JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 40, 397-419 (2024)
DOI: 10.6688/JISE.202403_40(2).0012

Distribution-Based Time-Varying Ensemble Scientific Data Reduction for Uncertainty Visualization and Analysis

ZHI-RONG LIN[1], AN-LUNG HSIAO[1], GUAN LI[2] AND KO-CHIH WANG[1,+]
[1]*Department of Computer Science and Information Engineering*
National Taiwan Normal University
Taipei, 106 Taiwan
[2]*Computer Network Information Center*
Chinese Academy of Sciences
Beijing, 100083 P.R. China
E-mail: {60947081s; 61147063s}@ntnu.edu.tw; liguan@sccas.cn; kcwang@ntnu.edu.tw[+]

Scientists often study physical phenomena using computer simulation models. The same simulation can generate different datasets because of different input parameter configurations or internal random variables. Therefore, each grid point is represented by multiple data values from simulation runs, and we call this type of data ensemble dataset. To gain insight into the physical phenomenon, scientists often have to visualize and analyze the ensemble datasets with uncertainty. Distribution-based data representation is a popular approach to handling the ensemble dataset and supporting uncertainty visualization. However, storing a time-varying ensemble dataset needs hundreds or even thousands of times storage size. Given the size of the time-varying ensemble dataset, it is natural to develop storage-reduced data representation to facilitate the time-varying ensemble data exploration. We propose a novel data representation to compactly represent the time-varying scientific data for uncertainty visualization and analysis. Our approach decouples data on the temporal domain into two types of distributions and stores. One distribution summarizes the data values on the temporal domain, and the other distribution describes the occurrence probability of a data value on the temporal domain. Our approach can provide time-varying ensemble scientific data analysis with uncertainty quantification and detailed temporal feature evolution with less storage requirement.

Keywords: data reduction, ensemble data, large-scale data, probability distribution, statistical modeling

1. INTRODUCTION

Scientists often study physical phenomena using computer simulation models, which are usually affected by initial configurations and internal random variables. Therefore, a single simulation run is insufficient for scientists to study the inherent uncertainty and deeply understand the physical phenomenon. Scientists usually have to change the initial configurations to run the simulation and generate data for the same experiment. The collection of the data from simulation runs is called *ensemble* dataset. In an ensemble dataset, each data point is represented by multiple data values from simulation runs. Therefore, the dataset size is much larger than a regular scientific dataset. To handle the large ensemble dataset and efficiently analyze and visualize the data with the uncertainty inherent in the simulation, a common practice is to represent each data point by a probability density function and store the function by a compact distribution representation, such as a histogram

Received December 27, 2022; revised February 15, 2023; accepted June 15, 2023.
Communicated by Mu-Yen Chen.
[+] Corresponding author.

or a Gaussian Mixture Model (GMM). The distribution representation can summarize data values at a data point and reduce the storage requirement to facilitate the data analysis pipeline. Many uncertainty analysis and visualization works [1-4] have been done based on the distribution data representations.

With the advance of supercomputers, simulations can produce time-varying ensemble datasets with hundreds or even thousands of time steps nowadays. Although the distribution representation can compactly represent the ensemble dataset for data analysis, storing a time-varying ensemble dataset needs hundreds or even thousands of times the storage size. Given the size of the time-varying ensemble dataset, it is natural to develop storage-reduced data representation to facilitate the time-varying ensemble data exploration. One common practice to handle time-varying scientific data is to reduce the sampling rate in the temporal domain [5-7]. However, scientists may lose track of the feature evolution between two sampled time steps. Lossy compression [8-10] is another choice to handle the large-scale dataset, but it may introduce unacceptable data quality when high compression is required.

This work proposes a compact distribution-based representation to facilitate the uncertainty analysis and visualization of time-varying ensemble data. The main idea is to transform the probability density functions into a few distributions with a better tradeoff between storage cost and representation accuracy. At a spatial grid point, we transform adjacent probability density functions in the temporal domain into one numerical distribution and multiple temporal distributions. The numerical distribution summarizes the probability density functions at the grid point, and each temporal distribution describes the occurrence probability of a data value interval over the temporal domain. Because data samples with similar values usually concentrate at neighboring time steps instead of randomly spread over the temporal domain, the temporal distribution can be well represented by a Gaussian Mixture Model (GMM), a parametric model that costs less storage space. In addition, we also utilize spatial coherence to reduce our representation's storage cost further. The temporal distributions in the neighboring spatial region are often similar. We use the SLIC-based algorithm [11] to efficiently identify similar temporal distributions that can be represented by one represented temporal distribution without scarifying too much quality. When analyzing and visualizing the dataset, the probability density function at any given grid point and time step can be reconstructed by Bayes' rule. We show that our data representation can represent the time-varying ensemble dataset with a better tradeoff between the storage cost and representation quality than alternatives.

This paper is organized as follows. We review and discuss related works in Section 2. In Section 3, we provide an overview of the background and our approach. Section 4 and Section 5 introduce our data representation and data reconstruction process in detail, respectively. Sections 6 and 7 quantitatively and qualitatively compare our approach with alternatives. In Section 8, we report the timing of our approach and discuss the influence of hyperparameters in our approach. Section 9 concludes this paper and discusses future works.

2. RELATED WORKS

Large-scale Data Representation: Due to the storage limitation, raw data of largescale simulations are challenging to preserve. Sub-sampling is a popular method to reduce the data size by selecting a subset from raw data. In addition to the simple sub-sampling method, importance sampling is also another popular method, such as saving critical time steps or some locations with drastic changes [12-14]. However, the inevitable information loss will damage the data analysis quality. Another common way to get a high data compression ratio is to use lossy compression techniques. Di *et al.* [8] propose an error-bounded data compression method that allows the user to specify an absolute or relative error tolerance. A high compression rate from the lossy compression approach usually results in an unacceptable data loss. O'Leary *et al.* [15] proposed an image-based system that directly renders images of data in situ to greatly reduce the disk I/O time to facilitate large-scale data visualization. He *et al.* [16] proposed a GAN-based framework to synthesize the isosurface images or volume rendering images of any given rendering, view, and simulation parameters. Chen *et al.* [17] can improve the pathline visualization from the downsampling data by integrating the error quantification into the pathline computation pipeline. Li *et al.* [18] surveyed compact data representation techniques for scientific datasets and categorized the techniques by use cases.

Distribution-based Data Representation: In visualization applications, distribution-based data representations can not only reduce the data size but also provide the potential for uncertainty estimation. Therefore, statistic-based presentation for scientific data has been the favored approach in the past few years. Thompson *et al.* [2] use a histogram to encode the distribution of values in each local block. Lee *et al.* [19] utilize integral histograms, an extension of the summed-area tables, and discrete wavelet transform. Wang *et al.* [20] divide the space into blocks and use the spatial Gaussian mixture model to represent the data of a block. Wang *et al.* [21] use the probability distribution model to cut the data on a camera ray into several segments, and each part uses a probability distribution model to represent the data and then reconstruct the information for data visualization. Li *et al.* [22] proposed an adaptive space subdivision approach to represent dark matter particles within each subregion dataset by GMM. Hazarika *et al.* [23] utilized the copula-based function to compactly represent the multi-variant scientific dataset for posthoc data analysis and visualization. Yang *et al.* used [24] data-parallel primitives to develop the efficient distribution-based scientific data modeling algorithms based on VTK-*m* [25] library to facilitate the large-scale scientific data processing pipeline. Dutta *et al.* [26] computed the statistical values of subblocks in situ to compactly represent the jet engine simulation data for engine stall detection. Those approaches can achieve a reasonable storage space reduction rate but are not focused on ensemble data sets.

Visualization and Analysis Techniques for Ensemble Data: This paragraph discusses visualization and analysis techniques associated with ensemble data. Thompson *et al.* [2] use histograms to summarize values of the data points and visualize the uncertainty in the ensemble dataset. Hazarika *et al.* [27] use the Gaussian copula to preserve the correlation between each grid. Their approach can use different distribution representations to flexibly represent each grid's distribution and enable uncertainty visualization. Liu *et al.* [1] use a Gaussian Mixture Model to represent data values on a voxel to produce a compact ensemble data representation. They repeatedly resample values from the GMMs of grid points to render consecutive images to show the uncertainty of the data by the color variable on a pixel. Wang *et al.* [28] use parallel coordinates plots to help the user explore and analyze ensemble data in different resolutions. Sanya *et al.* [29] proposed a software tool called

Noodles to enhance the spaghetti plot for weather ensemble dataset understanding. Sun *et al.* [30] proposed a deep-learning-based visual analysis system. The system not only uses an artificial neural network to learn the behavior of a cosmology simulation but also builds a visual system based on the knowledge learned from the machine learning model to guide users in identifying critical ensemble members. Ovis was created by Höllt *et al.* [31] for visualization of ensemble simulations of sea surface heights for ocean forecasting. Wang *et al.* [32] collected ensemble visualization work from the past decade and categorized ensemble visualization techniques. However, the above works focus on the efficient presentation of ensemble datasets instead of dealing with the large-scale data problem of the time-varying ensemble dataset.

3. OVERVIEW

An ensemble dataset is the collection of the simulation outputs from multiple simulation runs, and each simulation run has slightly different simulation initial conditions or random parameter settings. Each output from a simulation run is called an ensemble member. Fig. 1 shows a time-varying ensemble dataset with *M* ensemble members. To visualize and analyze the uncertainty inherent in the computer simulation, modeling the data values at the same grid point and time step as a probability density function (PDF) [1, 2] is a popular choice because this approach can preserve the statistical characteristic and reduce the memory footprint. Fig. 1 also illustrates this representation. Different distribution representations can be chosen to represent the PDF according to its complexity. If the complexity is low, a single Gaussian could be sufficient. Gaussian Mixture Model (GMM) could be a choice if the PDF is a multi-model distribution. Histogram is usually used to represent the most complex PDF.

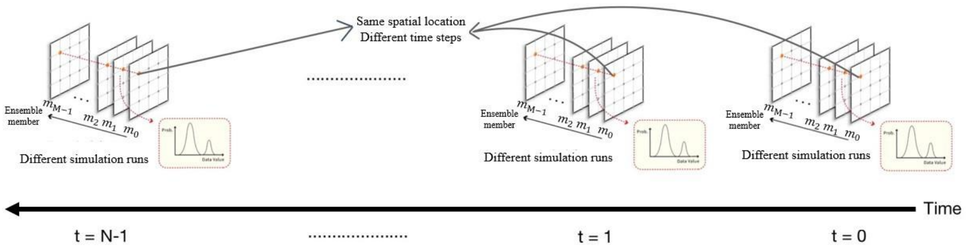


Fig. 1. Illustration of time-varying an ensemble dataset. In this example, the dataset has *N* time steps and consists of *M* ensemble members.

The main purpose of our representation is to compactly represent the probability density functions (PDFs) from the ensemble datasets to support the uncertainty visualization and analysis for the time-varying ensemble dataset. Fig. 2 shows the overview of our method. Our method consists of two main stages, data modeling, and PDF estimation. Data modeling transforms a time-varying ensemble dataset to our compact data representation that consists of numerical and temporal distributions. The PDF estimation stage calculates the PDFs from our data representation at all grid points and an arbitrary time step for uncertainty visualization and analysis.

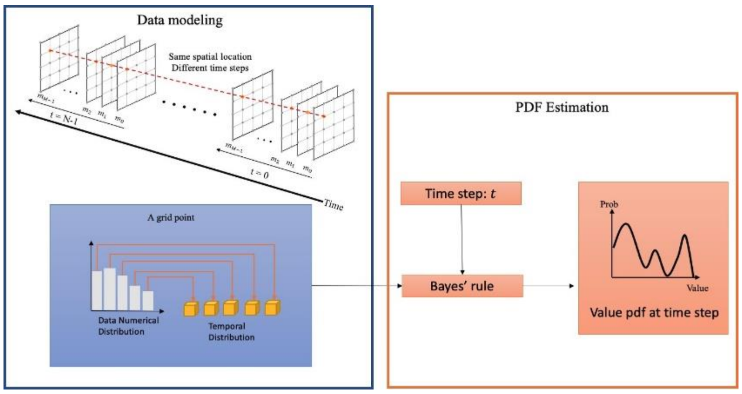


Fig. 2. Overview of our approach. In data modeling (the left part), we collect all data values at the same grid point over time to calculate a numerical distribution and multiple temporal distributions for the grid point. When visualizing and analyzing the uncertainty of the dataset, the PDF at a time step can be calculated by combining the distributions using Bayes' rule.

4. DATA MODELING

In this section, we will introduce the data modeling process. Our data representation aims to decompose the PDFs at the same grid point over the temporal domain into one numerical distribution and multiple temporal distributions. After the decomposition, the data can be stored by a more compact distribution representation. We collect all the samples for each grid point over the temporal domain to compute a numerical distribution represented by a histogram. We also compute a temporal GMM for each bin of the numerical distribution. A temporal GMM stores the occurrence probability over the temporal domain for the value interval of the corresponding bin. Fig. 3 shows the overview of the data modeling process.

4.1 Numerical Distribution

For each grid point and a temporal interval, we use a histogram to summarize all data values from all ensemble members, as shown in Fig. 3. A histogram has B bins, and the user specifies the number of bins. The histogram bins equally subdivide the data value range into B intervals. If B is the total number of samples in bin_i and Q is the total number of samples at the grid point, B/Q is the probability of bin_i in the numerical distribution. Although the histogram is compact and summarizes all data values from time steps, the histogram lacks temporal information of the data samples. When visualizing and analyzing the uncertainty, the accurate PDF at arbitrary time steps is not available. Therefore, we will also compute the temporal distribution for each bin that stores the occurrence probability of the data samples within the value interval of the bin.

4.2 Temporal Distribution

The temporal distribution represents the occurrence probability of data samples within the value interval of a bin in the numerical distribution. Therefore, any bin with non-zero probability must have a temporal distribution. In our work, we use a Gaussian Mixture Model (GMM) to represent a temporal distribution, and we call it temporal GMM.

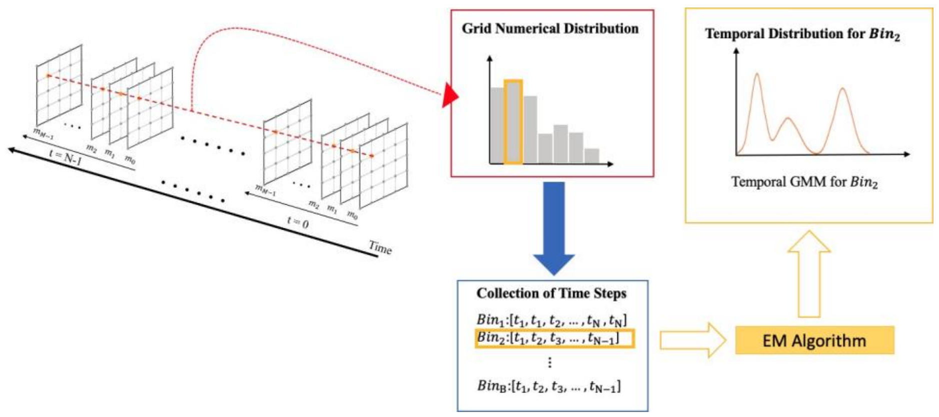

Grid Numerical Distribution
Temporal Distribution for Bin_2
Temporal GMM for Bin_2
Collection of Time Steps
Bin_1:[$t_1, t_1, t_2, \ldots, t_N, t_N$]
Bin_2:[$t_1, t_2, t_3, \ldots, t_{N-1}$]
Bin_B:[$t_1, t_2, t_3, \ldots, t_{N-1}$]
EM Algorithm
t = N-1
t = 0
Time

Fig. 3. This figure illustrates our data representation for the data values at a grid point and in a temporal interval; The red line on the left-hand side shows the data at a grid point and in a temporal interval. The top middle figure shows the numerical distribution of the data on the red line; The red box at the bottom middle figure shows the time stamps of data samples on the red line whose data values are within the value interval of Bin_1; These time stamps are used to compute the temporal distribution for the bin Bin_1 in the numerical distribution.

The Gaussian mixture model is widely used because it can adapt to the different complexity of the distribution and well represent the distributions by adjusting the number of Gaussian components. In addition, each Gaussian component is a parameter distribution representation, and it is compact because each Gaussian component only stores a weight, mean, and standard deviation. Eq. (1) shows the temporal GMM of a bin.

$$TGmm_{b_i}(t) = \sum_{j=1}^{k} w_j * \mathcal{N}(t \mid u_j, \sigma_j) \tag{1}$$

$TGmm_{b_i}(t)$ represents the temporal Gaussian Mixture Model of histogram bin b_i. The domain of the temporal GMM is the time steps. $TGmm_{b_i}(t)$ is the occurrence probability of data samples within the data interval of bin b_i at time step t. K is the number of Gaussian components, w_j, u_j and σ_j are the weight, mean, and standard deviation in the Gaussian Mixture Model, $\mathcal{N}(t \mid u_j, \sigma_j)$ is the probability density at time step t of the Gaussian distribution.

To compute a temporal GMM for a bin of the numerical distribution, we first collect all time stamps of the data samples within the value interval of the bin. To model a GMM, Expectation-Maximization (EM) is a popular algorithm. The EM algorithm takes the time stamps as the input and estimates the weights, means, and standard deviations of Gaussian components of the temporal GMM. The middle figure of Fig. 3 is an example. The data samples' time stamps of numerical distribution's Bin_1 are collected, and the EM algorithm computes the corresponding temporal GMM.

4.3 Temporal Distribution Sharing

To further reduce the storage costs, we propose a method to identify similar temporal distributions and only store one distribution to represent all of them. This task is a clustering problem to identify similar temporal distributions and use one distribution in a cluster to represent the cluster. However, clustering temporal distributions from all bins and all grid points is time-consuming and impractical because the number of temporal distributions is enormous.

In order to efficiently identify temporal distribution clusters, we utilize spatial coherence in the scientific dataset. In the scientific dataset, the data does not often change dramatically in a local region. Two temporal distributions in a local region have a higher chance of being similar enough to be represented by one temporal distribution without damaging the representation quality too much. Therefore, we use the SLIC-based [11] algorithm to identify the temporal distribution clusters. SLIC is a limited region K-means clustering algorithm that considers the spatial proximity to search similar samples. The limited region search can utilize the scientific data property to significantly reduce the computation for finding similar temporal distribution.

To use the SLIC-based algorithm, we have to define the "position" of a temporal distribution for the SLIC-based algorithm to estimate the spatial proximity between two temporal distributions. In our scenario, if the temporal distributions with both similar spatial locations and corresponding bin IDs, they will have a higher chance of being similar. Therefore, the "position" of a temporal distribution for the SLIC-based algorithm is defined in an s+1 dimensional space where s is the dimensionality of the dataset's spatial space and the extra one is the bin ID. We call the s+1 dimensional space "Grid-Bin" space. For example, if the dataset is a 3D dataset, the position of a temporal distribution for the SLIC-based algorithm is a point in the Grid-Bin space, and the point should be a 4D point, (x, y, z, b), where b is the bin ID. The steps of our SLIC-based algorithm are outlined below.

Step 1: Initialization: For a dataset, we have $G \times B$ temporal distributions for a dataset, where G is the number of the grid point of the dataset and B is the bin count of a histogram. In the initialization step, we will subdivide the Grid-Bin space of the dataset into sub-blocks. First, we create P partitions by regularly dividing the spatial domain. We also subdivide the bin dimension into l blocks. Therefore, we can create a $P \times l$ nonoverlapping Grid-Bin sub-blocks and temporal distributions in a sub-block belonging to the same initial cluster. Therefore, we initially have $P \times l$ clusters, and each cluster has G/P C temporal distribution, where $C = B/l$. We set the temporal distribution at the center of a Grid-Bin sub-block to be the center temporal distribution of the cluster. Fig. 4 shows the initialization step.

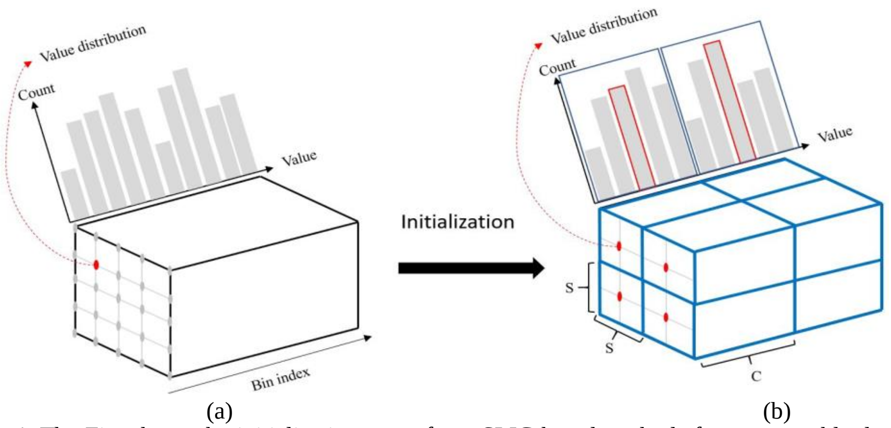


Fig. 4. The Fig. shows the initialization step of our SLIC-based methods for an ensemble dataset with 2D spatial space and 1 histogram bin space; (a) illustrates the Grid-Bin space; (b) The 2D spatial space is divided into P = 4 sub-regions, and the bin space is divided into l = 2 regions. The corresponding temporal distributions of the red bins are the initial center of clusters.

Step 2: Assignment: In the second stage, our SLIC-based algorithm will assign a new center to each temporal distribution based on the similarity between temporal distributions to centers. However, our SLIC-based algorithm does not compute the similarity between all temporal distributions and centers. As shown in Fig. 5, a cluster center only computes the similarity with the temporal distributions within a local Grid-Bin region whose length of each dimension is twice longer than the Grid-Bin sub-block used in the initialization step. Earth Mover's Distance (EMD) [33] defines the similarity between two temporal distributions, a popular metric for measuring the distance between two probability distributions. If the EMD distance between a temporal distribution and the most similar cluster center is smaller than a given threshold, the temporal distribution is assigned to the cluster. If the EMD distance between a temporal distribution and the most similar cluster center is not smaller than a given threshold, the temporal distribution creates a new cluster, and the temporal distribution is the only member in this new cluster. After the similarity computation and the new cluster center assignment, we have to update the cluster center of each cluster. We first compute each cluster's average position in the Grid-Bin space. However, the temporal distribution at the average position may not belong to this cluster. We will find and assign the temporal distribution closest to the average position to be the new cluster center. We will iteratively run the above cluster assignment and cluster center updating steps until the number of iterations meets the user-given upper bound or the cluster centers become stable.

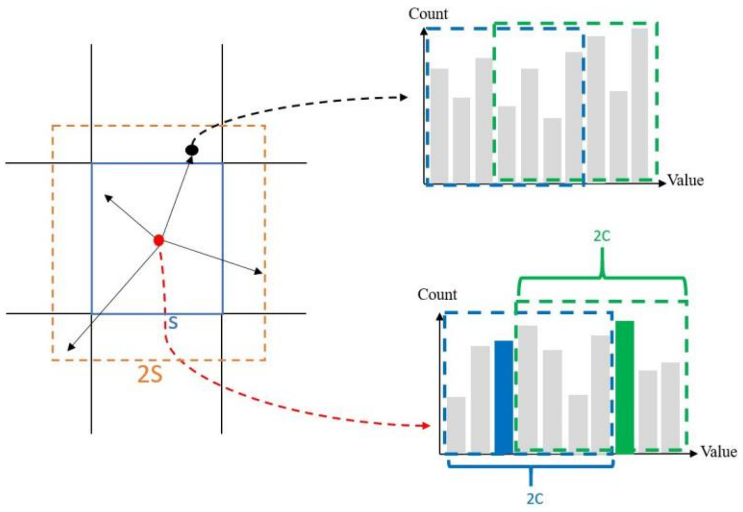


Fig. 5. The figure illustrates the SLIC-based algorithm search area to update the cluster center for a temporal distribution; The red point in the left figure shows a spatial point with at least one cluster center, and the red box shows the search area in the spatial space; The bottom left figure is an example that shows two cluster centers (blue and green bars) on the bin index domain, and the dotted boxes show the corresponding search region in the bin index domain.

Step 3: Represented Temporal GMM Creation: After the second step, we assign each bin to a cluster and each cluster is represented by the temporal GMM of the cluster center. Because we have a user-defined EMD threshold in the SLIC-based algorithm, the error between the actual temporal distribution and the temporal distribution of the corresponding cluster center must be bounded by the threshold. We only compute and store the temporal GMMs at the centers of clusters and store a pointer for each bin to indicate the represented temporal GMM. In this way, we can significantly reduce the storage space cost of temporal information.

4.4 Data Structure

Fig. 6 illustrates the data structure of our data representation. The data structure has two parts. One is bin information of the numerical distributions of all grid points (the blue block of Fig. 6). The information of every bin consists of two parts, one is the probability of the bin of the numerical distribution, and the other one is a pointer (model label) that points to the corresponding represented temporal distribution. Note that we use the sparse representation to store the bin information. If the probability of a bin is zero, we do not store the probability and the model label. The second part is the array of temporal GMMs (gray block of Fig. 6). Only the represented temporal GMMs computed from the SLIC-based algorithm are stored in this array. Each GMM stores weights, means, and standard deviations of the *k* Gaussian components. To access the corresponding temporal GMM of a bin of a grid point, we first get the "model label" of the bin and access the corresponding temporal GMM from the temporal GMMs array.

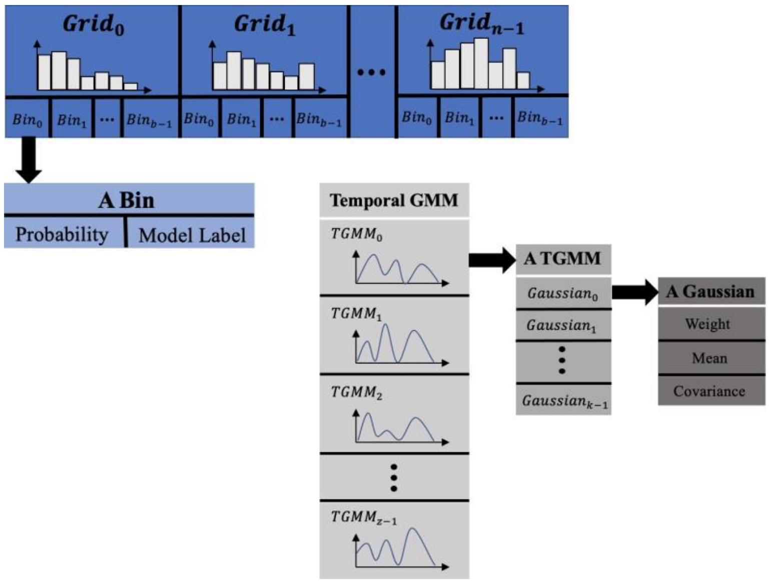


Fig. 6. Data structure of our data representation; There are two parts of our data structure, bin information, and temporal GMMs.

5. DATA RECONSTRUCTION

To visualize and analyze the uncertainty of the data, a basic operation is to access the PDF at any given time step and grid point. In Section 4.4, we have introduced the data structure of our data representation and how to access the represented temporal GMM of a given bin of a numerical distribution. This section will introduce how to compute a PDF at a given time step from the given numerical distribution and the corresponding temporal distributions.

5.1 PDF Computation by Bayes' Rule

The main purpose of data reconstruction is to compute the PDF at a given time step. We develop our data reconstruction method by Bayes' rule. The Bayes' rule is a wellknown theorem and it can combine the prior probability and evidence to compute the posterior probability. By analogy with our PDF computation, the posterior probability is the PDF at a time step, the prior probability comes from the numerical distribution, and the evidence is the probability at the time step from the temporal distribution. We can use Equation 2 to compute the probability of bin b_i of the PDF at time step t.

$$P(I = b_i \mid T = t) = \frac{f_T(t \mid I = b_i)P(I = b_i)}{f_T(t)} \tag{2}$$

$$f_T(t) = \sum_{n=0}^{B-1} f_T(t \mid I = b_n)P(I = b_n) \tag{3}$$

where $P(I = b_i \mid T = t)$ is the reconstructed probability of bin b_i of the PDF at the specified time t, $f_T(t|I = b_i)$ is the probability at time step t computed from the temporal GMM of bin b_i, and $P(I = b_i)$ is the probability of bin b_i of the numerical distribution. $f_T(t)$ in Equation 3 is the normalization factor for $P(I = b_i \mid T = t)$ to ensure the sum of the reconstructed PDF is one.

6. QUANTITATIVE EVALUATION

In this section, we use three datasets to show the storage cost and quality of our representation. The experiments are carried out on a computer with two Intel® Xeon® Platinum 8280L 28 Cores 2.7GHz with 90GB memory. The following is a short introduction to the three datasets. The first one is a 2D Global Climate Forecast ensemble dataset from the Research Center of Environmental Changes. The spatial resolution of the dataset is 160 320 and the temporal resolution of the dataset is 150 time steps. The second dataset is the dark matter density field of a cosmology ensemble dataset [34]. The spatial resolution of the data is 32^3 and the temporal resolution is 200 time steps. The third dataset is a Red Sea salinity ensemble dataset from the IEEE Visualization 2020 competition. The spatial resolution is $25\times20\times120$ and the temporal resolution is 60 time steps. Note that our work focuses on supporting the tasks of visualizing and analyzing the uncertainty in ensemble datasets, so the ground truth in the experiment is the data representation that represents the data values at each grid point by a histogram with 128 bins.

6.2 Reconstruction Quality

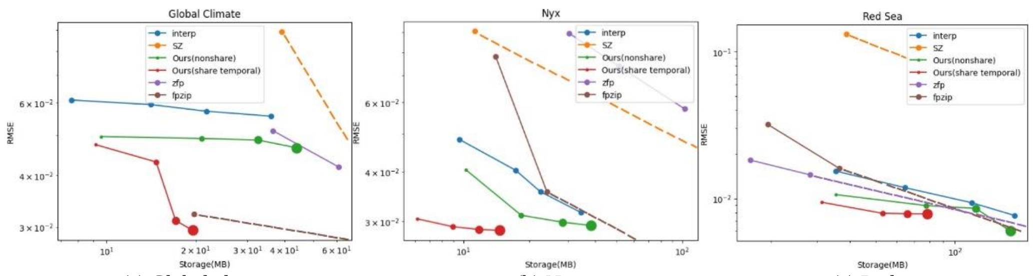
We compare our data representation with histogram interpolation [35], and SZ [8], ZFP [10] and FPZIP [9] lossy compression approaches. The histogram interpolation only stores histograms of grid points at the sampled time steps. The histograms at non-sampled time steps are computed from the histogram interpolation approach proposed by [35]. SZ, ZFP and FPZIP are popular lossy compression approaches for scientific datasets. These compression approaches compress the ground truth histograms of all time steps. We also compare our data representation with that without the temporal distribution sharing approach. To quantitatively evaluate the PDF estimation accuracy at a given grid point and a given time step, we use Root Mean Square Error (RMSE) as the metric.

$$RMSE(P_t^{\ell}, x_t^{\ell}) = \frac{\sqrt{\sum_{i=0}^{B-1}(P_t^{\ell}(b_i)) - x_t^{\ell}(b_i))^2}}{B} \tag{4}$$

where the variable t is a given time step, ℓ is a given grid point, P^{ℓ} is the reconstructed PDF at grid point p and time step t, and x_i^{ℓ} is the ground truth PDF at the grid pint p and the time step t. The RMSE of a time step is calculated by averaging the RMSE values from all grid points. A lower RMSE represents the reconstructed data is closer to the ground truth.

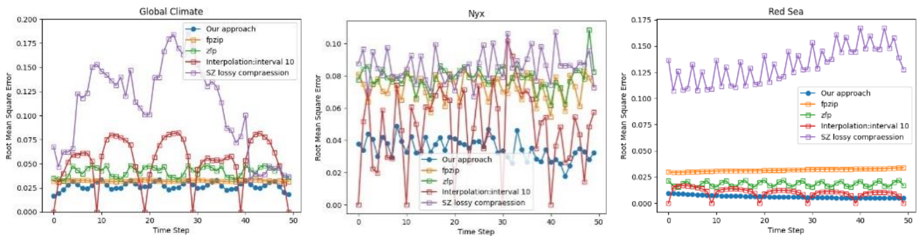
Fig. 7 shows the tradeoff between the representation quality and the storage cost among our approach and the alternatives. We control the number of Gaussian components to generate the lines for our approach with and without temporal GMM sharing. The line of the histogram interpolation is generated by controlling the number of stored time steps. For the SZ compression approach, we control the error bound to plot the line. However, the RMSE and storage change dramatically when we change the error-bound a little bit when a high compression rate is required. Therefore, we only generate two points and connect them by a dotted line, and only one point from SZ is shown to make sure the figures' scale is proper for comparison. For the FPZIP and ZFP compression approaches, we control the precision and rate to plot the lines, respectively. In Nyx dataset, the ZFP's RMSE values decrease when the compress rates are higher. We think that is because the grid does not change dramatically and compressed data happened close to the original data values.

In Fig. 7, a lower curve indicates a better trade-off between the representation quality and storage cost. The figures show that the trade-off of our approach outperforms the histogram representation with interpolation, and SZ, ZFP and FPZIP compression approaches across all three test datasets. In addition, Fig. 7 also shows that our approach significantly improves the trade-off when the temporal GMM sharing approach is used.



(a) Global climate (b) Nyx (c) Red sea

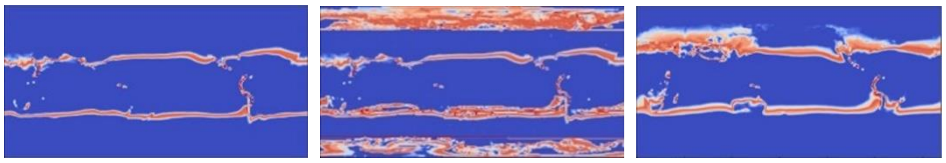
Fig. 7. The trade-off between RMSE and storage cost when comparing our approach using different Gaussian components with SZ compression, ZFP compression, FPZIP compression, and histogram interpolation. The size of the point represents the number of components of our method from small to large is 1, 3, 5, and 7.



(a) Global climate (b) Nyx (c) Red sea

Fig. 8. Quantitative evaluation of our approach, histogram interpolation, SZ lossy compression, Zfp lossy compression and Fpzip lossy compression by Root Mean Square Error.

Fig. 8 shows the RMSE at each time step to evaluate the reconstruction quality of our method with histogram interpolation and SZ lossy compression. We adjust the storage cost of our approach to be close to or even less than the other approaches. The figure shows that the reconstruction quality of our approach is better than lossy compression approaches at all time steps. The Histogram interpolation approach applied on Nyx has a lower error for some time steps. The reason is that the value range of Nyx is very large, which causes the values to be concentrated in specific bins and the data changes in the initial time step are small, so there are relatively low errors in some time steps. However, the reconstruction quality of our approach has a lower error at most time steps.


(a) Ground truth (359.47MB) (b) SZ (39.05MB) (c) Interpolation (35.97MB)

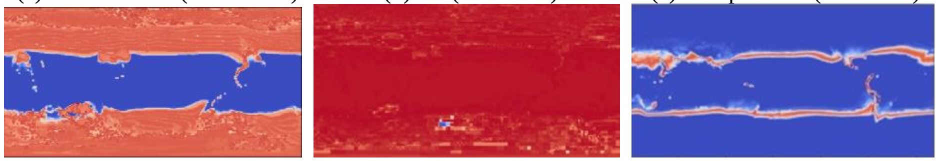
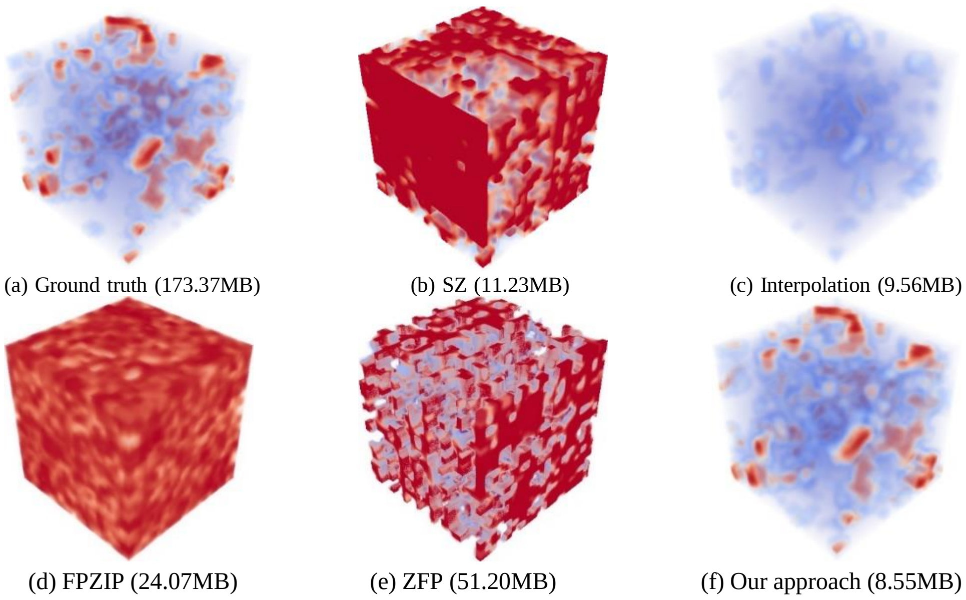

(d) FPZIP (19.73MB) (e) ZFP (60.80MB) (f) Our approach (19.75 MB)

Fig. 9. Uncertain isosurface (isovalue = 290) of the Global Climate dataset; (a) is the true isosurface from the raw data; (b) is the result of SZ compression with a high compression ratio; (c) is the result of the histogram interpolation method; The temporal sampling interval is 10 time steps; (d) is FPZIP compression and the precision is 8; (e) is ZFP compression and the rate is 0.5; (f) is our approach with 7 Gaussian components of the temporal GMM and the number of consecutive time steps is 50.

7. QUALITATIVE EVALUATION

In this section, we use two visualization techniques, uncertainty isosurface and re-sampling volume rendering to qualitatively compare the visualization results from our data representation and other alternatives. Uncertain isosurface computes level crossing probability field [36] to show the occurrence likelihood of a given isovalue. The resampling volume rendering [1] repeatedly generates a scalar field by resampling from the distributions on grid points and rendering the scalar field. The process can generate an animation that consists of the volume rendering images and the pixels with higher color variation indicates a higher uncertainty. In this section, we show one image frame from the animation. The number of consecutive time steps used to build one independent data representation in this experiment is 50, 50, and 60 for Global Climate, Nyx, and the Red Sea datasets, respectively. The number of Gaussian components for the Global Climate, Nyx, and Red Sea datasets is 7, 3, and 5, respectively. We will let the data size of the histogram interpolation and SZ, ZFP and FPZIP compression approaches be as close as possible to our data representation. We control their sizes by adjusting the temporal sampling rate for the histogram interpolation approach, error-bound for SZ compression, precision for FPZIP compression, and rate for ZFP compression, respectively.


(a) Ground truth (173.37MB) (b) SZ (11.23MB) (c) Interpolation (9.56MB)

(d) FPZIP (24.07MB) (e) ZFP (51.20MB) (f) Our approach (8.55MB)

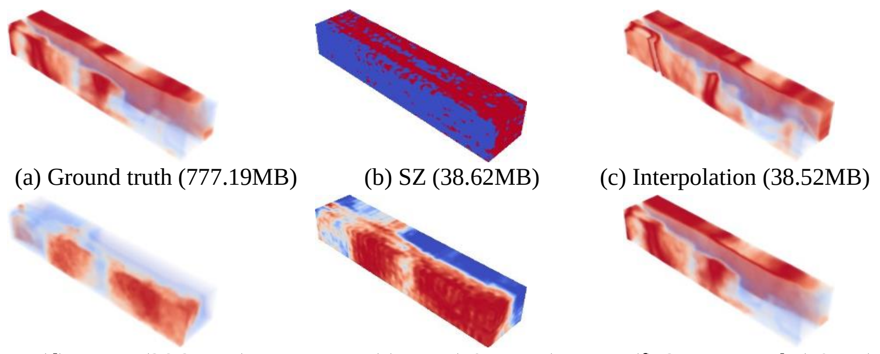
Fig. 10. Uncertain isosurface (isovalue = 1010) of the Nyx density dataset; (a) is the true isosurface from the raw data; (b) is the result of SZ compression with a high compression ratio; it shows a lot of artifacts; (c) is the result of the histogram interpolation method. The temporal sampling interval is 20 time steps; (d) is FPZIP compression and the precision is 8; (e) is ZFP compression and the rate is 0.5; (f) is our approach with 3 Gaussian components of the temporal GMM and the number of consecutive time steps is 50.

7.1 Uncertain Isosurface

We use the level-crossing probability [36] to compute the uncertain isosurface for ensemble datasets. It computes the probability of a given isovalue passing through the cells and renders the probability field. Eq. (5) shows the formula to compute the probability at a cell.

$$p_{crossing}(c) = 1 - \prod_{k=0}^{7} \int_{-\infty}^{c} P_{l_k}(v)dv - \prod_{k=0}^{7} \int_{c}^{\infty} P_{l_k}(v)dv \qquad (5)$$

where c is an isovalue a p_l is the PDF for the data values at the grid point, and P_{l_0} ... P_{l_7} represent eight grid points of a cell.


(a) Ground truth (777.19MB) (b) SZ (38.62MB) (c) Interpolation (38.52MB)

(d) FPZIP (36.32MB) (e) ZFP (28.12MB) (f) Our approach (29MB)

Fig. 11. Uncertain isosurface for isovalue 35.95567 of the Rea Sea salinity; (a) is the true isosurface from the raw data; (b) is the result of SZ compression with a high compression ratio; it loses a lot of data features; (c) is the result of the histogram interpolation method. The temporal sampling interval is 30 time steps; (d) is FPZIP compression and the precision is 8; (e) is ZFP compression and the rate is 0.5; (f) is our approach with 5 Gaussian components of the temporal GMM and the number of consecutive time steps is 60.

Figs. 9-11 show the images of the Global Climate, Nyx and Red Sea datasets at different time steps and isovalues. Figs. 9-11 (a) show the uncertain isosurface generated by ground truth data. Compared to alternatives, our approach (Figs. 9-11 (f)) shows more similar results to the ground truth images.

7.2 Resampling Volume Rendering

In this experiment, we show the visual comparison using resampling volume rendering. Figs. 12-14 show the image of the Global Climate, Nyx and Red Sea datasets at different time steps. Figs. 12-14 (a) are the ground truth resampling volume rendering image. Compared to the other approaches, our approach (Figs. 12-14 (f)) show much better image quality with less storage cost. In addition, Table 1 shows the data storage of the setting we use to generate images in this section. Note that the ground truth in the experiment is the data representation that represents the data values at each grid point by a histogram with 128 bins. And the interpolation approach only stores histograms of grid points at the sampled time steps. We can observe that our approach can provide good visualization quality with a high data reduction rate. In addition, with similar data storage usage, our data representation quality is significantly better than alternatives.

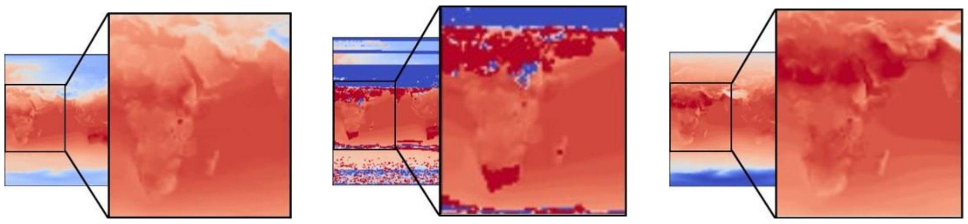

(a) Ground truth (359.47MB) (b) SZ (39.05MB) (c) Interpolation (35.97MB)

Fig. 12. Visual comparison of resampling volume rendering in Global climate dataset at time step 144; (a) is the ground truth image from the raw data; (b) is the result of SZ compression with a high compression ratio; (c) is the result of the histogram interpolation method.

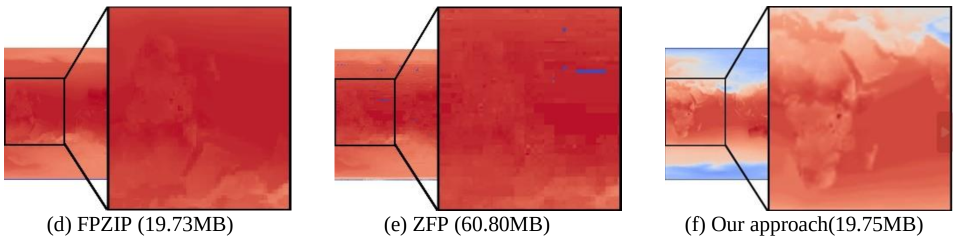
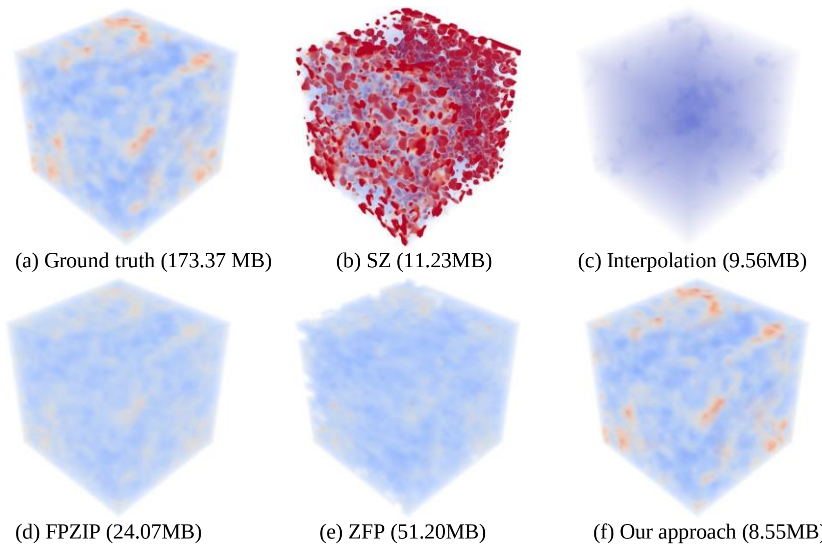


Fig. 12. Visual comparison of resampling volume rendering in Global climate dataset at time step 144; The temporal sampling interval is 10 time steps. (d) is FPZIP compression and the precision is 8 (e) is ZFP compression and the rate is 0.5 (f) is our approach using 7 Gaussian, and the number of consecutive time steps is 50.


(a) Ground truth (173.37 MB)
(b) SZ (11.23MB)
(c) Interpolation (9.56MB)
(d) FPZIP (24.07MB)
(e) ZFP (51.20MB)
(f) Our approach (8.55MB)
Fig. 13. Visual comparison of resampling volume rendering in Nyx density dataset at time step 95; (a) is the ground truth image from the raw data; (b) is the result of SZ compression with a high compression ratio; (c) is the result of the histogram interpolation method. The temporal sampling interval is 10 time steps; (d) is FPZIP compression and the precision is 8; (e) is ZFP compression and the rate is 0.5; (f) is our approach using 3 Gaussian, and the number of consecutive time steps is 50.

8. DISCUSSION

Data Processing Time: We report the processing time per time step of our data representation of the three datasets used for our experiments. We measure the processing time of our data representation using the same settings and the same machine in Section 6. The data processing time varies because of the complexity of the data. If the dataset has a higher spatial resolution, the processing time is usually longer because we have to process more data values, and the time for the SLIC-based algorithm also significantly increases. In addition, the processing time will increase if the number of Gaussian components used in the data representation increases. In our experiment, the computation time in average per time step is 17.86 seconds for the Global Climate dataset, 11.77 seconds for the Nyx dataset, and 115.2 seconds for the Red Sea dataset.

Reconstruction Time: The data reconstruction process of data point at any given time step and grid point, as described in Section 5, is independent. This implies that our data reconstruction program lends itself to parallelization. Therefore, we implement our reconstruction algorithm by VTK-*m* library [25]. VTK-m not only provides many parallel implementations of scientific data processing and visualization algorithms but also provides a platform for developing parallel programs on multi-core processors and GPU. We conduct this experiment on a Tesla V100 GPU and measure the execution time. Table 2 shows the data reconstruction time of one time step of the datasets. In addition, we also implement the process of generating the resampling volume and the probability field of uncertain isosruface from a reconstruction dataset. Because generating the resampling volume and the probability field of uncertain isosurface require extra work after reconstructing the data, these two tasks take slightly longer than purely reconstructing the data. The data point of individual grid points can be reconstructed independently; so with the parallel implementation and running it on a decent GPU, our data representation and data reconstruction algorithm can potentially serve the need for interactive data exploration.

Impact of Parameters: We have four hyperparameters in our approach. The first is the number of Gaussians of the temporal GMM. The second is the number of consecutive time steps used to construct an independent data representation. The other two are the number of initial clusters and the threshold for determining if temporal distributions are sufficiently similar. We carry out experiments for the studies of these four parameters. Figs. 15-17 show the results of the three datasets.

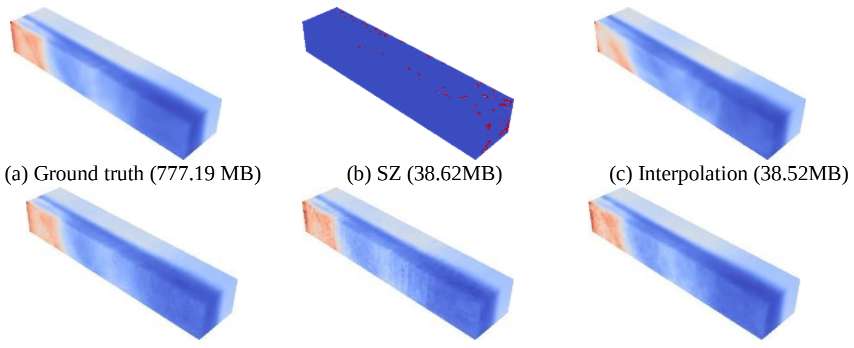
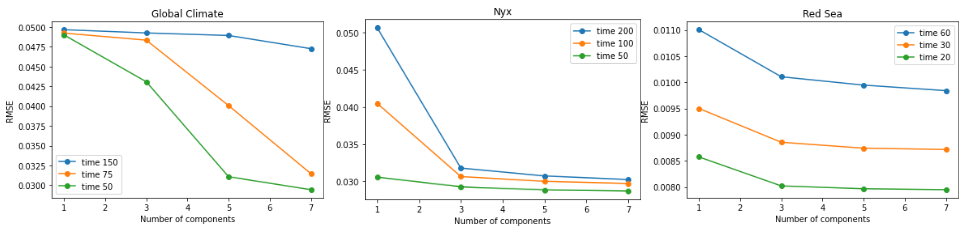

(a) Ground truth (777.19 MB) (b) SZ (38.62MB) (c) Interpolation (38.52MB)
(d) FPZIP (36.32MB) (e) ZFP (28.12MB) (f) Our approach (29MB)

Fig. 14. Visual comparison of resampling volume rendering in Red Sea dataset at time step 50; (a) is the ground truth image from the raw data; (b) is the result of SZ compression with a high compression ratio; (c) is the result of the histogram interpolation method. The temporal sampling interval is 30 time steps; (d) is FPZIP compression and the precision is 8; (e) is ZFP compression and the rate is 0.5; (f) is our approach using 5 Gaussian, and the number of consecutive time steps is 30.

• *Consecutive Time Steps:* Fig. 15 shows that if other parameters are fixed and the number of consecutive time steps used to build one independent data representation decreases, the information is easier to describe and the RMSE will decrease. In addition, using larger consecutive time steps to build one independent data representation creates less numbers of independent data representations for one dataset and reduces the data representation size. Therefore, if users can afford a larger data representation size, a smaller number of consecutive time steps can be used. Otherwise, a larger number of consecutive time steps can be chosen to sacrifice the quality for smaller storage usage.



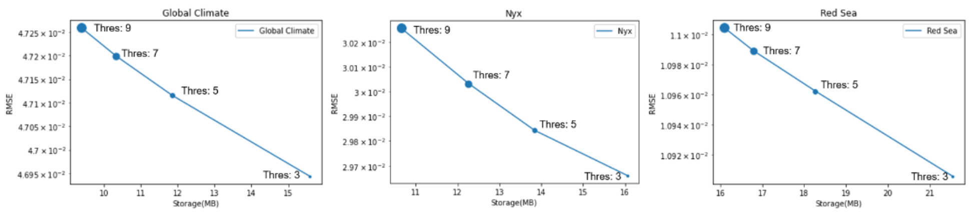
(a) Global Climate (b) Nyx (c) Red Sea

Fig. 15. Comparison of the effect of different time intervals and the number of Gaussian components on RMSE. The color of the curves is the number of consecutive time steps used to build one independent data representation. Each point represents the number of Gaussian of the temporal GMM.

Table 1. Data storage summary of each dataset.

Data Sets	Approach	RMSE	Memory (MB)	Reduction
Global Climate	SZ	0.0889	39.05	89.13%
	Interpolation	0.0556	35.97	89.9%
	FPZIP	0.0322	19.73	94.51%
	ZFP	0.0419	60.80	83.08%
	Ours	0.0271	19.75	94.5%
Nyx	SZ	0.0906	11.23	93.52%
	Interpolation	0.0483	9.56	94.48%
	FPZIP	0.0356	24.07	86.11%
	ZFP	0.0740	51.20	70.46%
	Ours	0.02857	8.55	95.06%
Red Sea	SZ	0.1315	38.62	95.03%
	Interpolation	0.0153	38.52	95.04%
	FPZIP	0.0160	36.32	93.40%
	ZFP	0.0144	28.12	96.39%
	Ours	0.0087	29.00	96.26%

● *Gaussian Component Count:* Fig. 15 also demonstrates that if other parameters are fixed and the number of Gaussian components increases for each GMM, the data representations receive lower RMSE values in different datasets. However, diminishing marginal returns can also be observed when the number of Gaussian components increases. In addition, increasing the Gaussian count also increases the data representation storage. Therefore, selecting a proper Gaussian count can provide a better tradeoff between the data representation quality and storage size. Akaike Information Criterion (AIC) [37] or Bayesian Information Criterion (BIC) [38] often helps with the tasks of Gaussian count selection. By using AIC or BIC, the different Gaussian counts of GMMs in our data representation can choose and the data representation can even receive a better tradeoff between the data representation quality and storage size. However, evaluating the AIC or BIC scores requires modeling data by GMMs with different Gaussian counts and it will significantly increase the data modeling time. Thus, we suggest that if longer data modeling is affordable, users can use AIC or BIC to assign different numbers of Gaussians to different GMMs to get a better tradeoff between the data representation and data storage size. Otherwise, users can sample a subset of data, such as 5% of the dataset, to evaluate the AIC or BIC scores and determine one Gaussian count for all GMMs of the dataset.



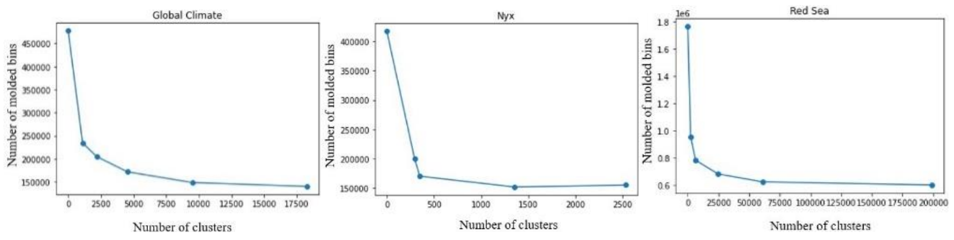
(a) Global climate (b) Nyx (c) Red sea

Fig. 16. Tradeoff between RMSE and storage cost at different thresholds.

• *Initial Cluster Count and Threshold for Cluster Similarity:* Fig. 16 demonstrates the tradeoff between the RMSE and storage costs at different thresholds. If the threshold value increases, the number of bins that share temporal information will increase, resulting in reduced storage costs, but the RMSE will increase slightly. This is because the more bins can share temporal information, the number of the temporal GMMs can be reduced. The Fig. also shows that the increase in RMSE in the limited threshold range is low, but can effectively reduce the cost of storage.

Table 2. Data reconstruction, volume resampling, and uncertain isosurface generation time. The time reported is measured in seconds.

Data Sets	Reconstruction	Volume Resampling	Uncertain Isosurface
Global Climate	0.0164s	0.0165s	0.0167s
Nyx	0.0104s	0.0107s	0.0129s
Red Sea	0.0136s	0.0162s	0.0165s

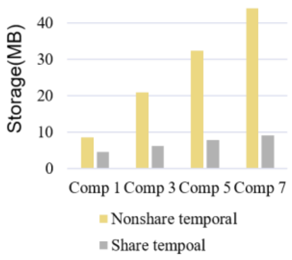


(a) Global Climate (b) Nyx (c) Red Sea

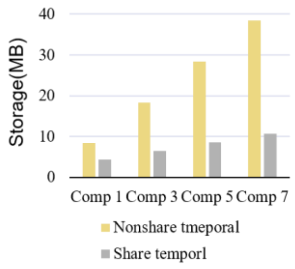
Fig. 17. Comparison of the effect of sharing temporal information for various clusters number.

If storage cost reduction is the primary goal, users can use a higher threshold value to obtain a better compression ratio. Fig. 17 shows the effect of sharing temporal information for different numbers of clusters at the same threshold. We can observe that the number of temporal GMMs decreases as the number of clusters increases. However, as the number of clusters increases to a certain extent, the downward trend of the curve will become slower, and the number of temporal GMMs even increases in Fig. 17 (b). Because there are too many clusters, the search range of each cluster is too small to effectively share temporal information of bins.

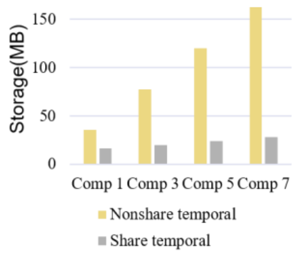
Effect of temporal distribution sharing: In order to further reduce the storage cost, we propose a SLIC-based algorithm in Section 4.3 to achieve temporal distribution sharing. Fig. 18 demonstrate the impact of temporal distribution sharing the yellow rectangle represents our approach without temporal sharing and the gray rectangle represents our approach with temporal sharing. Figs. 18 (a)-(c) compare the storage cost under different numbers of Gaussian components. As the number of Gaussian components increases, the cost of storage increases considerably without using the temporal sharing method. If the sharing method is used, the cost of storage will only increase slightly, and the cost of storage will be substantially reduced from the method without temporal information sharing. Figs. 18 (d)-(f) compare the RMSE under the different number of Gaussian components. We can observe that the RMSE of the gray rectangles is very close to the yellow rectangles, which means that the use of temporal distribution sharing does not significantly degrade the quality of the representation. This experiment demonstrates that our temporal distribution sharing method can significantly reduce storage costs and maintain high data representation quality.



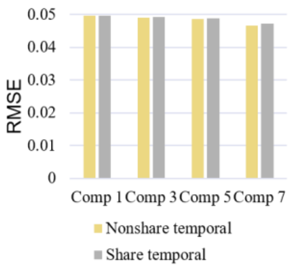
(a) Global climate



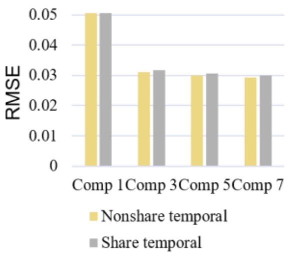
(b) Nyx



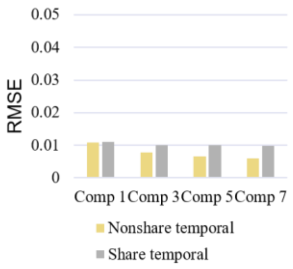
(c) Red sea



(d) Global climate



(e) Nyx



(f) Red sea

Fig. 18. Comparison of our approach with and without temporal GMM sharing in terms of the storage and RMSE.

9. CONCLUSION AND FUTURE WORK

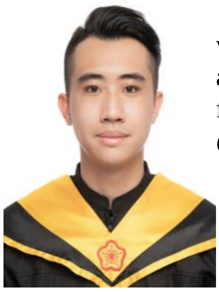
This paper proposes a compact distribution-based representation for the uncertainty visualization and analysis of time-varying ensemble data. We transform adjacent probability density functions in the temporal domain into one numerical distribution and several temporal distributions. We use the Gaussian mixture model to represent the temporal distribution because the Gaussian mixture model can well represent the temporal distribution with less storage space. In addition, we use a SLIC-based algorithm to identify similar temporal distributions. Similar temporal distributions are represented by one represented temporal distribution to save more storage costs. As the results are shown in our experiment, our data representation can represent the time-varying ensemble data with a better tradeoff between the quality and storage cost. In future work, we will extend our approach to multivariate ensemble datasets. Because multivariate ensemble datasets will require a higher storage cost, we hope to develop better data representations to support efficient multivariant and time-varying ensemble data exploration and analysis. In addition, In addition, the deep learning approach is one potential approach to deal with large-scale scientific data problems. A model can be trained to take a time step and a grid point coordinate as input and predict the corresponding value distribution.

REFERENCES

1. S. Liu, J. A. Levine, P.-T. Bremer, and V. Pascucci, “Gaussian mixture model based volume visualization,” in *Proceedings of IEEE Symposium on Large Data Analysis and Visualization*, 2012, pp. 73-77.
2. D. Thompson, J. A. Levine, J. C. Bennett, P.-T. Bremer, A. Gyulassy, V. Pascucci, and P. P. Pe´bay, “Analysis of large-scale scalar data using hixels,” in *Proceedings of IEEE Symposium on Large Data Analysis and Visualization*, 2011, pp. 23-30.
3. A. Luo, D. Kao, and A. Pang, “Visualizing spatial distribution data sets,” in *VisSym*, Vol. 3, 2003, pp. 29-38.
4. K. Pöthkow and H.-C. Hege, “Nonparametric models for uncertainty visualization,” in *Computer Graphics Forum*, Vol. 32, 2013, pp. 131-140.
5. C.-M. Chen, A. Biswas, and H.-W. Shen, “Uncertainty modeling and error reduction for pathline computation in time-varying flow fields,” in *Proceedings of IEEE Pacific Visualization Symposium*, 2015, pp. 215-222.
6. K.-L. Ma, “In situ visualization at extreme scale: Challenges and opportunities,” *IEEE Computer Graphics and Applications*, Vol. 29, 2009, pp. 14-19.
7. H. Yu, C. Wang, R. W. Grout, J. H. Chen, and K.-L. Ma, “In situ visualization for large-scale combustion simulations,” *IEEE Computer Graphics and Applications*, Vol. 30, 2010, pp. 45-57.
8. S. Di and F. Cappello, “Fast error-bounded lossy HPC data compression with SZ,” in *Proceedings of IEEE International Parallel and Distributed Processing Symposium*, 2016, pp. 730-739.
9. P. Lindstrom and M. Isenburg, “Fast and efficient compression of floating-point data,” *IEEE Transactions on Visualization and Computer Graphics*, Vol. 12, 2006, pp. 1245-1250.
10. P. Lindstrom, “Fixed-rate compressed floating-point arrays,” *IEEE Transactions on Visualization and Computer Graphics*, Vol. 20, 2014, pp. 2674-2683.
11. R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, “Slic superpixels compared to state-of-the-art superpixel methods,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 34, 2012, pp. 2274-2282.

12. A. Tikhonova, C. D. Correa, and K.-L. Ma, "Visualization by proxy: A novel framework for deferred interaction with volume data," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 16, 2010, pp. 1551-1559.
13. R. Sicat, J. Krüger, T. Möller, and M. Hadwiger, "Sparse pdf volumes for consistent multi-resolution volume rendering," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 20, 2014, pp. 2417-2426.
14. J. Woodring, J. P. Ahrens, J. Figg, J. Wendelberger, S. Habib, and K. Heitmann, "In-situ sampling of a large-scale particle simulation for interactive visualization and analysis," *Computer Graphics Forum*, Vol. 30, 2011, pp. -- --.
15. P. O'Leary, J. Ahrens, S. Jourdain, S. Wittenburg, D. H. Rogers, and M. Petersen, "Cinema image-based in situ analysis and visualization of MPAS-ocean simulations," *Parallel Computing*, Vol. 55, 2016, pp. 43-48.
16. W. He, J. Wang, H. Guo, K.-C. Wang, H.-W. Shen, M. Raj, Y. S. Nashed, and T. Peterka, "InSituNet: Deep image synthesis for parameter space exploration of ensemble simulations," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 26, 2019, pp. 23-33.
17. C.-M. Chen, A. Biswas, and H.-W. Shen, "Uncertainty modeling and error reduction for pathline computation in time-varying flow fields," in *Proceedings of IEEE Pacific Visualization Symposium*, 2015, pp. 215-222.
18. S. Li, N. Marsaglia, C. Garth, J. Woodring, J. Clyne, and H. Childs, "Data reduction techniques for simulation, visualization and data analysis," in *Computer Graphics Forum*, Vol. 37, 2018, pp. 422-447.
19. T.-Y. Lee and H.-W. Shen, "Efficient local statistical analysis via integral histograms with discrete wavelet transform," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 19, 2013, pp. 2693-2702.
20. K.-C. Wang, K. Lu, T.-H. Wei, N. Shareef, and H.-W. Shen, "Statistical visualization and analysis of large data using a value-based spatial distribution," in *Proceedings of IEEE Pacific Visualization Symposium*, 2017, pp. 161-170.
21. K.-C. Wang, N. Shareef, and H.-W. Shen, "Image and distribution based volume rendering for large data sets," in *Proceedings of IEEE Pacific Visualization Symposium*, 2018, pp. 26-35.
22. G. Li, J. Xu, T. Zhang, G. Shan, H.-W. Shen, K.-C. Wang, S. Liao, and Z. Lu, "Distribution-based particle data reduction for in-situ analysis and visualization of large-scale n-body cosmological simulations," in *Proceedings of IEEE Pacific Visualization Symposium*, 2020, pp. 171-180.
23. S. Hazarika, S. Dutta, H.-W. Shen, and J.-P. Chen, "Codda: A flexible copula-based distribution driven analysis framework for large-scale multivariate data," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 25, 2019, pp. 1214-1224.
24. H.-Y. Yang, Z.-R. Lin, and K.-C. Wang, "Efficient and portable distribution modeling for large-scale scientific data processing with data-parallel primitives," *Algorithms*, Vol. 14, 2021, p. 285.
25. K. Moreland, C. Sewell, W. Usher, L.-T. Lo, J. Meredith, D. Pugmire, J. Kress, H. Schroots, K.-L. Ma, H. Childs *et al.*, "Vtk-m: Accelerating the visualization toolkit for massively threaded architectures," *IEEE Computer Graphics and Applications*, Vol. 36, 2016, pp. 48-58.

26. S. Dutta, C.-M. Chen, G. Heinlein, H.-W. Shen, and J.-P. Chen, "In situ distribution guided analysis and visualization of transonic jet engine simulations," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 23, 2016, pp. 811-820.
27. S. Hazarika, A. Biswas, and H.-W. Shen, "Uncertainty visualization using copula-based analysis in mixed distribution models," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 24, 2018, pp. 934-943.
28. J. Wang, X. Liu, H.-W. Shen, and G. Lin, "Multi-resolution climate ensemble parameter analysis with nested parallel coordinates plots," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 23, 2017, pp. 81-90.
29. J. Sanyal, S. Zhang, J. Dyer, A. Mercer, P. Amburn, and R. Moorhead, "Noodles: A tool for visualization of numerical weather model ensemble uncertainty," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 16, 2010, pp. 1421-1430.
30. C. Sun and K.-C. Wang, "DLA-VPS: Deep learning assisted visual parameter space analysis of cosmological simulations," *IEEE Computer Graphics and Applications*, Vol. 42, 2022, pp. 41-52.
31. T. Höllt, A. Magdy, P. Zhan, G. Chen, G. Gopalakrishnan, I. Hoteit, C. D. Hansen, and M. Hadwiger, "OVIS: A framework for visual analysis of ocean forecast ensembles," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 20, 2014, pp. 1114-1126.
32. J. Wang, S. Hazarika, C. Li, and H.-W. Shen, "Visualization and visual analysis of ensemble data: A survey," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 25, 2019, pp. 2853-2872.
33. A. Y. Fu, L. Wenyin, and X. Deng, "Detecting phishing web pages with visual similarity assessment based on earth mover's distance (EMD)," *IEEE Transactions on Dependable and Secure Computing*, Vol. 3, 2006, pp. 301-311.
34. A. S. Almgren, J. B. Bell, M. J. Lijewski, Z. Lukić, and E. van Andel, "Nyx: A massively parallel AMR code for computational cosmology," *Astrophysical Journal*, Vol. 765, 2013, p. 39.
35. A. Read, "Linear interpolation of histograms," *Nuclear Instruments and Methods in Physics Research. Section A, Accelerators, Spectrometers, Detectors and Associated Equipment*, Vol. 425, 1999, pp. 357-360.
36. T. Athawale, E. Sakhaee, and A. Entezari, "Isosurface visualization of data with nonparametric models for uncertainty," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 22, 2016, pp. 777-786.
37. H. Akaike, "Information theory and an extension of the maximum likelihood principle," *Selected Papers of Hirotugu Akaike*, 1998, pp. 199-213.
38. G. Schwarz, "Estimating the dimension of a model," *The Annals of Statistics*, Vol. 6, 1978, pp. 461-464.



Zhi-Rong Lin is with National Taiwan Normal University, Taiwan. His research interests include large-scale scientific data analysis and visualization. Lin received his MS degree in Computer Science from National Taiwan Normal University. Contact him at 60947081s @ntnu.edu.tw.

An-Lung Hsiao is a master's student at National Taiwan Normal University. He is majoring in Computer Science and Information Engineering and his research interests include data visualization and data analysis. Contact him at 61147063s@ntnu.edu.tw.

Guan Li is an Assistant Professor at Computer Information Network Center, Chinese Academy of Sciences. He received his Ph.D. degree in Computer Software and Theory from University of Chinese Academy of Sciences in 2021. His research interests include scientific visualization and visual analytics.

Ko-Chih Wang is an Associate Professor at National Taiwan Normal University, Taipei, 116, Taiwan. His research interests include large-scale scientific data visualization, visual analytics, computer graphics, and high-performance computing. Wang received his Ph.D. degree in Computer Science from The Ohio State University. He is the corresponding author of this article. Contact him at kcwang @ntnu.edu.tw.