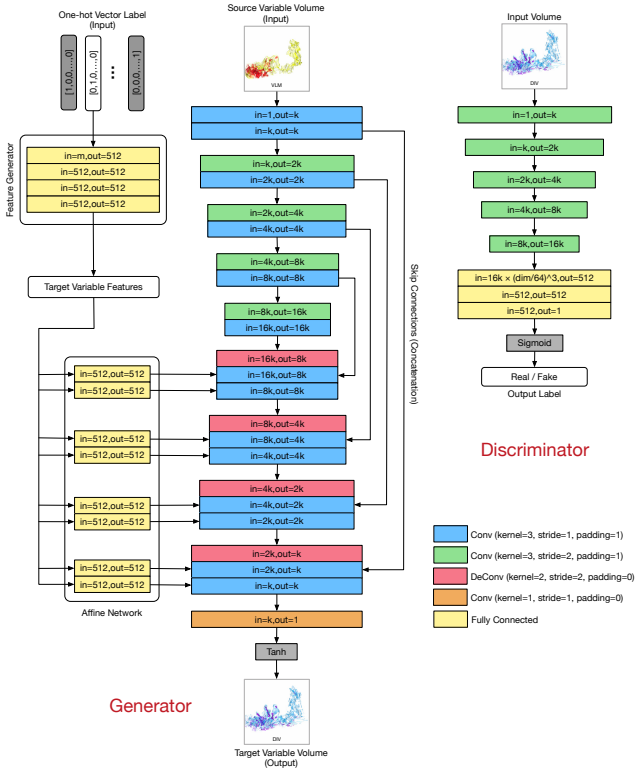


Fig. 1: The overview of our GMT framework.

3.1. Overview

As shown in Figure 1, given a source variable, we leverage a set of convolutional (Conv) layers as an encoder to extract its hidden representation. At the same time, a *one-hot label* with m components, where m is the number of variables used, is fed into the feature generator, which consists of several fully-connected (FC) layers to learn a dense representation of the target variable. The dense representation adjusts the affine parameters in the affine network. In the decoding part of the generator, the outputs of every two Conv layers following a deconvolutional (DeConv) layer are modified by the *adaptive instance normalization* (AdaIN) [28] using the affine parameters. Consequently, the dense representation of the target variable is merged with the representation of the source variable during the decoding process. Finally, the synthesized output volume of the target variable is produced. Besides, we remove residual blocks in the StarGAN model, which is ineffective in our practice. Instead, we add *skip connection* [8] between different Conv layers to improve the generation quality. These long skip connections help the decoder use information from earlier encoder layers to produce fine-grained and more accurate results.

3.2. Loss Functions

To optimize GMT, we investigate different types of loss functions, discussed as follows.

Mean squared error (MSE). MSE estimates the difference between the predicted and ground-truth (GT) variables at each voxel. MSE is defined as

$$\mathscr{L}_{\text{MSE}} = \sum_{i}^{N} (v_i - \hat{v}_i)^2, \tag{1}$$

where N is the number of voxels in the volume, v_i is the GT volume, and $\hat{v}_i$ is the synthesized volume.

Structural dissimilarity index (DSSIM). DSSIM [29] was originally designed for image similarity analysis. We adopt it to calculate the dissimilarity between two volumes based on their constituting subvolumes. DSSIM is defined as

$$\mathscr{L}_{\text{DSSIM}} = 1 - \frac{(2\mu_v\mu_{\hat{v}} + c_1)(2\sigma_{v\hat{v}} + c_2)}{(\mu_v^2 + \mu_{\hat{v}}^2 + c_1)(\sigma_v^2 + \sigma_{\hat{v}}^2 + c_2)}, \tag{2}$$

where c_1 and c_2 are small constants for numerical stability, μ is the mean, σ^2 is the variance, and $\sigma_{v\hat{v}}$ is the covariance.

Derivative. In addition to considering the constraints in the original data space, we take the first-order derivative into account. The derivative loss is defined as

$$\mathscr{L}_{\text{DER}} = \sum_{i=1}^{N} \Big[\Big(\frac{\partial v_i}{\partial x} - \frac{\partial \hat{v}_i}{\partial x}\Big)^2 + \Big(\frac{\partial v_i}{\partial y} - \frac{\partial \hat{v}_i}{\partial y}\Big)^2 + \Big(\frac{\partial v_i}{\partial z} - \frac{\partial \hat{v}_i}{\partial z}\Big)^2 \Big], \tag{3}$$

where $\frac{\partial}{\partial x}$, $\frac{\partial}{\partial y}$, and $\frac{\partial}{\partial z}$ denote the partial derivatives along the x, y, and z directions, respectively.

Adversarial. Since GMT follows a GAN formulation, we study the adversarial loss for the generator (G) and discriminator (D), respectively. The losses are defined as

$$\begin{aligned} \mathscr{L}_G &= (D(G(v^s)) - 1)^2, \\ \mathscr{L}_D &= \frac{1}{2}(D(v^t) - 1)^2 + \frac{1}{2}(D(G(v^s)))^2, \end{aligned} \tag{4}$$

where v^s is the source variable and v^t is the target variable.

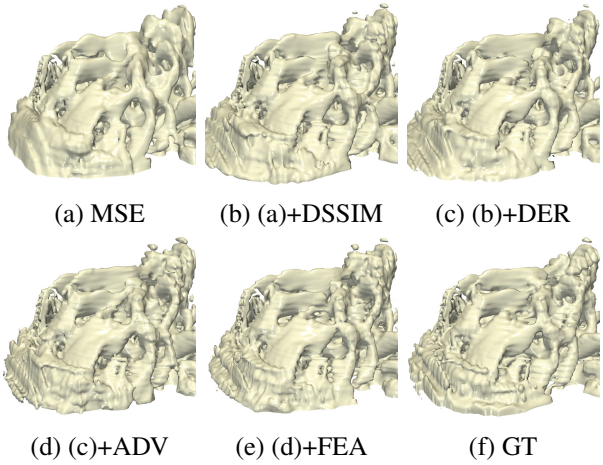
Feature. Finally, we compare high-level features extracted from the prediction and GT using the discriminator (D). The feature loss is defined as

$$\mathscr{L}_{\text{FEA}} = \sum_{i}^{N} (F(v_i) - F(\hat{v}_i))^2. \tag{5}$$

Table 1 compares these loss combinations in terms of PSNR and LPIPS (refer to Section 4.3). Figure 2 displays the isosurface rendering results. Although the quantitative differences are not necessarily significant, we favor the combination of all five loss terms, as Table 1 and Figure 2 indicate that it leads to the best overall results.

Table 1: Performance comparison of loss combinations using the Tangaroa data set (best ones in bold). The weights are determined empirically based on their respective ranges. We report the average PSNR (dB) and LPIPS (for volume rendering) across all time steps.

	(1): MSE		(2): (1)+0.1×DSSIM		(3): (2)+2.0×DER		(4): (3)+10^{-3}×ADV		(5): (4)+10^{-2}×FEA	
	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓
VLM→VTM	45.94	0.0837	45.95	0.0823	45.77	0.0953	**46.21**	0.0823	46.18	**0.0822**
VLM→DIV	49.65	0.1476	49.34	0.1393	50.03	0.1382	49.70	0.1422	**50.35**	**0.1325**
VLM→ACC	29.92	0.3126	30.28	**0.2843**	**30.36**	0.2951	30.29	0.2861	30.25	**0.2843**



(a) MSE (b) (a)+DSSIM (c) (b)+DER

(d) (c)+ADV (e) (d)+FEA (f) GT

Fig. 2: Isosurface rendering comparison under different loss combinations using the Tangaroa (VLM→DIV) data set at time step 1. The chosen isovalue $v = 0.58$. (a) to (f) show the zoom-in views.

3.3. Injection Strategies

To achieve multi-domain translation, we investigate three different injection strategies to manipulate the output of GMT.

Injection I. A straightforward way is to use a one-hot label, which controls what variable to generate by GMT. The one-hot label is defined as

$$\mathbf{l} = [0, \cdots, 1, \cdots, 0], \tag{6}$$

where the i-th element is 1 and the rest are 0. The 1 indicates that GMT aims to synthesize the i-th variable in the target variable pool. Combing the source variable and one-hot label, this injection is defined as

$$v = [v^s, \mathbf{l}], \tag{7}$$

where $[\cdot]$ represents concatenation. We illustrate this injection strategy in Figure 3 (a).

Injection II. Since a one-hot label is binary, it may not provide enough information for translation. A follow-up strategy is to learn a dense representation ($\mathbf{d}$) from the one-hot label through a multilayer perceptron (MLP), i.e., a set of FC layers. Then this representation is injected into the source variable feature ($\mathbf{f}^s$) extracted by Conv layers through AdaIN. The injection formulation is defined as

$$\sigma_{\mathbf{d}} \frac{\mathbf{f}^s - \mu_{\mathbf{f}^s}}{\sigma_{\mathbf{f}^s}} + \mu_{\mathbf{d}}, \tag{8}$$

where μ and σ are the mean and standard deviation. Such an illustration is shown in Figure 3 (b).

Injection III. The final injection strategy is to leverage all available target variables' time steps to control the generation of GMT. That is, given all available target variables' time steps (i.e., $[v_1^t, v_2^t, \cdots, v_k^t]$), we stacked these data together and fed into an encoder to learn a hidden representation $\mathbf{f}^t$. After that, AdaIN is applied to fuse $\mathbf{f}^s$ and $\mathbf{f}^t$. Written in the equation,

$$\sigma_{\mathbf{f}^t} \frac{\mathbf{f}^s - \mu_{\mathbf{f}^s}}{\sigma_{\mathbf{f}^s}} + \mu_{\mathbf{f}^t}. \tag{9}$$

Figure 3 (c) shows how this injection works.

Table 2 shows the performance comparison of these three injection strategies. Figure 4 displays the volume rendering results. As we can observe, Injection II is the winner, achieving the best PSNR and LPIPS scores, even though the rendering image only shows a slight advantage. Table 3 indicates that the training time for Injection II is the shortest, albeit incurring a medium model size. Therefore, we choose Injection II for variable translation.

Table 2: Performance comparison of injection strategies using the ionization data set. We report the average PSNR (dB) and LPIPS (for volume rendering) across all time steps.

	Injection I		Injection II		Injection III	
	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓
H→H+	51.69	0.010	**61.99**	**0.001**	53.02	0.004
H→He	44.54	0.047	**46.43**	**0.039**	45.70	0.045
H→He+	44.07	0.048	**47.08**	**0.036**	46.19	0.040

Table 3: Total training time (H:M:S) and model size (MB) under different injection strategies using the mantle data set.

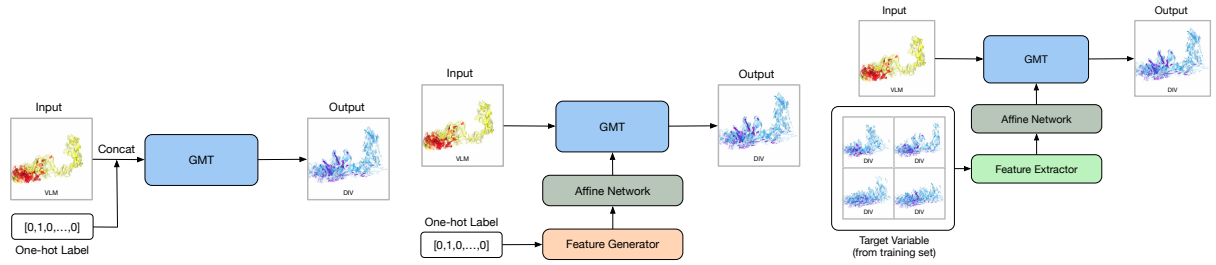
	Injection I	Injection II	Injection III
training time	21:51:12	20:57:34	31:07:14
model size	86.1	92.9	111

3.4. GMT Architecture

The architecture of GMT follows a GAN formulation, where G is an encoder-decoder framework that leverages Conv layers, their transposed versions, and AdaIN, and D is an encoder structure.

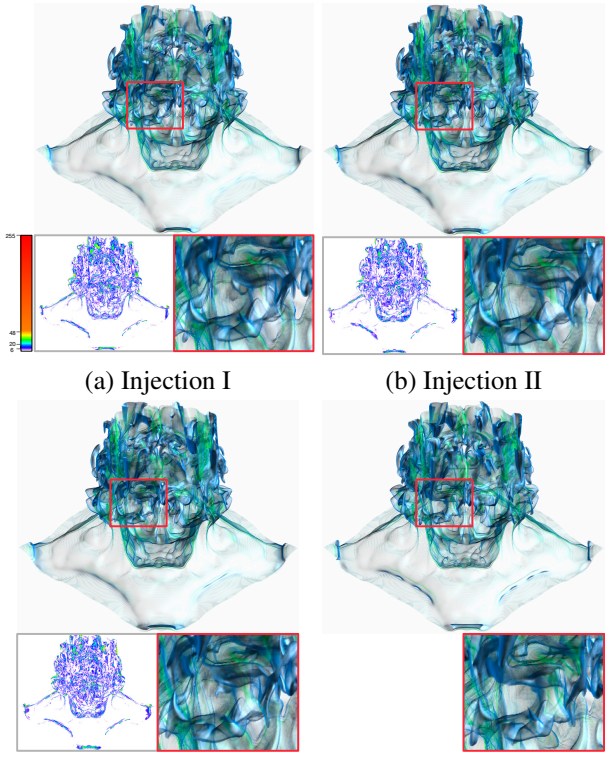
Generator. As shown in Figure 1, in the encoding stage, given the source variable, a block of two 3×3 Conv layers increases the number of channels from 1 to 32 and maintains the resolution as $64 \times 64 \times 64$ (no downsampling). Then we use four consecutive blocks of two 3×3 Conv layers to double the number of channels and downsample the feature maps. In each block, the first 3×3 Conv layer with a stride of 2 is responsible for downsampling the feature maps and doubling their channels. The second 3×3 Conv layer aims to extract higher-level features. Eventually, this leads to a $4 \times 4 \times 4 \times 512$ tensor encoding the given source variable.

Before we decode the encoded representation of the source variable, given the one-hot label that represents the target variable, we use four FC layers as the feature generator to transform



(a) Injection I (b) Injection II (c) Injection III

Fig. 3: Illustration of three strategies of information injection for one-to-many variable translation.



(a) Injection I (b) Injection II

(c) Injection III (d) GT

Fig. 4: Volume rendering comparison under different injection strategies using the ionization (H→He) data set at time step 100.

the label to a feature vector of 512 dimensions. Then, the feature vector is used as the input of an affine network to modulate the outputs of Conv layers through AdaIN for decoding.

In the decoding stage, four blocks of layers upscale the encoded representation back to the original resolution. The first layer of each block is a 2×2 transposed Conv layer, which is responsible for doubling the resolution and halving the number of channels. The output of the first layer is concatenated with the output of the corresponding block in the encoding stage via skip connections. The second and third layers are two 3×3 Conv layers controlled through AdaIN by the feature vector generated according to the given one-hot label. We choose *leaky rectified linear unit* (Leaky ReLU) [30] with $\alpha = 0.2$ as the activation function for all layers in the generator except for the final Conv layer in the decoder. In that layer, we utilize a 1×1 Conv layer to reduce the number of channels back to 1 and a tanh function to regulate the output in $[-1, 1]$.

Discriminator. As shown in Figure 1, we apply five Conv layers with a stride of 2 to downsample the input volume for the discriminator. After that, a MLP is utilized to predict a score that indicates the realness of the volume. Leaky ReLU [30] with $\alpha = 0.2$ is followed after each Conv layer.

3.5. Pre-training

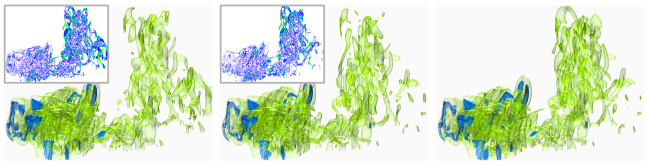
GMT utilizes a pre-training process before fine-tuning to improve training efficiency and effectiveness. First, we pre-train GMT as an autoencoder on the reconstruction task. Then, with the existing knowledge, GMT is fine-tuned on the translation task.

Suppose we directly optimize the trainable parameters in GMT on the translation task without pre-training. In that case, the stability of GMT is largely affected by the number of variables involved in the training. GMT is more likely to be optimized to a local minimum without pre-training, favoring some variable translations but degrading the quality of other translations. Thus, we pre-train GMT on the relatively easy reconstruction task and then utilize the local minimum of the pre-training task as the starting point for fine-tuning.

To optimize GMT during pre-training, we leverage the MSE loss (Equation 1). After sufficient pre-training, GMT can produce lossless reconstruction inferences, which indicates that the encoder of GMT can extract meaningful high-level features from the input volume of any seen variables. Additionally, the feature generator and the affine network have learned the style encoding of different variables in pre-training. As shown in Table 4 and Figure 5, pre-training and fine-tuning generally allow GMT to produce higher-quality inferred data and more visually accurate rendering results than the directly trained GMT model. It is worth noting that fine-tuning a pre-trained GMT only takes half to two-thirds of the total training time (including pre-training time) to achieve the same performance as a directly trained GMT.

Table 4: Performance comparison under different training procedures using the Tangaroa data set. We report the average PSNR (dB) and LPIPS (for volume rendering) across all time steps.

	w/o pre-training		w pre-training	
	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓
VLM→VTM	44.50	0.123	**46.18**	**0.082**
VLM→DIV	48.93	0.171	**50.35**	**0.132**
VLM→ACC	**30.93**	**0.280**	30.25	0.284



(a) w/o pre-training (b) w pre-training (c) GT

Fig. 5: Volume rendering comparison under pre-training using the Tangaroa (VLM→VTM) data set at time step 1.

4. Results and Discussion

4.1. Data Sets and Variable Selection

We experimented with the four MTVD listed in Table 5. The variable set of the combustion data set includes stoichiometric mixture fraction (MF), scalar dissipation (CHI), OH mass fraction (YOH), and heat release (HR). The Tangaroa data set also has four variables: velocity magnitude (VLM), acceleration (ACC), divergence (DIV), and vorticity magnitude (VTM). Note that the Tangaroa data set contains derived variables whose mathematical relationships are known. We use this data set to illustrate how the relationships between different variables influence the difficulty of their translations. For ionization and mantle, we started with seven and six variables, respectively, and followed the solution of V2V [2] to decide the variable subsets suitable for translation. The seven variables of ionization are total particle density (PD), gas temperature (T), H mass abundance (H), H+ mass abundance (H+), He mass abundance (He), He+ mass abundance (He+), H2 mass abundance (H2). The six variables of the mantle are temperature (T), velocity magnitude (VLM), conductivity anomaly (CA), density anomaly (DA), expansivity anomaly (EA), and temperature anomaly (TA). Figure 6 shows the t-SNE projections of these two data sets where each circle represents a single time step. We selected four variables for each based on the proximity of variable trajectories, as determined by the distances between their respective centroids in the t-SNE projection. The excluded variables are shown with strikethroughs in Table 5. We normalized the value range to [0, 1] across all time steps for each data set and variable. The isovalues reported in the figures and tables are based on this range.

Table 5: The variables and dimensions of each data set.

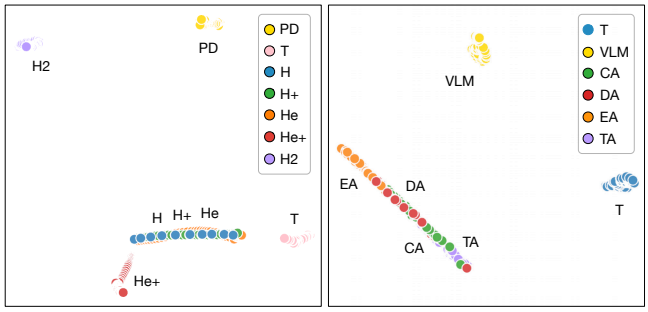
data set	variables	dimension ($x \times y \times z \times t$)
combustion [31]	MF, CHI, YOH, HR	$480 \times 720 \times 120 \times 100$
ionization [32]	~~PD~~, ~~T~~, H, H+, He, He+, ~~H2~~	$600 \times 248 \times 248 \times 100$
mantle [33]	~~T~~, ~~VLM~~, CA, DA, EA, TA	$360 \times 201 \times 180 \times 150$
Tangaroa [34]	VLM, ACC, DIV, VTM	$300 \times 180 \times 120 \times 100$

Table 6: Training time (H:M:S) and model size (MB) for GMT's one-to-many variable translation.

data set	source variable	target variables	training time	model size
combustion	MF	CHI, YOH, HR	18:53:09	92.9
ionization	H	H+, He, He+	20:57:34	92.9
mantle	CA	DA, EA, TA	27:46:06	92.9
Tangaroa	VLM	ACC, DIV, VTM	23:13:19	92.9

4.2. Network Training

For network training, GMT takes full-resolution volumetric data from a given data set and crops the original volume randomly for training. Such a treatment reduces GPU memory consumption and improves training efficiency. The crop size in either dimension should be a multiple of two to the power of the number of times (in our case, four) GMT downsamples the input volume. We recommend a crop size larger than 32 since we downsample the volume five times in the discriminator; otherwise, removing a downsampling layer is necessary. For all data sets, we used the middle 40% time steps for training and the rest of the 60% time steps at both ends for inference. The input volume is always renormalized to [-1,1], corresponding to the range of output volume from a tanh function. We initialized the trainable parameters of GMT with Kaiming initialization [35] and optimized the parameters with Adam algorithm [36]. We used 1×10^{-4} as the learning rate of GMT, and $\beta_1 = 0.9$, $\beta_2 = 0.999$. We tested five loss function settings (Section 3.2) and three injection strategies (Section 3.3) to identify the best choices. Table 6 lists the training time and model size for GMT's one-to-many variable translation. We can see that the model size remains the same across all cases, and the training time is mainly determined by the number of time steps and forms a linear relationship.



(a) ionization (b) mantle

Fig. 6: T-SNE projections for variable selection.

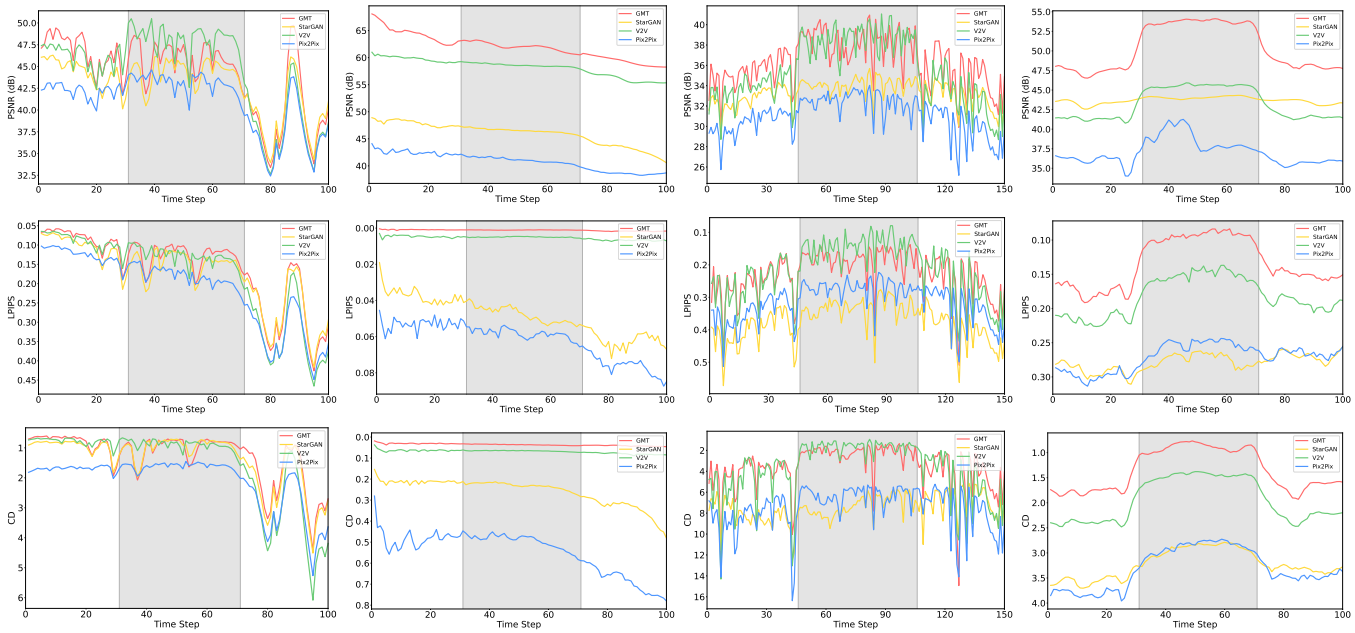
Table 7: Average PSNR (dB), LPIPS (for volume rendering), and CD (for isosurface rendering, refer to Section 4.3) values across all time steps. The data set column also reports the source and target variables as well as the chosen isovalue v.

data set	method	PSNR ↑	LPIPS ↓	CD ↓
combustion (MF→CHI, $v = 0.02$)	Pix2Pix	41.07	0.211	2.08
	V2V	**44.49**	0.177	1.54
	StarGAN	42.84	0.173	1.37
	GMT	44.20	**0.159**	**1.28**
ionization (H→H+, $v = 0.5$)	Pix2Pix	40.82	0.062	0.55
	V2V	58.25	0.005	0.07
	StarGAN	45.97	0.049	0.26
	GMT	**61.99**	**0.001**	**0.04**
mantle (CA→TA, $v = 0.55$)	Pix2Pix	31.03	0.315	7.47
	V2V	35.38	**0.217**	3.79
	StarGAN	33.29	0.376	7.49
	GMT	**36.37**	0.229	**3.57**
Tangaroa (VLM→DIV, $v = 0.58$)	Pix2Pix	37.04	0.271	3.31
	V2V	43.23	0.182	1.95
	StarGAN	43.68	0.278	3.23
	GMT	**50.35**	**0.134**	**1.36**

4.3. Results

Baselines. We compare our GMT with the following state-of-the-art deep learning methods:

- Pix2Pix [37]: Pix2Pix is a framework for paired image-to-image translation. We used the original Pix2Pix architecture for variable translation.



(a) combustion (MF→CHI) (b) ionization (H→H+) (c) mantle (CA→TA) (d) Tangaroa (VLM→DIV)

Fig. 7: Top to bottom: PSNR, LPIPS (for volume rendering), and CD (for isosurface rendering) of synthesized target variables (VTM, H+, TA, and DIV) under Pix2Pix, V2V, StarGAN, and GMT. The chosen isovalues are reported in Table 7. The shaded area denotes the time steps used for network training.

- StarGAN [7]: StarGAN is a unified GAN for multi-domain image-to-image translation. We treated variables as domains and modified StarGAN to work with volume-to-volume translation.
- V2V [2]: V2V is a paired volume-to-volume translation framework. We built and trained GMT with reference to the V2V architecture.

For all methods and across all data sets, we use the same crop size of 64^3 and train each model for 4000 epochs to ensure that the model converges sufficiently. We present the training time comparison for all methods in Table 8, including the time we spent on pre-training GMT for 500 epochs. StarGAN and GMT perform one-to-many variable translation, and Pix2Pix and V2V only perform one-to-one variable translation. Therefore, the total time to train Pix2Pix or V2V for an equivalent one-to-many (three in this case) variable translation task should be roughly tripled.

Table 8: Training time (H) for Pix2Pix, V2V, StarGAN, and GMT.

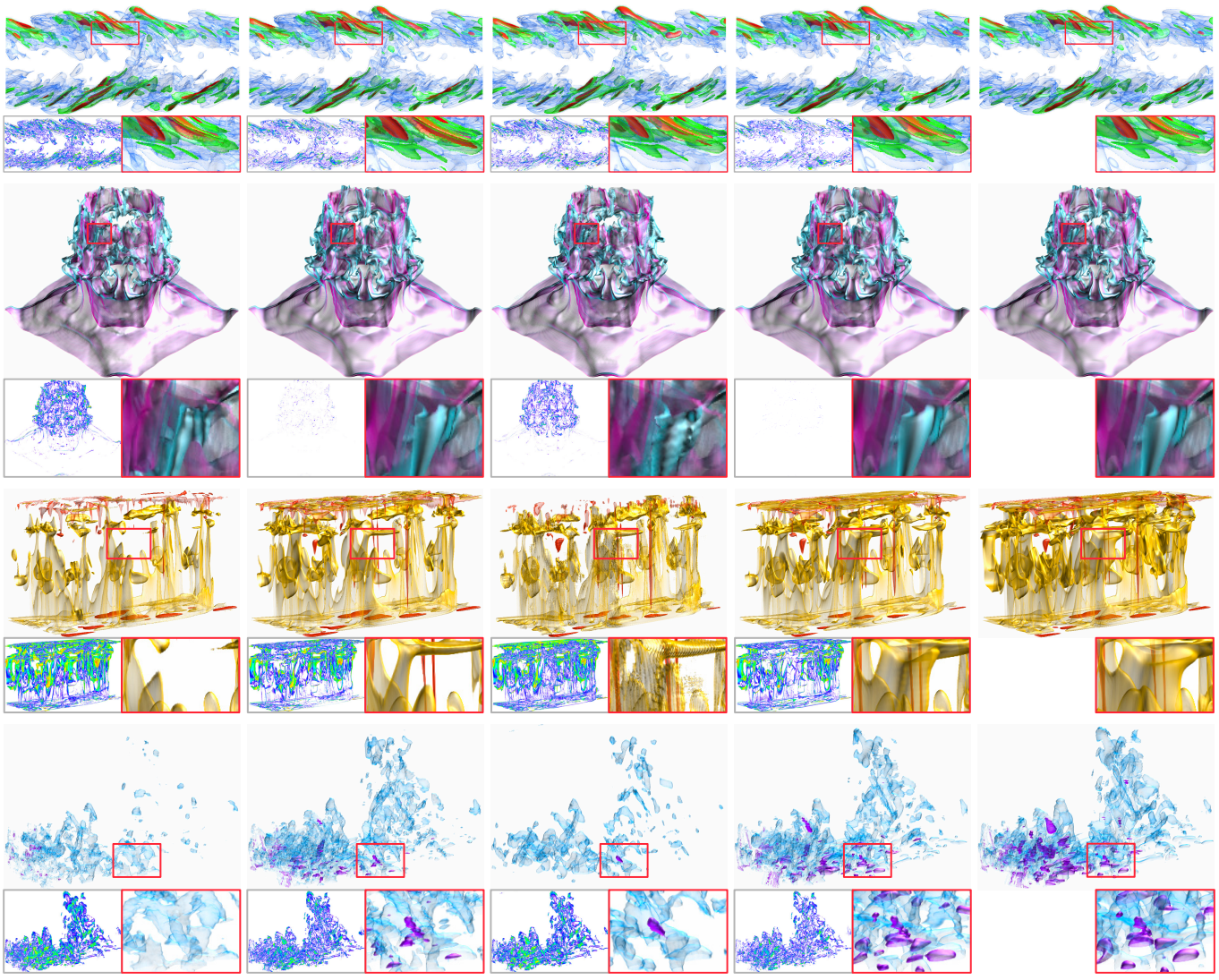
	Pix2Pix		V2V		StarGAN	GMT	
	1-to-1	1-to-3	1-to-1	1-to-3	1-to-3	pre-train	1-to-3
combustion	6.1	18.3	11.7	35.1	26.5	2.5	18.9
ionization	7.3	21.9	14.1	42.3	35.9	4.6	21.0
mantle	10.8	32.4	22.9	68.7	54.7	3.4	27.8
Tangaroa	7.0	21.0	15.5	46.5	35.5	2.3	23.2

Except for the rendering of GT, all visualization results presented in this paper are rendering results from synthesized data. Since the prediction results of a time step furthest from the training set better reflect the performance of each model, we picked a time step at either end of the sequence for the comparison. The accompanying video shows the rendering of all time steps of selected cases.

Except for Figure 2, for closer examination, we calculate rendering differences with respect to the GT rendering in the CIELUV color space. Noticeable pixel differences are mapped to purple, blue, green, yellow, and red, showing low to high pixel-wise differences (refer to the lower-left corner of Figure 4 for the colormap legend). Optionally, we provide zoom-in comparisons as well. Parameter studies of GMT, including training sample, training epoch, and crop size, are presented in the appendix.

Evaluation metrics. For quantitative evaluation, we employ three metrics: *peak signal-to-noise ratio* (PSNR), *learned perceptual image patch similarity* (LPIPS) [38], and *chamfer distance* (CD) [39]. These three metrics evaluate the quality of synthesized data at different levels. At the *data* level, PSNR evaluates the quality of the synthesized data with respect to the original data (higher PSNR means better quality). At the *image* level, LPIPS evaluates the similarity of volume rendering or isosurface rendering images from the synthesized and original data. As a learned metric based on neural networks, it computes a weighted average of the activations at hidden layers to predict relative image similarities, which correlate well with perceptual judgments (lower LPIPS means better quality). Finally, at the *feature* level, CD evaluates the similarity between isosurfaces extracted from the synthesized and original data. It calculates the bidirectional overall vertex-wise Euclidean distance between the two surfaces (lower CD means better quality).

Quantitative comparison. Table 7 reports the average PSNR, LPIPS, and CD values of inferred data across all time steps. GMT is the winner across all metrics and data sets, except for PSNR for combustion (MF→CHI) and LPIPS for mantle (CA→TA). In these cases, GMT is a close second-best behind V2V. V2V is the second best among these four methods, only losing slightly to StarGAN on PSNR for Tangaroa (VLM→DIV). Between Pix2Pix and StarGAN, StarGAN beats Pix2Pix across all metrics for combustion (MF→CHI) and ion-



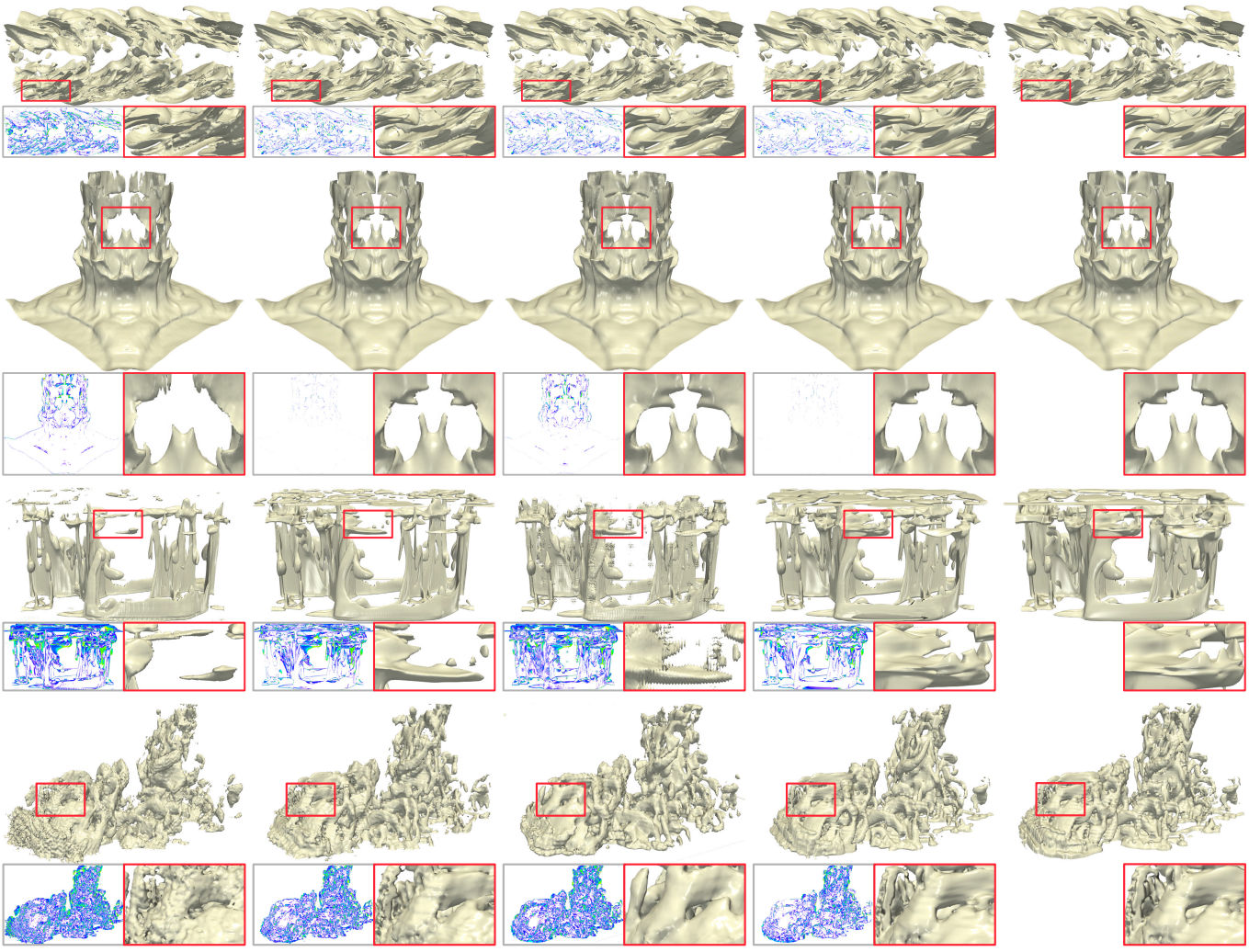
(a) Pix2Pix (b) V2V (c) StarGAN (d) GMT (e) GT

Fig. 8: Volume rendering comparison among different methods. Top to bottom: combustion (MF→CHI), ionization (H→H+), mantle (CA→TA), and Tangaroa (VLM→DIV). The time steps displayed are 6, 100, 150, and 1, respectively.

ization (H→H+). The results are mixed for the other two data sets, with StarGAN winning over Pix2Pix for PSNR, Pix2Pix winning over StarGAN for LPIPS, and a tie for CD. Overall, we can conclude that GMT is the best, followed by V2V, StarGAN, and Pix2Pix.

Figure 7 plots PSNR, LPIPS, and CD values for each time step. For PSNR, GMT is the best for both training and inferred time steps for ionization and Tangaroa, in a strong tie with V2V for combustion and mantle for training time steps but is better for inferred time steps. For LPIPS, similar observations can be made for combustion, ionization, and Tangaroa, while for mantle, V2V outperforms GMT for training time steps, but GMT is in a strong tie with V2V for inferred time steps. For CD, GMT is the best for ionization and Tangaroa, in a strong tie with V2V for training time steps of combustion and mantle, but beats V2V for inferred time steps. For the combustion data set, since CHI has more drastic changes than MF in later time steps, the performance curves exhibit strong spikes, bounded by the sharp dips around time steps 80 and 95. The fluctuation of performance curves for the mantle data set is due to its lack of temporal coherence in the original data. We expect the performance to peak at the training time steps and decline at the inferred time steps (i.e., exhibiting the bell curve), and most of the results support this expectation. Exceptions across all four data sets are the early inferred time steps for PSNR, LPIPS, and CD curves of combustion and ionization. These exceptions may be due to the relatively straightforward data patterns exhibited in the early time steps of the data set.

Qualitative comparison. Figures 8 and 9 show volume rendering and isosurface rendering comparisons. For volume rendering, GMT is the winner for mantle and Tangaroa. For combustion, the results of GMT are very close to those of V2V and StarGAN, but the details generated from GMT are more accurate than the other two. For ionization, GMT and V2V lead to very close results, but the difference images show that GMT is still better than V2V. Overall, Pix2Pix yields the worst results across all four data sets, followed by StarGAN. For isosurface rendering, we can draw similar conclusions.



(a) Pix2Pix (b) V2V (c) StarGAN (d) GMT (e) GT

Fig. 9: Isosurface rendering comparison among different methods. Top to bottom: combustion (MF→CHI), ionization (H→H+), mantle (CA→TA), and Tangaroa (VLM→DIV). The time steps displayed are 6, 100, 150, and 1, respectively. The chosen isovalues are reported in Table 7.

One-to-many variable translation. Both StarGAN and GMT have the capability of achieving one-to-many variable translation. Figures 10 and 11 show their volume rendering and isosurface rendering comparisons using the ionization data set where H is the source variable. We use a time step at one end (time step 100) furthest away from the training time steps, demonstrating the worst-case results. Although the differences are subtle, GMT generates better results than StarGAN for volume and isosurface rendering (refer to the difference images and zoom-in comparisons). The quantitative comparison in Table 9 also indicates that overall, GMT outperforms StarGAN over the three quality metrics. GMT is consistently better across all time steps.

Many-to-many variable translation. The GMT design makes it easy to train all pairs bidirectionally within a single framework. The training time proportionally increases as the number of variable pairs multiplies. For example, given four variables, the number of pairs increases from three (one-to-many) to 12 (many-to-many). The training time of many-to-many variable translation is expected to be four times that of one-to-many.

Figures 12 and 13 show volume rendering and isosurface rendering comparisons using the Tangaroa data set. Again, we use a time step at the other end (time step 1), furthest away from the training time steps, demonstrating the worst-case results. Examining Figure 12 across the rows and columns, we can observe that overall, VLM is the best source variable (i.e., the source leading to the best translation result regardless of the target), and VTM is the best target variable (i.e., the most easily transferrable target irrespective of the source). Checking Figure 13 leads to the same observation.

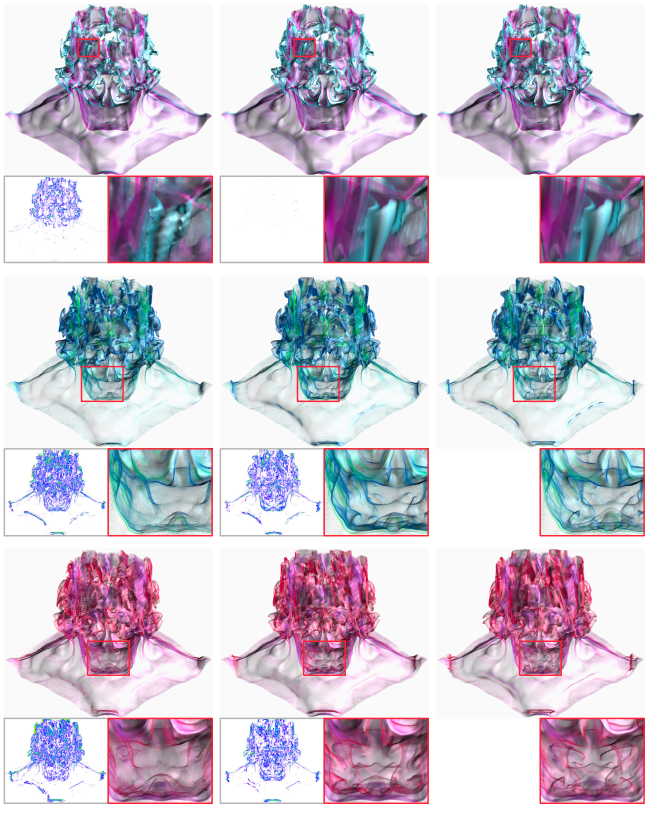
These visual results echo the pairwise LPIPS (for volume rendering) and CD (for isosurface rendering) comparisons reported in Table 10. However, PSNR comparisons at the data level do not agree (for PSNR, VLM is the best source, and DIV is the best target). Table 10 also shows that the bidirectional translation is not symmetric. For example, VLM→DIV achieves a PSNR value of 47.75 dB, while DIV→VLM only leads to a PSNR value of 34.72 dB. For LPIPS, the widest gap happens between VTM→DIV and DIV→VTM, and that for

Table 9: PSNR (dB), LPIPS (for volume rendering), and CD (for isosurface rendering) values at time step 100 and their averages across all time steps for one-to-many variable translation using the ionization data set. For CD, $v = 0.5$, 0.7, and 0.1 for H+, He, and He+, respectively.

	time step 100						average across all time steps					
	PSNR ↑		LPIPS ↓		CD ↓		PSNR ↑		LPIPS ↓		CD ↓	
	StarGAN	GMT	StarGAN	GMT	StarGAN	GMT	StarGAN	GMT	StarGAN	GMT	StarGAN	GMT
H→H+	40.58	**58.25**	0.067	**0.002**	0.48	**0.05**	45.97	**61.99**	0.049	**0.001**	0.26	**0.04**
H→He	33.66	**34.10**	0.135	**0.090**	1.23	**1.10**	41.21	**46.43**	0.079	**0.039**	0.51	**0.30**
H→He+	32.95	**34.11**	0.158	**0.090**	1.50	**1.11**	39.65	**47.08**	0.103	**0.036**	1.04	**0.51**

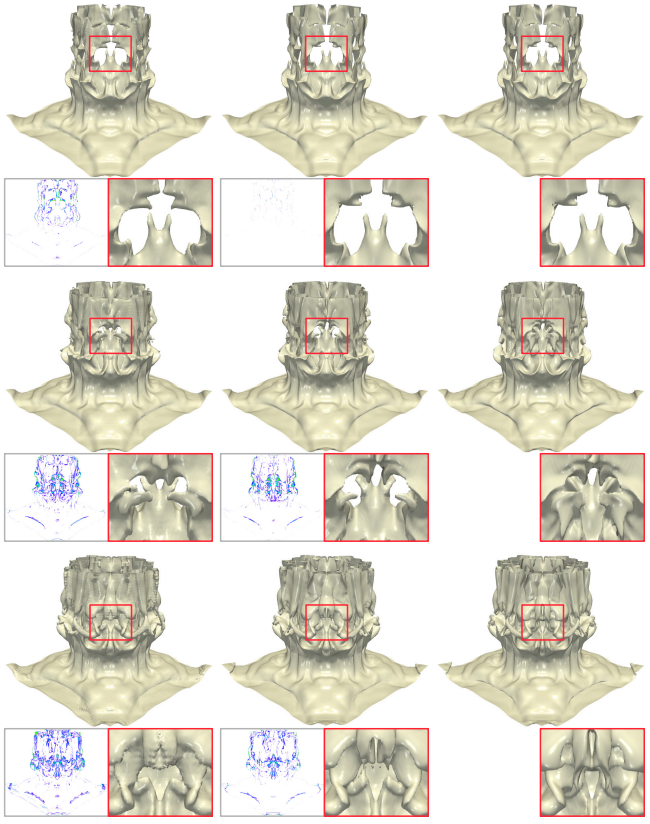
Table 10: PSNR (dB), LPIPS (for volume rendering), and CD (for isosurface rendering) values at time step 1 for GMT's many-to-many variable translation using the Tangaroa data set. The *rows* are the *source* variables, and the *columns* are the *target* variables.

PSNR ↑	DIV	VLM	VTM	LPIPS ↓	DIV	VLM	VTM	CD ↓	DIV	VLM	VTM
DIV	—	34.72	42.39	DIV	—	0.152	0.121	DIV	—	2.17	1.32
VLM	47.75	—	43.62	VLM	0.168	—	0.103	VLM	1.81	—	1.12
VTM	47.58	34.73	—	VTM	0.186	0.158	—	VTM	1.89	2.16	—



(a) StarGAN (b) GMT (c) GT

Fig. 10: Volume rendering comparison of one-to-many variable translation using the ionization data set at time step 100. Top to bottom: H→H+, H→He, and H→He+.



(a) StarGAN (b) GMT (c) GT

Fig. 11: Isosurface rendering comparison of one-to-many variable translation using the ionization data set at time step 100. Top to bottom: H→H+, H→He, and H→He+. $v = 0.5$, 0.7, and 0.1 for H+, He, and He+, respectively.

CD is between VTM→VLM and VLM→VTM.

Comparison with compression methods. SZ3 [40] and TTHRESH [41] are two state-of-the-art error-bounded lossy compressors for scientific data. We designed two experiments to comprehensively compare GMT and these compression methods. In the first experiment, we compress the volume data using SZ3 and TTHRESH to match the same average PSNR as the volumes generated by GMT. In the second experiment, we compress the volume data using SZ3 and TTHRESH to match the same compression rate (CR) as GMT. That is, the total size of compressed volume data of the target variables should be as close as possible to the size of the GMT model, which is 92.9 MB. We present the results of both experiments in Table 11. Note that TTHRESH cannot successfully compress and decompress 15 time steps of H+ of the ionization data set, 111 time steps of TA of the mantle data set, and all 100 time steps of DIV of the Tangaroa data set. We test on machines with different platforms, and the same issue persists. Hence, we mark the metrics with * to denote the average results of successfully compressed and decompressed time steps.

Additionally, to demonstrate the performance of GMT compared to SZ3 and TTHRESH when dealing with larger data sets, we double the size in each spatial dimension for the combustion data set via trilinear interpolation, resulting in volumes eight times the size of the original ones. The results shown in Table 11 indicate that, as the resolution of the data set increases, both SZ3 and TTHRESH produce larger compressed

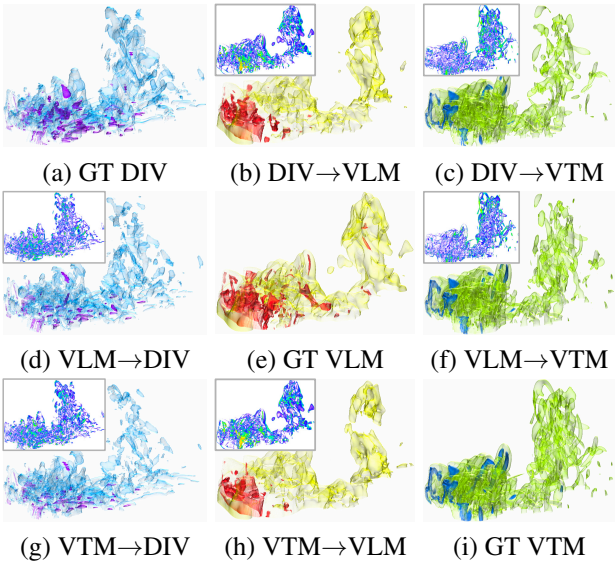


Fig. 12: Volume rendering for GMT's many-to-many variable translation using the Tangaroa data set at time step 1.

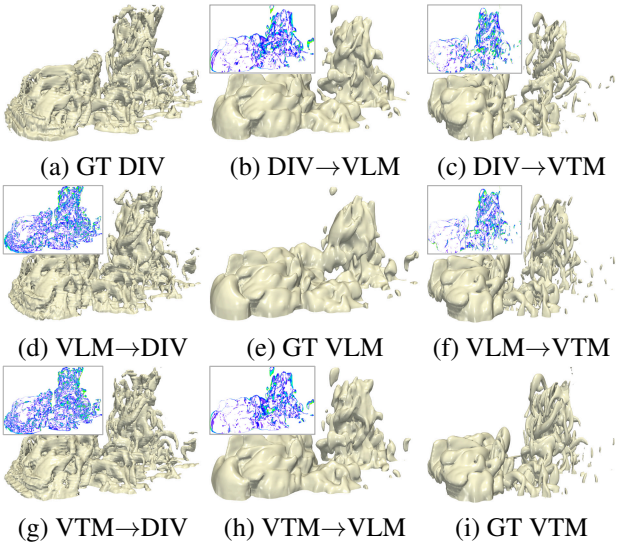


Fig. 13: Isosurface rendering for GMT's many-to-many variable translation using the Tangaroa data set at time step 1. $v = 0.58$, 0.5, and 0.1 for DIV, VLM, and VTM, respectively.

Table 11: Average CR, PSNR (dB), LPIPS (for volume rendering), and CD (for isosurface rendering) values across all time steps. The data set column also reports the source and target variables. For CD, $v = 0.02$, 0.5, 0.55, and 0.58, respectively. Metrics of partially failed cases are labeled with *, and those of completely failed cases are denoted by —.

		same PSNR			same CR		
data set	method	CR ↑	LPIPS ↓	CD ↓	PSNR ↑	LPIPS ↓	CD ↓
combustion (MF→CHI)	SZ3	**1332**	0.338	4.80	**47.29**	0.179	1.40
	TTHRESH	831	0.325	2.67	46.92	0.228	1.42
	GMT	511	**0.159**	**1.28**	44.20	**0.159**	**1.28**
combustion 2× (MF→CHI)	SZ3	3677	0.189	1.57	45.41	0.203	1.77
	TTHRESH	3847	0.239	1.59	45.65	0.249	1.68
	GMT	**4087**	**0.146**	**1.27**	**46.15**	**0.146**	**1.27**
ionization (H→H+)	SZ3	183	0.003	**0.02**	45.83	0.048	0.11
	TTHRESH	616*	0.003*	0.03*	46.11*	0.013*	0.09*
	GMT	**455**	**0.001**	0.04	**61.99**	**0.001**	**0.04**
mantle (CA→TA)	SZ3	**963**	0.333	**3.39**	**47.79**	**0.158**	**0.75**
	TTHRESH	3091*	0.381*	3.19*	47.31*	0.112*	0.51*
	GMT	241	**0.229**	3.57	36.37	0.229	3.57
Tangaroa (VLM→DIV)	SZ3	**607**	**0.122**	3.38	**67.62**	**0.013**	**0.28**
	TTHRESH	—	—	—	—	—	—
	GMT	80	0.134	**1.36**	50.35	0.134	1.36

files to maintain the same PSNR. Moreover, GMT performs the best with upscaled volumes because the network processes finer blocks. Since the model size of GMT stays the same with the upscaled data, neither SZ3 nor TTHRESH can beat GMT in both experiments. Based on the metrics, we conclude that GMT is more likely to outperform SZ3 and TTHRESH by achieving either a better generation quality or a higher CR when dealing with larger data (the upscaled combustion data set in our experiments) which hints that GMT is more suitable for compressing large-scale data sets.

We point out that GMT needs to perform an extrapolation task, as it does not have information about 60% of the time steps of the target variables, while SZ3 and TTHRESH see all time steps. That could also explain why GMT may perform worse than SZ3 in some cases. Compared to SZ3 and TTHRESH, we also observe much less noise and visual artifacts on the rendering images generated by GMT, as shown in Figure 14. Unlike volume data compression methods such as SZ3 and TTHRESH, GMT does not guarantee error bounds since it is a machine-learning approach. Despite the risks of using an approach without guaranteed error bounds, we believe GMT can be useful and reliable in exploring complex MTVD, identifying relationships of variables, and assisting scientists in developing hypotheses. In the appendix, we visualize and discuss the errors introduced by the reconstructed volumes using different methods, with volume renderings showing voxel-wise errors and box plots showing error distributions.

4.4. Limitations

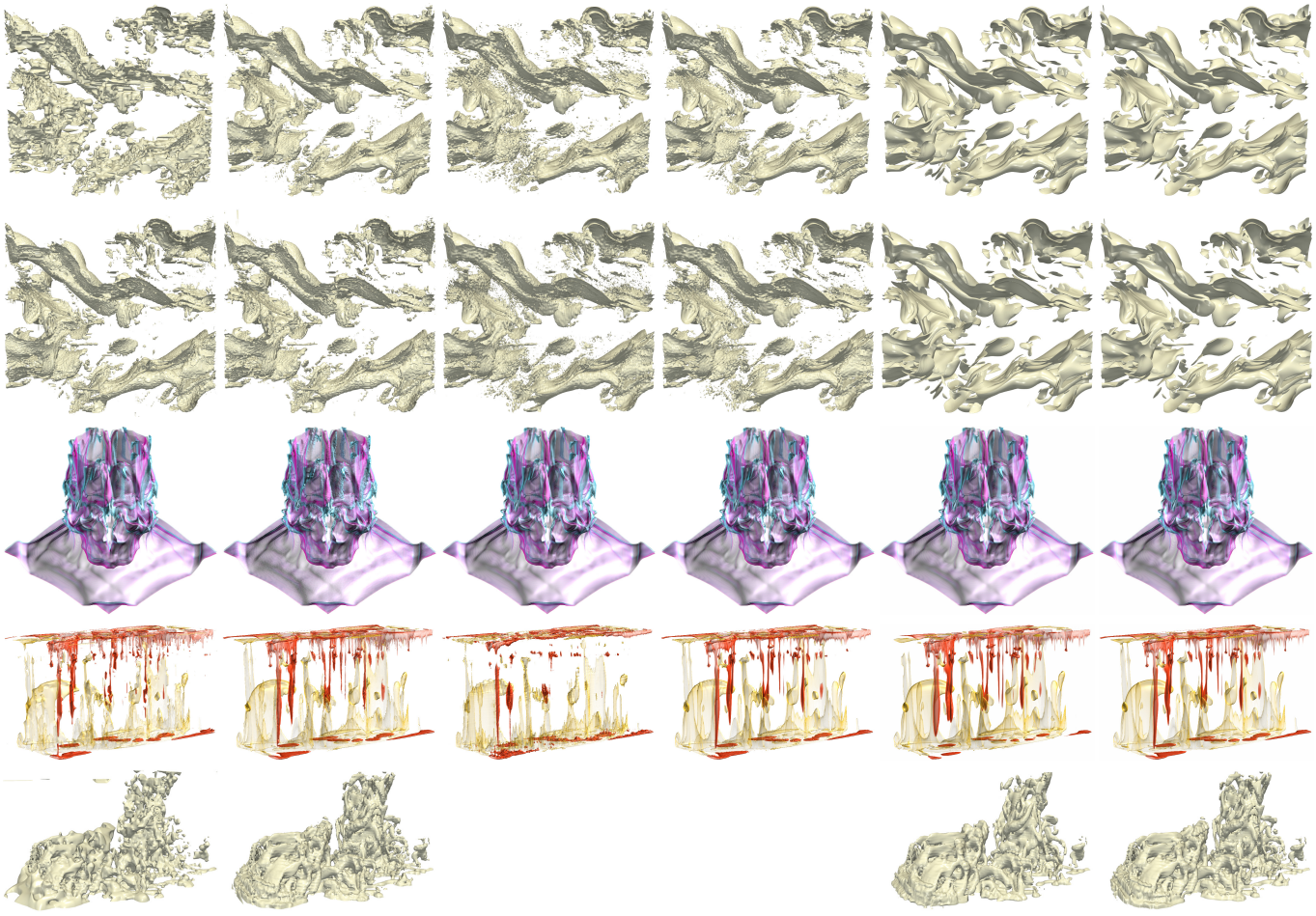
Compared with V2V, while GMT generalizes from one-to-one variable translation to one-to-many and many-to-many variable translation, it still has three limitations.

First, like V2V, GMT does not guarantee that any variable pair is transferrable. So, it relies on a variable selection step that chooses a subset of variables from the given MTVD for the subsequent variable translation task. Nevertheless, as demonstrated in the appendix, although not universal, handling multiple pairs in GMT could boost translation performance. Furthermore, using other variables not suggested by our variable selection step will likely lead to worse translation quality.

Second, given a pair of variables, the translation quality is not bidirectionally symmetric (refer to Table 10, and Figures 12 and 13). This is what we expect as the *transferable difficulty* defined by Han et al. [2] is not symmetric. Nevertheless, designing a more powerful solution to reduce the gap and improve quality in both directions could lead to a more balanced bidirectional translation.

Third, we observe that translating the same pair using a many-to-many network may perform worse than a one-to-many network. For example, a one-to-many network can achieve an average PSNR value of 50.35 dB for VLM→DIV (Table 7), whereas a many-to-many network only gets 47.75 dB (Table 10).

Despite these limitations, GMT significantly improves the existing V2V in generalizing from one-to-one to many-to-many variable translation. This critical step promotes solution adoption by domain scientists.



(a) SZ3-1 (b) SZ3-2 (c) TTHRESH-1 (d) TTHRESH-2 (e) GMT (f) GT

Fig. 14: Rendering comparison of different methods. (a) and (b) show SZ3 under the same PSNR and CR, respectively, as GMT. (c) and (d) show TTHRESH under the same PSNR and CR, respectively, as GMT. Top to bottom: isosurface rendering of combustion (MF→CHI, $v = 0.02$) at time step 87, isosurface rendering of $2\times$ upscaled combustion (MF→CHI, $v = 0.02$) at time step 87, volume rendering of ionization (H→H+) at time step 87, volume rendering of mantle (CA→TA) at time step 92, and isosurface rendering of Tangaroa (VLM→DIV, $v = 0.58$) at time step 1. TTHRESH leads to empty rendering for Tangaroa (VLM→DIV) because the compression fails.

5. Conclusions and Future Work

We have presented GMT, a generalized deep-learning approach for MTVD that supports one-to-many and many-to-many variable translation within a single framework. By allowing *simultaneously* training multiple variable pairs, GMT eliminates the need to train each variable pair *separately*, which is a tedious job given many variables. Furthermore, we optimize the GMT model according to our task requirement to reduce training costs and improve stability. To illustrate the improvement, we employ StarGAN as one of the baselines in multiple comparisons. The results suggest that GMT is more suitable than StarGAN for the MTVD translation task.

GMT is the first unified multi-domain translation focusing on scientific volumetric data. We believe this work represents a timely and significant step forward in deep learning for scientific visualization literature. As the research in this direction quickly grows, generalization is a challenge the community must tackle to tame the commonly produced large amount of multivariate or ensemble time-varying simulation data generated from scientific applications.

In the future, besides multivariate data sets, we will experiment with GMT on ensemble data sets to verify this method's efficacy. We will also address the uneven translation quality among variable pairs to maximize the benefit of training all variable pairs while minimizing their potential negative interinfluence. We point out that diffusion models, such as DALL·E 2 [42], stable diffusion [43], Imagen [44], and eDiff-I [45], have outperformed other deep generative models like GANs, variational autoencoders, and flow-based models in many image generation tasks. Since conditioning can be applied to the denoising operation to designate the output variable, it is possible to develop a solution using diffusion models for the generalized multivariate translation task. We offer two reasons for not using diffusion models in this work. First, Dhariwal et al. [46] and Rombach et al. [43] argued that since the denoising process usually takes multiple steps to achieve outstanding quality, diffusion models are slower than GANs in sampling speed. For large-scale volumetric data, we need to infer thousands or more blocks (e.g., 64^3) to fill the entire volume. Considering we also have multiple variables, each with a hundred or more time steps, it could be impractical to use diffusion models. Second,

according to Yang et al. [47], unlike GANs, the effectiveness of diffusion models for specific tasks is much less studied theoretically. Particularly, it is hard to find evidence that diffusion models outperform GANs when the objective is voxel-wise accuracy instead of likelihood. Therefore, we believe GAN is a more practical first step to achieving our goal. Nevertheless, we would like to explore the potential of diffusion models in addressing some of the issues we have found in the current implementation of GMT.

Acknowledgements

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

References

[1] Wang, C, Han, J. DL4SciVis: A state-of-the-art survey on deep learning for scientific visualization. IEEE Transactions on Visualization and Computer Graphics 2022;Accepted.

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

[3] Farokhmanesh, F, Höhlein, K, Westermann, R. Deep learning-based parameter transfer in meteorological data. Artificial Intelligence for the Earth Systems 2023;2.

[4] Chu, M, Thuerey, N, Seidel, HP, Theobalt, C, Zayer, R. Learning meaningful controls for fluids. ACM Transactions on Graphics 2021;40(4):100:1–100:13.

[5] Gu, P, Han, J, Chen, DZ, Wang, C. Scalar2Vec: Translating scalar fields to vector fields via deep learning. In: Proceedings of IEEE Pacific Visualization Symposium. 2022, p. 31–40.

[6] Kim, B, Azevedo, VC, Thuerey, N, Kim, T, Gross, MH, Solenthaler, B. Deep Fluids: A generative network for parameterized fluid simulations. Computer Graphics Forum 2019;38(2):59–70.

[7] Choi, Y, Choi, MJ, Kim, M, Ha, JW, Kim, S, Choo, J. StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2018, p. 8789–8797.

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

[9] van der Maaten, L, Hinton, G. Visualizing data using t-SNE. Journal of Machine Learning Research 2008;9(11):2579–2605.

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

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

[12] Han, J, Wang, C. SSR-TVD: Spatial super-resolution for time-varying data analysis and visualization. IEEE Transactions on Visualization and Computer Graphics 2022;28(6):2445–2456.

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

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

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

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

[17] Engel, D, Ropinski, T. Deep volumetric ambient occlusion. IEEE Transactions on Visualization and Computer Graphics 2020;27(2):1268–1278.

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

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

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

[21] Gu, P, Zheng, H, Zhang, Y, Wang, C, Chen, DZ. kCBAC-Net: Deeply supervised complete bipartite networks with asymmetric convolutions for medical image segmentation. In: Proceedings of International Conference on Medical Image Computing and Computer Assisted Interventions. 2021, p. 337–347.

[22] Shi, N, Xu, J, Wurster, SW, Guo, H, Woodring, J, Van Roekel, LP, et al. GNN-Surrogate: A hierarchical and adaptive graph neural network for parameter space exploration of unstructured-mesh ocean simulations. IEEE Transactions on Visualization and Computer Graphics 2022;28(6):2301–2313.

[23] Liu, MY, Tuzel, O. Coupled generative adversarial networks. In: Proceedings of Advances in Neural Information Processing Systems. 2016, p. 469–477.

[24] Liu, MY, Breuel, T, Kautz, J. Unsupervised image-to-image translation networks. In: Proceedings of Advances in Neural Information Processing Systems. 2017, p. 700–708.

[25] Huang, X, Liu, MY, Belongie, S, Kautz, J. Multimodal unsupervised image-to-image translation. In: Proceedings of European Conference on Computer Vision. 2018, p. 179–196.

[26] Brock, A, Donahue, J, Simonyan, K. Large scale GAN training for high fidelity natural image synthesis. In: Proceedings of International Conference on Learning Representation. 2019,.

[27] Hao, Z, Huang, X, Belongie, S. Controllable video generation with sparse trajectories. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2018, p. 7854–7863.

[28] Huang, X, Belongie, S. Arbitrary style transfer in real-time with adaptive instance normalization. In: Proceedings of IEEE International Conference on Computer Vision. 2017, p. 1501–1510.

[29] Wang, Z, Bovik, AC, Sheikh, HR, Simoncelli, EP. Image quality assessment: From error visibility to structural similarity. IEEE Transactions on Image Processing 2004;13(4):600–612.

[30] Maas, AL, Hannun, AY, Ng, AY. Rectifier nonlinearities improve neural network acoustic models. In: Proceedings of International Conference on Machine Learning. 2013,.

[31] Hawkes, ER, Sankaran, R, Sutherland, JC, Chen, JH. Scalar mixing in direct numerical simulations of temporally evolving plane jet flames with skeletal CO/H2 kinetics. Proceedings of the Combustion Institute 2007;31(1):1633–1640.

[32] Whalen, D, Norman, ML. Ionization front instabilities in primordial H II regions. The Astrophysical Journal 2008;673:664–675.

[33] Shahnas, MH, Yuen, DA, Pysklywec, RN. Mid-mantle heterogeneities and iron spin transition in the lower mantle: Implications for mid-mantle slab stagnation. Earth and Planetary Science Letters 2017;458:293–304.

[34] Popinet, S, Smith, M, Stevens, C. Experimental and numerical study of the turbulence characteristics of airflow around a research vessel. Journal of Atmospheric and Oceanic Technology 2004;21(10):1575–1589.

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

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

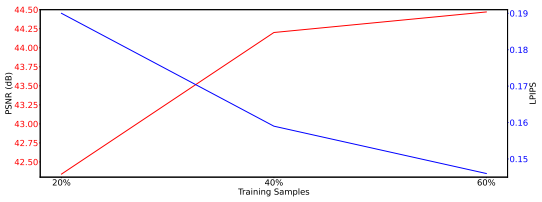
[37] Isola, P, Zhu, JY, Zhou, T, Efros, AA. Image-to-image translation with conditional adversarial networks. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2017, p. 1125–1134.

[38] Zhang, R, Isola, P, Efros, AA, Shechtman, E, Wang, O. The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2018,

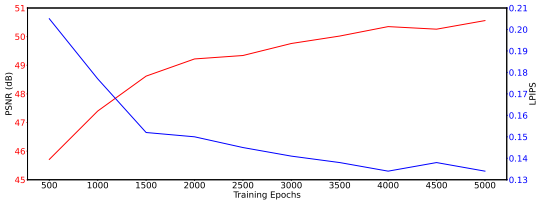
p. 586–595.
[39] Barrow, HG, Tenenbaum, JM, Bolles, RC, Wolf, HC. Parametric correspondence and chamfer matching: Two new techniques for image matching. In: Proceedings of International Joint Conference on Artificial Intelligence. 1977, p. 659–663.
[40] Liang, X, Zhao, K, Di, S, Li, S, Underwood, R, Gok, AM, et al. SZ3: A modular framework for composing prediction-based error-bounded lossy compressors. IEEE Transactions on Big Data 2022;Accepted.
[41] Ballester-Ripoll, R, Lindstrom, P, Pajarola, R. TTHRESH: Tensor compression for multidimensional visual data. IEEE Transactions on Visualization and Computer Graphics 2020;26(9):2891–2903.
[42] Ramesh, A, Dhariwal, P, Nichol, A, Chu, C, Chen, M. Hierarchical text-conditional image generation with clip latents. arXiv preprint arXiv:220406125 2022;.
[43] Rombach, R, Blattmann, A, Lorenz, D, Esser, P, Ommer, B. High-resolution image synthesis with latent diffusion models. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2022, p. 10684–10695.
[44] Saharia, C, Chan, W, Saxena, S, Li, L, Whang, J, Denton, E, et al. Photorealistic text-to-image diffusion models with deep language understanding. arXiv preprint arXiv:220511487 2022;.
[45] Balaji, Y, Nah, S, Huang, X, Vahdat, A, Song, J, Kreis, K, et al. eDiff-I: Text-to-image diffusion models with ensemble of expert denoisers. arXiv preprint arXiv:221101324 2022;.
[46] Dhariwal, P, Nichol, A. Diffusion models beat GANs on image synthesis. In: Ranzato, M, Beygelzimer, A, Dauphin, Y, Liang, P, Vaughan, JW, editors. Proceedings of Advances in Neural Information Processing Systems; vol. 34. 2021, p. 8780–8794.
[47] Yang, L, Zhang, Z, Song, Y, Hong, S, Xu, R, Zhao, Y, et al. Diffusion models: A comprehensive survey of methods and applications. arXiv preprint arXiv:220900796 2022;.

Appendix

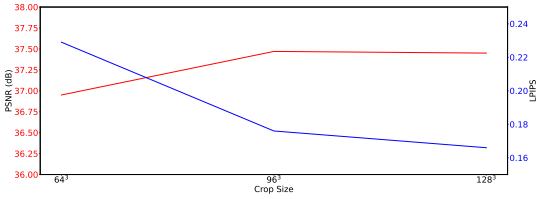
Besides loss combinations and injection strategies investigated in Sections 3.2 and 3.3 of the paper, we study three parameters influencing GMT training: training sample, training epoch, and crop size. In addition, we perform additional experiments to explore how variable selection results impact the subsequent quality of variable translation and investigate error bounds using different methods.



(a) combustion (MF→CHI)



(b) Tangaroa (VLM→DIV)

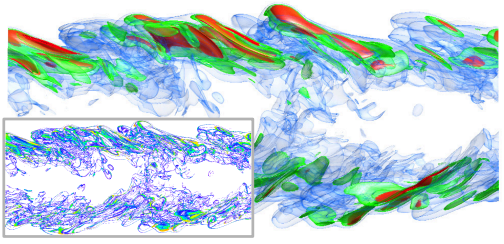


(c) mantle (CA→TA)

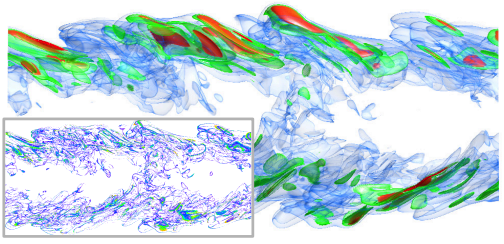
Fig. 1: Average PSNR and LPIPS for GMT under different (a) training samples, (b) training epochs. LPIPS is for volume rendering.

Training sample. We train GMT using the combustion data set under different percentages of the data as training samples (20%, 40%, 60%). Figure 1 (a) shows the PSNR and LPIPS values for MF→CHI translation as the percentage increases. As we can see, there are significant performance gains for PSNR and LPIPS when the training samples increase from 20% to 40%. After that, the gains are much narrower. Figure 2 compares volume rendering results under different percentages of the data as training samples. We can observe clear visual quality improvement when the percentage increases from 20% to 40%. However, when 60% of the data are used for training, visual quality improvement is less substantial. As the training cost increases almost linearly with the training samples, we take 40% of the data to train GMT for quality and speed tradeoffs.

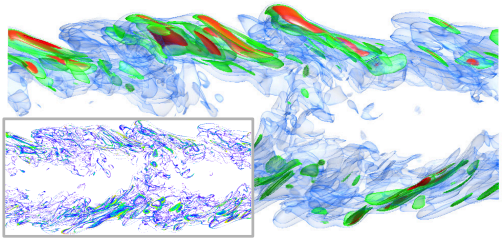
Training epoch. We train GMT using the ionization data set under different epochs (500 to 5000 with a step size of 500). Figure 1 (b) shows the PSNR and LPIPS values for VLM→DIV translation as the number of epochs increases. As we can see, the performance continues to increase until 4000 epochs. There is a drop from 4000 to 4500 epochs, and the improvement from 4000 to 5000 epochs is also much less significant. Figure 3 compares volume rendering results under different epochs. We can observe significant visual quality improvement from 500 to 1000 epochs, followed by 1000 to 1500 epochs. Afterward, details are still getting refined as the epoch increases. Since performance starts to drop after 4000, we use 4000 epochs for training GMT.



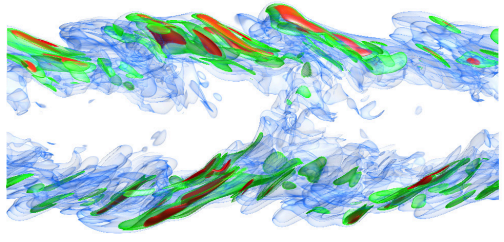
(a) 20%



(b) 40%



(c) 60%



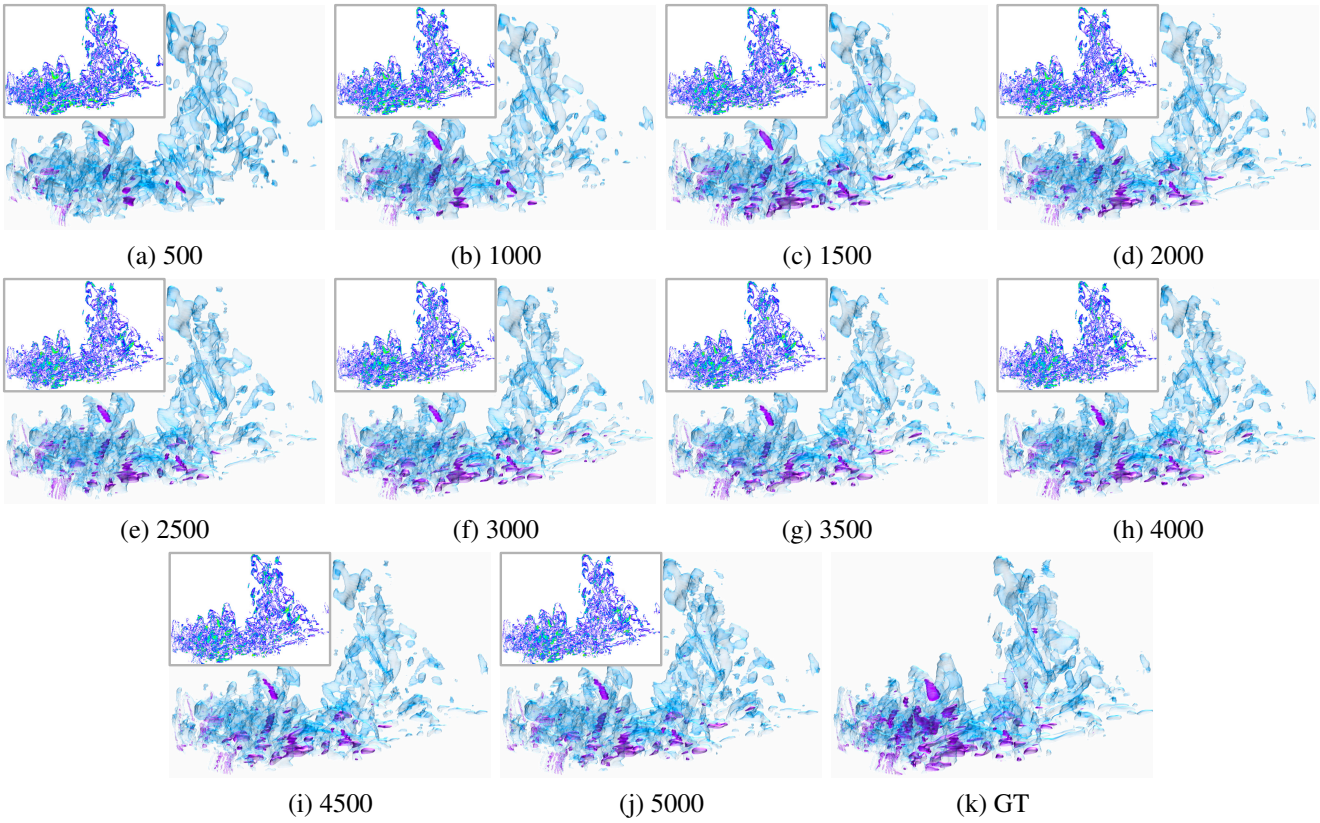
(d) GT

Fig. 2: Volume rendering comparison for GMT under different percentages of the data as training samples using the combustion (MF→CHI) data set at time step 6.

Table 1: GMT training time (H:M:S) of the mantle data set using different crop sizes on the one-to-three variable translation task.

crop size	64^3	96^3	128^3
training time	27:46:12	147:13:12	257:53:59

Crop size. We train GMT using the mantle data set under different crop sizes (64^3, 96^3, 128^3). A larger crop size leads to a more extensive receptive field, helping the network capture more information. However, as the crop size increases, the training time increases dramatically. The training time for different crop sizes is shown in Table 1. With the crop size of 128^3, we spent more than 257 hours training GMT for 4000 epochs, which is more than 9× the training time with the crop size of 64^3. The crop size of 96^3 also costs about 5× the train-

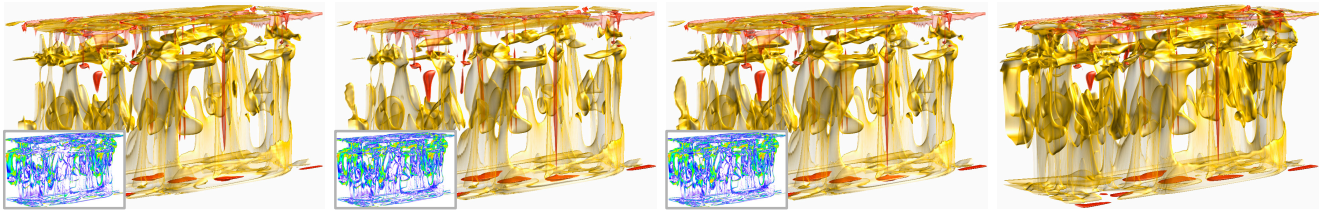


(a) 500 (b) 1000 (c) 1500 (d) 2000

(e) 2500 (f) 3000 (g) 3500 (h) 4000

(i) 4500 (j) 5000 (k) GT

Fig. 3: Volume rendering comparison for GMT under different training epochs using the Tangaroa (VLM→DIV) data set at time step 1.



(a) 64^3 (b) 96^3 (c) 128^3 (d) GT

Fig. 4: Volume rendering comparison for GMT under different crop sizes using the mantle (CA→TA) data set at time step 150.

ing time with the crop size of 64^3. Figure 1 (c) shows the PSNR and LPIPS values for CA→TA translation as the crop size increases. As we can see, as the crop size increases from 64^3 to 96^3, both PSNR and LPIPS values improve. When the crop size increases from 96^3 to 128^3, the PSNR value stays almost the same, but the LPIPS value decreases, which implies visual quality improvement. We can observe in Figure 4 that 96^3 and 128^3 crop sizes produce visually better results compared to 64^3 crop size. Considering the significant amount of extra time for training using a crop size of 96^3 or 128^3, we choose the crop size of 64^3 to train GMT for efficiency.

Variable selection. To illustrate the effectiveness of variable selection on variable translation, we compare the performance of GMT in three settings for the ionization and mantle data sets. First, we train three individual one-to-one models with one of the three variable pairs respectively (H→H+, H→He, and H→He+ for ionization) (CA→TA, CA→DA, and CA→EA for mantle). Second, we train a one-to-many model with selected variables (H→H+, He, and He+ for ionization) (CA→TA, DA, and EA for mantle). Last, we train a one-to-many model with all variables of the ionization dataset (H→H+, He, He+, T, H2, and PD for ionization) (CA→TA, DA, EA, T, and VLM for mantle).

From the results reported in Table 2, we can see that, for the ionization dataset, the translation quality of the 1-to-3 model is as good as separate 1-to-1 models. However, after we add three extra variables (T, H2, and PD) far from the existing variables in the t-SNE projection (see Figure 6 (a) in the paper), the translation quality of the existing pairs reduces considerably. The metrics show that the translation quality of both T and PD is worse than other variables. Although the PSNR of H2 is not as bad as T and PD because most of the voxels have relatively small values (below 0.01 after normalization), both LPIPS and CD of H2 are still much worse than H+, He, and He+. The translation performance of the ionization data set is consistent with the proximity pattern demonstrated in the t-SNE projection (i.e., variables with closer footprints in the projection are easier in translation). Hence, the variable selection step is effective by

Table 2: PSNR (dB), LPIPS (for volume rendering), and CD (for isosurface rendering) values averaged across all time steps for different variable selection settings using the ionization and mantle data sets. For CD of the ionization data set, $v = 0.5$, 0.7, 0.1, 0.05, 0.01, and 0.4 for H+, He, He+, T, H2, and PD, respectively. For CD of the mantle data set, $v = 0.55$, 0.6, 0.75, 0.6, and 0.15 for TA, DA, EA, T, and VLM, respectively.

	average across all time steps								
	PSNR ↑			LPIPS ↓			CD ↓		
ionization	1-to-1	1-to-3	1-to-6	1-to-1	1-to-3	1-to-6	1-to-1	1-to-3	1-to-6
H→H+	**63.83**	61.99	56.05	**0.001**	**0.001**	0.004	**0.03**	0.04	0.08
H→He	**46.71**	46.43	45.01	**0.039**	**0.039**	0.049	**0.30**	**0.30**	0.35
H→He+	46.12	**47.08**	44.91	0.040	**0.036**	0.047	0.56	**0.51**	0.63
H→T	—	—	30.30	—	—	0.175	—	—	6.22
H→H2	—	—	44.77	—	—	0.133	—	—	1.75
H→PD	—	—	34.96	—	—	0.222	—	—	3.29
	PSNR ↑			LPIPS ↓			CD ↓		
mantle	1-to-1	1-to-3	1-to-5	1-to-1	1-to-3	1-to-5	1-to-1	1-to-3	1-to-5
CA→TA	32.94	**36.37**	34.62	0.257	**0.229**	0.237	6.83	**3.57**	6.80
CA→DA	30.72	31.06	**31.24**	0.315	0.318	**0.291**	16.13	**13.11**	13.35
CA→EA	31.28	31.25	**31.33**	0.488	0.530	**0.480**	29.19	**27.75**	28.17
CA→T	—	—	33.59	—	—	0.198	—	—	5.15
CA→VLM	—	—	27.78	—	—	0.508	—	—	17.22

eliminating variables far away to maintain good performance of GMT.

For the mantle data set, the situation is a bit different. Although their metrics are pretty close, 1-to-1 models perform the worst compared to the 1-to-3 model and 1-to-6 model. We believe that one-to-many models may create a better latent space during pre-training, which allows the network to generate more meaningful results. With T and VLM included in the training, the translation quality of existing variable pairs is hardly affected. This is because two adjacent time steps in the mantle data set are 500 million years apart, meaning the data set has significantly less temporal coherence than others. Although the selected variables (CA, TA, DA, and EA) have better correlations according to the t-SNE projection, predicting the corresponding time step between selected variables is still extremely difficult for GMT. Therefore, the variable selection step might be less effective for a data set with less temporal coherence.

Error bounds. GMT is a deep-learning method that does not guarantee error bounds for its generated volumes. In scientific data analysis, it is necessary to know the bounds and distributions of the errors introduced by the synthesized data. In Figure 5, we compare volume rendering of voxel-wise errors derived from generated volumes using SZ3, TTHRESH, and GMT. To illustrate error distributions, we present box plots in Figure 6 for each corresponding volume shown in Figure 5. Considering both error visualizations and box plots, we point out two main observations. First, for all cases except ionization, the errors associated with the generated results of GMT are much less varied than those of SZ3-1 and TTHRESH-1 (i.e., SZ3 and TTHRESH under the same PSNR). With a higher variance of errors, the volumes generated by SZ3 and TTHRESH exhibit more artifacts and noises. With closer bounds and fewer outliers under the same PSNR, GMT is more trustworthy for data reduction. Second, the errors in volumes produced by GMT are mostly concentrated on certain data regions. Values in these regions are sensitive to time and variable changes. With highlighted anomalies, visualizing errors and their statistic patterns may also provide insights into the drives of particular trends over time and relationships between variables captured by GMT.

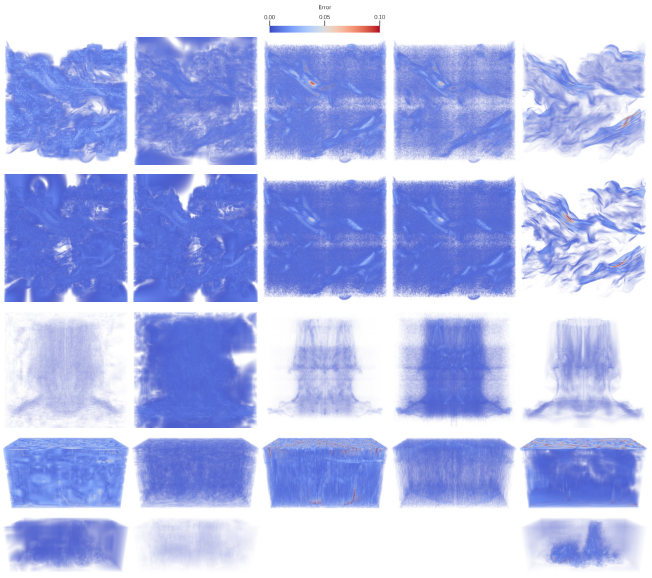


Fig. 5: Volume rendering comparison of voxel-wise errors between the synthesized volumes generated by different methods and the GTs. (a) to (e): SZ3-1, SZ3-2, TTHRESH-1, TTHRESH-2, and GMT. (a) and (b) show SZ3 under the same PSNR and CR, respectively, as GMT. (c) and (d) show TTHRESH under the same PSNR and CR, respectively, as GMT. Top to bottom: combustion (MF→CHI) at time step 87, 2× upscaled combustion (MF→CHI) at time step 87, ionization (H→H+) at time step 87, mantle (CA→TA) at time step 92, and Tangaroa (VLM→DIV) at time step 1. TTHRESH leads to empty rendering for Tangaroa (VLM→DIV) because the compression fails.

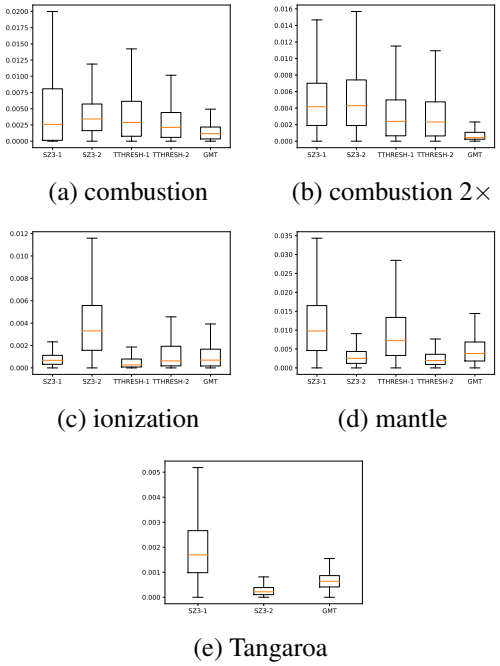


Fig. 6: Box plots of voxel-wise errors between the synthesized volumes generated by different methods and the GTs. SZ3-1 and SZ3-2 show SZ3 under the same PSNR and CR, respectively, as GMT. TTHRESH-1 and TTHRESH-2 show TTHRESH under the same PSNR and CR, respectively, as GMT. (a) to (e): combustion (MF→CHI) at time step 87, 2× upscaled combustion (MF→CHI) at time step 87, ionization (H→H+) at time step 87, mantle (CA→TA) at time step 92, and Tangaroa (VLM→DIV) at time step 1. TTHRESH does not have box plots for Tangaroa (VLM→DIV) because the compression fails.