Statistical Super Resolution for Data Analysis and Visualization of Large Scale Cosmological Simulations

Ko-Chih Wang[*]
The Ohio State University

Jiayi Xu[†]
The Ohio State University

Jonathan Woodring[‡]
Los Alamos National Laboratory

Han-Wei Shen[§]
The Ohio State University

Abstract

Cosmologists build simulations for the evolution of the universe using different initial parameters. By exploring the datasets from different simulation runs, cosmologists can understand the evolution of our universe and approach its initial conditions. A cosmological simulation nowadays can generate datasets on the order of petabytes. Moving datasets from the supercomputers to post data analysis machines is infeasible. We propose a novel approach called statistical super-resolution to tackle the big data problem for cosmological data analysis and visualization. It uses datasets from a few simulation runs to create a prior knowledge, which captures the relation between low- and high-resolution data. We apply *in situ* statistical down-sampling to datasets generated from simulation runs to minimize the requirements of I/O bandwidth and storage. High-resolution datasets are reconstructed from the statistical down-sampled data by using the prior knowledge for scientists to perform advanced data analysis and render high-quality visualizations.

1 Introduction

Exploring and modeling parameters of the initial conditions of our universe is one of the most important tasks in cosmology, which can help cosmologists to build more precise universe models [3, 7]. Recently, a cosmological simulation code called Nyx, developed by Lawrence Berkeley National Laboratory, is used for simulating large-scale cosmological phenomena [3]. It requires performing many cosmological simulations to conduct a large-scale, multi-dimensional parameter study to find the "best matching" set of parameters that matches the current state of the observable universe. In addition, every cosmological simulation output is 'scientifically rich' [27]. Cosmologists may discover new insights into the physical model by interactively exploring and analyzing the datasets [13]. When the size of parameter study is small, the data analysis pipeline consists of running the simulation with different parameter inputs, writing the output to disk, and repeatedly loading the data from disk to visualize and analyze the datasets. To compare ensemble simulation runs with observations and understand the evolution of the universe, it requires high-resolution simulations to match the desired accuracy. This easily produces datasets at the order of petabytes, and the traditional data analysis pipeline becomes infeasible because of disk bandwidth constraints and limited capacity of storage devices. Disk bandwidth constraints result in longer I/O time. Also, only a portion of the output can be kept in disk due to the limited capacity of disks. Therefore, it is essential to develop an approach to improve the whole cosmological data analysis pipeline.

To tackle such large-scale data problem, the concept of *in situ* data processing has been proposed. *In situ* techniques use supercomputer resources to generate compact data representations for analysis without moving the raw datasets. A naive approach is to down-sample the datasets, but the quality of this data representation would be insufficient for many data analysis tasks [28]. Statistically modeling the datasets can quantify the error of the down-sampled datasets, but the quality of the data representation is not improved significantly [10, 20]. Wang et al. [33] and Soumya et al. [12] propose accurate and compact data representations, but they need a long data processing time. Other *in situ* techniques produce the data representations for specific data analysis or visualization requirements [1, 13, 34].

The goal of this work is to develop an efficient technique which can compactly represent multi-variate (quantity) cosmological datasets and used for various data analysis and visualization tasks. We propose an *in situ* technique called *statistical-based super-resolution* (SbSR) for cosmological simulations. In computer vision and image processing community, the super-resolution technique [18, 36] enhances the resolution of an image using a prior knowledge database to predict high-resolution images from low-resolution images. Prior knowledge in our statistical-based super-resolution is created by the raw data from a small portion of simulation runs and used to reconstruct down-sampled data with other simulation parameters. The down-sampling is accomplished in situ by a Gaussian Mixture Model (GMM). We use GMMs to compactly represent low-resolution cosmology data, because data values of some quantities (e.g., *density*), from different cosmological simulation runs, can spread in an extremely wide value range, and a 3D down-sampled region can contain a large number of data values. Distributions provide richer information to facilitate the process of high-resolution data reconstruction. To minimize computation time for *in situ* statistical down-sampling, a parallel Expectation-Maximization (EM) algorithm based on the VTK-m framework [25] is implemented to train multiple GMMs concurrently by utilizing either GPU or multi-threads CPU. We show that our *in situ* statistical down-sampling can significantly reduce data storage and does not increase the total simulation time.

The remainder of this paper is organized as follows. Section 2 discusses related work. Section 3 is an overview of our approach. Section 4 describes the in situ statistical down-sampling. The prior knowledge creation is introduced in Section 5. Section 6 presents the process to reconstruct statistical down-sampled data to high-resolution data. Section 7 shows the quantitative and qualitative evaluations. Section 8 is the discussion of our approach. Finally, Section 9 concludes the paper and provides future work.

2 Related Work

Data representation for large datasets: Data representation plays an important role in large data analysis to reduce the memory footprint and save the I/O time. A traditional and straightforward approach of reducing the size of volume data is to down-sample the volume to a smaller resolution. However, this approach suffers from aliasing and inconsistent artifacts, as pointed out in [20, 28, 37]. Traditional statistical data reduction approaches, such as curve fitting, or PCA, may also suffer from inconsistent artifacts if a higher data reduction rate is required. The compression technique is the most popular technique to reduce the I/O overhead [5, 15, 21]. In the past decade, more sophisticated lossy compression techniques for scien-

[*] e-mail: wang.3182@osu.edu

[†] e-mail:xu.2205@osu.edu

[‡] e-mail:woodring@lanl.gov

[§] e-mail:shen.94@osu.edu

tific data analysis are developed studied in this paper. ISABELA [17] and SZ [8, 29] are two popular lossy compression techniques. Both of them model the data points in local regions by fitting curves. Because data values in most of the scientific data are changed smoothly in the local region, storing a few parameters of the model are often sufficient to represent the data well. However, a large error may be introduced if the dataset has an extreme value ranges with a high compression rate. Statistical summarization preserves distribution data for reconstruction. Dutta et al. [12] irregularly subdivided a volume by minimizing the data distribution entropy in each subregion to both reduce the proxy size and preserve the representation quality. However, their *in situ* data processing of one time step is much longer than the raw data I/O time for one time step. Wang et al. [33] used spatial distribution to improve the representation quality of statistical data summarization. However, they require around 1 hour to process a volume which contains only 2.5 million points because they need to estimate parameters from many multi-variate GMMs. Wei et al. [35] used stratified sampling to preserve both data distribution of each low-resolution block and samples' location distribution in the spatial domain for the accurate data reconstruction. Hazarika et al. [16] proposed a flexible distribution driven analysis framework, called CoDDA, targeted at *in situ* modeling of multivariate data. They used copula functions to model the correlation among multiple physical and spatial variables in their framework. The simulations which generate time-varying datasets can easily have hundreds or thousands of time steps, and simply create a huge amount of data. Chen et al. [6] used a quadratic Bezier curve to calculate the vector between two sampled time steps and compute the reliable pathlines. Gau et al. [14] used temporal coherence to increase rendering performance. In addition, the state-of-the-art data reduction techniques for scientific data visualization and analysis are summarized by Meyer et al. [24] and Li et al. [19]. The survey papers discussed and categorized the approaches in term of different use cases and techniques. Wang et al. [32] surveyed works related to scientific ensemble datasets, which includes data reduction techniques for ensemble datasets.

***In situ* technique:** *In situ* workflows tackle the constraint of the I/O bandwidth and disk capacity, by processing the datasets to produce compact data proxies using the same supercomputer resources. Ma et al. [22] pointed out the challenges of *in situ* and discussed the future directions of *in situ* applications. Tikhnova et al. [30, 31] proposed image-based in situ techniques which allow users to change the transfer function and explore volume datasets by storing a small number of image slices used to approximate pixel color. Ahrens et al. [1, 2, 26] proposed an image-based approach which directly renders a set of images in the supercomputer to reduce the data movement. Scientists can interactively explore the dataset via these images. Wang et al [34] used distribution to compactly model data projected to each image pixel to support volume rendering with transfer function exploration and error quantification. These approaches were targeted on data analysis tasks by volume rendering and may not available on other data analysis tool. Dutta et al. [10] modeled the dataset from a jet engine simulation by a set of Gaussian Mixture Model to reduce the data size and visualize the jet engine behavior by these GMMs only. Dutta et al. [11] proposed a prediction driven approach to the same application. It requires a small number of training set from the domain expert to train the model for the jet engine stall prediction. The model is used to predict the region of the jet engine stall and they only output the data around the stall region to persistent disk for analysis. Recently, the community starts to compare and integrate in situ and in-transit approaches. In-transit technique tries to tackle the same problem. In-transit technique could have better performance than in situ technique, but needs more invasive modification in simulation code. Bennett et al. [4] proposed an hybrid approach which combines in situ and in-transit workflows to enable the analysis on multiple scientific applications. Friesen et al. [13] compared in situ and in-transit workflows by using two popular analysis scenarios on cosmological simulation.

3 Overview

The Nyx cosmological simulation [3] solves compressible hydrodynamics with an N-body treatment of the dark matter. The simulation can have up to 10 different simulation input parameters. Fixing the values for these simulation input parameters will produce one *simulation run*. In each simulation run, Nyx produces both derived quantities volumes and particle data. In this work, we focus on data analysis of the derived quantities. Nyx can output up to seven derived quantities, *density*, *xmom*, *ymom*, *zmom*, *Temp*, *rho_e* and *phi_grav*. Each quantity is a regular grid and can have up to 4096^3 grid points according to the settings of the simulation. Nyx produces hundreds of time steps in each simulation run. The number of time steps varies according to the converging speed of the simulation run. Therefore, if the domain experts have P simulation input parameters of interest to study, have Q quantities of interest to study, take I value samples on each input parameter, set the output resolution of each quantity of interest to R^3 and each simulation run has T time steps, the scale of raw data size will be $O(I^P * T * Q * R^3)$. This raw data size usually is hundreds of TBs to several PBs. To output such huge datasets to disks and access these datasets in a post data analysis pipeline is challenging. We propose an *in situ* statistical super-resolution to address this challenge.

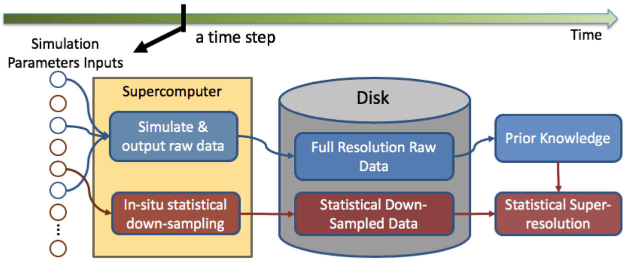


Figure 1: We illustrate the overview of the workflow of our proposed technique.

Figure 1 shows the workflow of our method for a single time step and a single quantity of Nyx. The leftmost circles in Figure 1 are simulation parameter inputs and each one is fed to Nyx to perform one simulation run. Our technique uses latin hypercube sampling [9] to draw a small number of simulation parameter inputs to run the simulation (blue arrows). These simulation runs will write the full resolution data to storages. These full resolution datasets are used to compute the prior knowledge. For other simulation parameter inputs (e.g. brown circles), we perform statistical down-sampling *in situ* to reduce the datasets generated from the simulation. A local data block size, B, is pre-set by users and a point (ℓ_p) in the low-resolution space is created from a B^3 data block (h_p) in the raw (high) resolution. When performing post data analysis, our approach uses the information from the prior knowledge to reconstruct the statistical down-sampled data to high-resolution data. We create independent prior knowledge for each quantity of interest. Because Nyx produces multiple time step data and the data has coherence in the temporal domain, we create the prior knowledge only at every τ time steps. Figure 2 illustrates this scheme. When reconstructing the statistically down-sampled data at time step t (red dot in Figure 2) to high resolution, the closest prior knowledge in the temporal domain is used.

4 *In situ* Statistical Down-sampling

A way to reduce the size of any dataset is to down-sample the data and store it at a lower resolution. The down-sampling process usu-

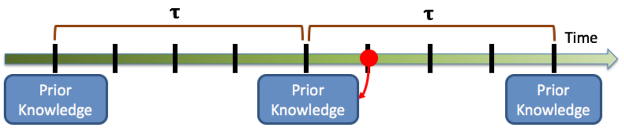


Figure 2: We display the overview of prior knowledge creation and usage in the temporal domain.

ally applies a filter to each spatial local region to calculate one value, such as an average value, to represent the region. Although this approach can effectively reduce the data size without complicated computation, only limited information is kept in the low-resolution proxy and it introduces error in the post analysis. For example, the minimum and maximum data values in some local data blocks of the density quantity can have a difference on the 10^{12} scale. The average value will introduce severe bias in the data analysis. Therefore, instead of using one value to represent the data in a spatial local block, we suggest a statistical approach to down-sample the dataset. The data in a local block is summarized by a distribution. Distributions can keep abundant information, such as the occurrence probabilities of values and statistical characteristics in the data block, to have the potential to better reconstruct data back to high resolution.

4.1 Gaussian Mixture Model

To compactly store a local data block and model the wide data value range in cosmological datasets, we adopt the Gaussian Mixture Model (GMM) to represent the distribution. GMM is a popular parametric distribution model which consists of multiple weighted Gaussian distributions. The unique statistical properties of Gaussian distribution allow GMM to model arbitrary distribution shape in a compact and flexible fashion. The proposed statistical down-sampling method uses GMM described in Equation 1 to model and store the samples in a high-resolution data block (h_p) corresponding to a low-resolution point (ℓ_p). Figure 3 illustrates an example of statistical down-sampling a volume of a quantity.

$$g_p(x) = \sum_{i=0}^{K} w_i * (x|\mu_i, \sigma_i) \tag{1}$$

where $g_p(x)$ is a probability density function which is represented by a GMM. K is the number of Gaussian components. w_i, μ_i and σ_i are the weight, mean and standard deviation of the i^{th} Gaussian component, respectively. The sum of the weights, $\sum_{i=0}^{K} w_i$, in a GMM must be 1. After statistical down-sampling, all local data blocks, from every quantity of interest and time step, are represented by independent GMMs.

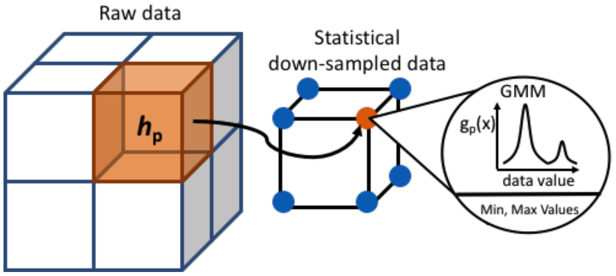


Figure 3: We illustrate the statistically down-sampling of a raw volume to 2^3 low-resolution grids. The orange block is a local data block. The orange point is a down-sampled point represented by a GMM which is computed from the orange block, and a minimum and a maximum value of the orange block.

4.2 *In situ* Workflow

To implement the statistical down-sampling for Nyx, we append an *in situ* call at the end of each time step. The *in situ* processing collects all data of all quantities of interest which reside in the same computing node to perform the statistical down-sampling. Our statistical down-sampling has to compute the GMM, minimum value, and maximum value for each local data block. Minimum and maximum values are used to filter out the values that are out of the value range when reconstructing the down-sampled data to high resolution. Expectation-Maximization (EM) algorithm is used to estimate the weights, means, and standard deviations of a GMM. The EM algorithm is an iterative method which adjusts the parameters of GMM to maximize the likelihood of input data values. To minimize the time for GMM modeling, we implement a data parallel EM algorithm based on the VTK-m framework [25]. The algorithm takes data from all data blocks of all quantities from one time step on one computing node to estimate the parameters of all GMMs in parallel, which can fully utilize the computational resource on the computing node. In addition, our system allows users to flexibly control the time of *in situ* call by sub-sampling partial data to run the EM algorithm.

5 Prior Knowledge

To reconstruct a down-sampled point back to a high-resolution data block, the data values can be obtained by re-sampling from the GMM. However, the data values lack the spatial location information to allocate them in space. Without predicting the spatial location for the re-sampled data values, the reconstruction quality will be poor. To address this, we propose prior knowledge to predict the location of the re-sampled data values from a GMM. The prior knowledge is computed from the raw data of a small number of simulation runs. We call these simulation runs *prior simulation runs*. The data generated by different parameter inputs have a correlation, and by creating the prior knowledge from the prior simulation runs, they are used to compensate the lack of the location information. The prior knowledge consists of two parts. One is the spatial information dictionary and the other one is a feature matching metric. The spatial information dictionary is a mapping between feature vectors to location information, and is used to assign each data value from a GMM to a position of high-resolution data. A feature vector is a descriptor of a given data block and computed from the statistical moments of the local neighboring data blocks. Figure 4 gives an overview of the spatial information dictionary. A feature matching metric defines a distance space to match a feature vector for a down-sampled point to a feature vector in the spatial information dictionary to access proper location information for high-resolution data reconstruction. This section introduces the details of the spatial information dictionary and the feature matching metric.

5.1 Spatial Information Dictionary

The spatial information dictionary maps feature vectors to spatial location information. Each entry in the dictionary is computed from a B^3 data block of the raw data from the prior simulation runs, where B is pre-defined down-sampled block size. A feature vector consists of the statistical moments from itself and those of the neighboring data blocks. We use the means and standard deviations from the data blocks to build the feature vector. This is illustrated in Figure 4. The feature vector is normalized by the minimum and maximum mean values in the vector to capture the relative trend of data values in neighboring regions. Equation 2 shows a feature vector of a data block, h_p.

$$f_{h_p}^{\Omega} = \frac{[\mu_{\Delta_0}\ \mu_{\Delta_1}\ \dots\ \mu_{\Delta_{n-1}}\ \sigma_{\Delta_0}\ \sigma_{\Delta_1}\ \dots\ \sigma_{\Delta_{n-1}}]}{Max([\mu_{\Delta_0}\ \dots\ \mu_{\Delta_{n-1}}]) - Min([\mu_{\Delta_0}\ \dots\ \mu_{\Delta_{n-1}}])} \tag{2}$$

where $f^{\Omega}_{h_p}$ is the feature vector of data block h_p. $\Omega = \{\Delta_0, \Delta_1, ..., \Delta_{n-1}\}$ defines n functions to locate the neighboring data blocks, which also includes h_p itself. The numerator is a vector, and μ_{Δ_i} and σ_{Δ_i} are the mean and standard deviation calculated from data block $h_{\Delta_i(p)}$. If a high resolution data block $h_{\Delta_i(p)}$ is out of the global spatial domain of the dataset, we pad the outside region to zero to calculate the mean and standard deviation. $Max([\mu_0 \; ... \; \mu_{n-1}])$ and $Min([\mu_0 \; ... \; \mu_{n-1}])$ are the maximum and minimum mean values, respectively. Since the neighbor locality function Ω is predefined, we use f_{h_p}, ignoring Ω, to indicate the feature vector of h_p in the rest of this paper.

Due to the spatial, temporal and parameter correlation between simulation runs, if the distributions of the data values in neighboring data blocks are similar, the locations for data values in the center data block are likely to be similar, too. For example, the locations of the largest values are similar in the local block. When reconstructing a down-sampled point back to high resolution, the re-sampled data will be assigned to space by using the location distribution of data values in the data block, whose feature vector is most similar to the feature vector of the down-sampled point. For example, the largest/smallest value among the re-sampled data will be assigned to the location which has the largest/smallest value in the data block. The detail of computing the feature vector from the down-sampled data is introduced in Section 6.

The distribution of data sample locations for each data block in the prior simulation runs is retrieved from the data block then stored as the location information in the dictionary. The location information (s_{h_p}) in the dictionary is represented by a sequence of location indexes. The sequence is sorted according to the corresponding data values and only the location indexes are stored in the dictionary without the data values. We convert the 3D location index to 1D, and this usually requires less than 2 bytes to save a location index in a local data block. Essentially, the stored location information is a function which maps the samples' order defined by samples' value among a group of samples to location indexes in the high-resolution domain. By giving a group of re-sampled data values and calculating the order of data values, s_{h_p}, we can map data values to locations in high-resolution space. Figure 4 illustrates the spatial information dictionary creation.

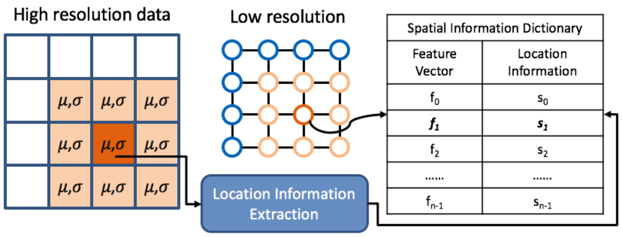


Figure 4: We illustrate a 2D example of creating an item, f_1 and s_1 in the spatial information dictionary from the dark orange point in low-resolution space. The orange points are the neighboring points, and the orange blocks in high-resolution data are the corresponding neighboring data blocks. The mean and standard deviations are directly computed from the data in these data blocks to generate f_1. The location information s_1 is extracted from the dark orange data block.

5.2 Feature Matching Metric

To recover low-resolution GMM point to high resolution, we need to find a matching feature vector in the dictionary to get the location information. This requires a feature matching metric to define a distance space between feature vectors of the point in the dictionary such that corresponding location information of a feature in the dictionary is more likely to better reconstruct a down-sampled grid point to a high resolution. The distance metric, $D(f_{h_i}, f_{h_j})$, is constructed from prior simulation runs to create a distance space which separates feature vectors if the location information retrieved from the corresponding data blocks of a feature vector does not well reconstruct the disorder data samples collected from the other one. For example, U_{h_i} and U_{h_j} are data samples' collected from data blocks h_i and h_j. Locations of data samples are unknown in U_{h_i} and U_{h_j}. $r^{h_j}_{h_i}$ is a reconstructed block by combining samples from U_{h_i} and the location information (s_{h_j}) which is retrieved from h_j and $r^{h_i}_{h_j}$ is a reconstructed block by combining samples from U_{h_j} and the location information (s_{h_i}) which is retrieved from h_i. We call the reconstruction error of $r^{h_j}_{h_i}$ and $r^{h_i}_{h_j}$ the cross reconstruction error of h_i and h_j. $D(f_{h_i}, f_{h_j})$ should approximate the cross reconstruction error of h_i and h_j. The cross error between h_i and h_j is shown in Equation 3.

$$CE(h_i, h_j) = \frac{NR(h_i, r^{h_j}_{h_i}) + NR(h_j, r^{h_i}_{h_j})}{2} \tag{3}$$

where $NR(h, r)$ is the function to compute normalized RMSE between the raw and reconstructed data. The detail is shown in Equation 4.

$$NR(h, r) = \frac{\sqrt{E^2(h, r)}}{Max(h) - Min(h)} \tag{4}$$

where h is a true data block and r is a reconstructed data block in high resolution. $Max(h)$ and $Min(h)$ calculate the maximum and minimum values in h. $E^2(h, r)$ calculates the average of the square differences of all data values at the same location. The fundamental form of distance between two feature vectors in our system is Mahalanobis distance [23] and Equation 5 shows the Mahalanobis distance.

$$D(f_{h_i}, f_{h_j}) = \sqrt{(f_{h_i} - f_{h_j}) * S^{-1} * (f_{h_i} - f_{h_j})} \tag{5}$$

where f_{h_i} and f_{h_j} are two feature vectors, and S is a covariance matrix. The distance between two feature vectors defined by Equation 5 should approximate the cross reconstruction error of the corresponding data blocks. This approximation is achieved by finding a proper covariance matrix S which transforms the feature distance space to approximate the cross reconstruction error space. This transformation is built from data blocks collected from prior simulation runs. Equation 6 shows the cost function to be optimized for the covariance matrix calculation.

$$\arg\min_{S} = \sum_{i=0}^{N-1} \sum_{j=i}^{N-1} (D(f_{h_i}, f_{h_j}) - CE(h_i, h_j))^2 \tag{6}$$

where N is the total number of data blocks collected from data of prior simulation runs. The covariance matrix is stored and used to reconstruct high-resolution data.

5.3 Data Collection for Prior Knowledge Creation

The prior knowledge is created from the prior simulation runs. To collect data which has good coverage and variation in the parameter space, we use Latin hypercube sampling [9] to sample the parameters and run the simulation to create the prior knowledge. Latin hypercube sampling is often used by domain experts to sample simulation parameters to run a few simulation runs for data analysis, when they cannot output and store results from many simulation runs. The Latin hypercube sampling is similar to the problem of having n rooks (towers) on a high-dimensional chess board without threatening each other. Figure 5 gives a 2D example.

When generating data for the prior knowledge creation, the simulation outputs the data at every τ timesteps instead of every time step. Due to data coherence in the temporal domain, the prior knowledge created at a time step can be used to reconstruct data at neighboring time steps. This also reduces the I/O of the prior simulation runs, prior knowledge size, and creation time. To create the prior knowledge, every raw volume of any quantity of interest and any output time step is subdivided into several spatial sub-blocks, called prior knowledge blocks, and each one creates independent prior knowledge. Due to the data coherence in the local spatial domain, mixing data from the whole spatial domain to create a single prior knowledge is not desirable and could lower the quality of the prior knowledge. Note that a prior knowledge block is different from a local data block (h_p) which is represented by a GMM. The whole data spatial domain contains multiple prior knowledge blocks and each prior knowledge block contains multiple local data blocks. For example, if the data spatial domain is divided into L prior knowledge blocks, a domain expert has Q quantities of interest, and T' is the output time steps, our system will create $(L * Q * T')$ independent prior knowledges. Figure 5 shows the prior knowledge creation workflow.

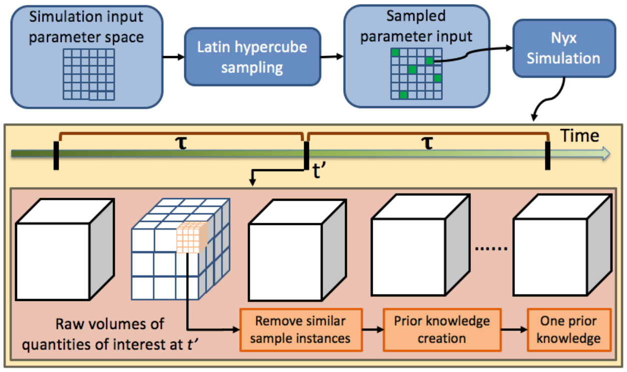


Figure 5: We illustrate prior knowledge creation workflow. The chess board at the left upper corner is a 2D example of the simulation parameter input space. The green squares are the result of the Latin hypercube sampling. We illustrate one sampled parameter, which is the input to Nyx, and one time step (t') output, for prior knowledge creation here. The white cubes at the bottom are the output volumes of quantities of interest at time step t'. The blue divisions illustrate the defined prior knowledge blocks. The orange block illustrates using a sliding block to extract sample instances creating one prior knowledge block.

To collect the data for prior knowledge creation, we use a "sliding block" with size B^3 to collect sample instances from the prior simulation runs, where B^3 is the predefined down-sampled block size. The sample instances are examined to only keep one instance to represent multiple similar sample instances across multiple prior simulation runs. The similarity is defined by the cross-error of Equation 3. This preprocess simplifies the prior knowledge creation and makes the data reconstruction process in Section 6 more efficient, because it shrinks the size of spatial information dictionary.

6 Statistical Super-resolution

After creating the prior knowledge from a few simulation runs, remaining simulation runs in the parameter space can be performed with *in situ* statistical down-sampling and later reconstructed with prior knowledge. Only the compact statistical down-sampled data are saved to disk. This does not only reduce the I/O time when writing data from supercomputers and loading data to the post-analysis machine, but also can keep output of more simulation runs on disk for later cosmological data analysis. This section will introduce reconstructing the statistical down-sampled data to high resolution. The step of up-sampling the low-resolution data to high resolution consists of (1) re-sampling data values from GMMs (2) creating the feature vectors from statistical down-sampled data and finding the best matching location information in prior knowledge, and (3) using the re-sampled data and the location information to reconstruct the high resolution data.

6.1 GMM Data Re-sampling

Each point in low resolution is represented by a GMM, and the GMM is computed from the data values within a B^3 data block in high resolution. The data values in a high-resolution data block can be reconstructed by drawing B^3 data values from the GMM. The procedure to draw a sample from a GMM has two steps. The first step is to select a Gaussian component from the GMM according to the weight associated with each Gaussian component. The second step is to draw a value from the selected Gaussian distribution. By repeating the sampling step B^3 times, the data values in the high-resolution block are reconstructed. In addition, the domain of a Gaussian distribution is defined in an infinite value space, so unnecessary values, though associated with extremely small probability in the Gaussian distribution, still have chances to be drawn. To avoid drawing these unnecessary values, we only accept values that are within 3 standard deviations of the mean, and between the stored minimum and maximum values of the data block. Otherwise, the sample is rejected and re-drawn from the Gaussian distribution. These re-sampling values do not have the location information to allocate them in the high-resolution spatial space yet.

6.2 Location Information Query

To query location information for the reconstruction of a low-resolution point, the first step is to compute the feature vector of the point. In Section 5.1, we introduced that the feature vector of a high-resolution data block consists of means and standard deviations of itself and those of the neighboring data blocks. In contrast to the spatial information dictionary creation in Section 5.1, the high-resolution data blocks are not available here. Therefore, we have to derive means and standard deviations from GMMs. Equation 7 and 8 show the mean and standard deviation calculation of a GMM.

$$\mu_{g_p} = \sum_{k=0}^{K-1} (w_i * \mu_i) \tag{7}$$

$$\sigma_{g_p} = \sqrt{\sum_{k=0}^{K-1} (w_i * \mu_i^2) + \sum_{k=0}^{K-1} (w_i * \sigma_i^2) - \mu_{g_p}^2} \tag{8}$$

where μ_{g_p} and σ_{g_p} are the mean and standard deviation of the GMM, g_p, at a low resolution point p. K is the number of Gaussian components of g_p. w_i, μ_i and σ_i are the i^{th} weight, mean and covariance matrix of g_p, respectively. After computing the mean and standard deviation of a GMM, the feature vector of a point can be computed by Equation 2. If the feature vector computation requests the mean and standard deviation from a point that is out of the data spatial domain, the mean and standard deviation are set to 0.

To query the location information for reconstruction of a low-resolution point with quantity q at time step t, we first look up the prior knowledge whose block spatial domain contains the point, and time step is the closest to t and quantity is q. The feature vector f_p computed from the point p is used to query the location information s' which is associated with the feature vector f' in the dictionary. f' is the feature vector which is closest to f_p in the spatial location dictionary. The distance between feature vectors was introduced in Section 5.2 and the covariance matrix S learned from data are involved to find the closest feature vector in the spatial location dictionary. Equation 9 shows how to find the feature vector f'.

$$f' = \arg\min_{f} \sqrt{(f - f_p) * S^{-1} * (f - f_p)} \tag{9}$$

After sampled values and location information (s') are known, they are used to reconstruct the high-resolution data block.

6.3 High Resolution Data Reconstruction

Reconstruction uses the location information function (s'), which maps samples' order defined by samples' value among a group of samples to location indexes, to assign the samples to the spatial domain. We first sort the samples drawn from GMM according to the data values to get an order of samples. Then using the location information (s'), we map each data sample to the spatial domain. Algorithm 1 summarizes the steps to reconstruct high-resolution data from a low-resolution grid point. Line 6 in Algorithm 1 allocates a buffer with size B^3 for high-resolution data reconstruction. Line 7 selects the prior knowledge according to the time step, quantity and spatial coordinate of the reconstructing grid point. The details of Line 7 are in Section 6.2. Line 8 and Line 9 generate the feature vector for p and find location information from the prior knowledge. Line 10 retrieves the GMM at location p. In Line 11, u_p is the collection of samples which are re-sampled from g_p. Line 12 sorts data in u_p and returns the sorted array to u'_p. The loop between Line 13 and 17 goes through the re-sampled values in order, and places data into the high-resolution data block buffer following the location information. Since the reconstruction of each low-resolution grid point is independent, we have a portable parallel statistical super-resolution implementation based on the VTK-m framework [25]. This implementation can increase the speed of super-resolution on either GPU (CUDA) or multicore CPU (Intel TBB) backend according to the available resource on the post-analysis machine by changing the compiler configuration, and without changing the code.

Algorithm 1 Statistical Super-resolution

```
 1: ▷ $L_{t,q}$: a statistical down-sampled data at time $t$ with quantity $q$
 2: ▷ $p$: the low resolution coordinate of a point for reconstruction
 3: ▷ $PK$: collection of all prior knowledge
 4: ▷ $B$: pre-defined down-sampled data block size
 5: procedure StatisticalSR($L_{t,q}$, $p$, $PK$, $B$)
 6:     $h_p$ = AllocateBuffer($B^3$)
 7:     $pk$ = PriorKnowledgeSelector($PK$, $t$, $q$, $p$)
 8:     $f_p$ = FeatureVectorGenerator( $L_{t,q}$, $p$ )
 9:     $s'$ = LocationInformationQuery($pk$, $f_p$)
10:     $g_p = L_{t,q}(p)$
11:     $u_p$ = GmmResampler($g_p$, $B^3$)
12:     $u'_p$ = Sorting($u_p$)
13:     for $i = 1 \to B^3$ do
14:         $\ell = s'(i)$
15:         $v = u'_p[i]$
16:         SetValueToLocation($h_p$, $v$, $\ell$)
17:     end for
18:     return $h_p$
19: end procedure
```

7 Evaluation

To carry out the evaluation, the simulation parameter space consists of three parameters of interest: hubble constant (*comoving_h*), the total density of baryons (*comoving_OmB*), and the total matter density (*comoving_OmM*). Due to the cosmologists' efforts in the past decades, they have shrunk the possible value ranges of these parameters to smaller intervals. The domain ranges of interest are *0.55* $\leq$ *comoving_h* $\leq$ *0.85*, *0.0215* $\leq$ *comoving_OmB* $\leq$ *0.0235* and *0.12* $\leq$ *comoving_OmM* $\leq$ *0.155*. 10 samples are regularly taken in each dimension of interest across the simulation input parameter space which results in 1000 possible simulation inputs. Each simulation run can have a different number of time steps according to the convergence speed of each simulation input. In this experiment, we use the first 200 time steps from each simulation run to carry out this evaluation, because all simulation runs have at least 200 time steps.

We evaluate our technique on 7 different output quantities: density of dark matter particles (*density*), X-momentum (*xmom*), Y-momentum (*ymom*), Z-momentum (*zmom*), internal energy of the gas (*rho_e*), temperature (*Temp*) and gravitational potential (*phi_grav*). Each output data is a regular grid with a resolution of 256^3 grid points. The prior knowledge is created from only 5 simulation runs, which is sampled from the simulation parameter input space by latin hypercube sampling. In addition, we create the prior knowledge at every 10 time steps and set the prior knowledge block size to 64^3. For the feature vector creation, the surrounding grid points (Ω in Equation 2) of a given grid point at (i, j, k) are 7 points which includes (i, j, k), $(i+1, j, k)$, $(i-1, j, k)$, $(i, j+1, k)$, $(i, j-1, k)$, $(i, j, k+1)$ and $(i, j, k-1)$. The *in situ* statistical down-sampling for all other simulations only uses 25% of grid points, in each 16^3 data block, to compute a GMM with 5 Gaussian components. Each simulation run is carried out on a node of supercomputer with two 14-core Intel (Broadwell) Xeon E5-2680 v4 processors and 128 GB DDR3 Memory.

7.1 Quantitative Evaluation

7.1.1 Storage

Our technique outputs the prior knowledge and the statistical down-sampled data for all output quantities, time step and simulation runs. The ratio of the statistical down-sampled data size to the raw data size can be computed by Equation 10.

$$Z_{SDS} = \frac{G * 3 + 2}{B^3} \tag{10}$$

where $G * 3$ means each GMM has G Gaussian components. Each Gaussian component is represented by a weight, a mean and a standard deviation. 2 represents the stored minimum and maximum values of a down-sampled data block. B is the down-sampled data block resolution. In our experiment, we use a GMM with 5 Gaussian components to represent a 16^3 data block. The storage consumption of the statistical down-sampled data is only 0.42% of the raw data. Our prior knowledge for all 7 quantities is generated at every tenth times step. The total size of the prior knowledge is 25GB. If we store raw data for 1000 simulation runs with 200 time steps and 7 quantities, it requires 85 TB storage. In contrast, our approach only consumes 388 GB (25 GB for prior knowledge and 363 GB for all statistical down-sampled data), which is a total of 0.44% of the raw data.

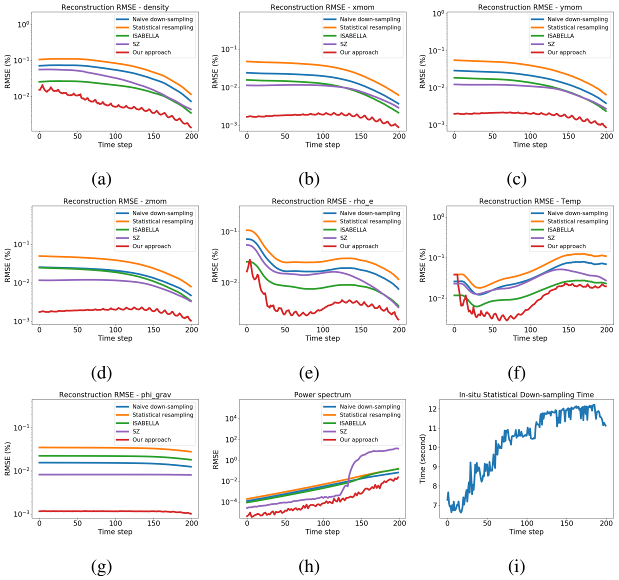
7.1.2 Error of Reconstruction

We compare the reconstruction error with 4 other different approaches. The first is a naive down-sampling, which uses the average value to represent a local data block and reconstructs data by interpolation. The second is statistical down-sampling, without the prior knowledge. It reconstructs data by re-sampling data from a GMM, and randomly placing samples to a data block. The third and fourth approaches are scientific data lossy compression techniques, "ISABELA" [17] and "SZ" [8].

To achieve a fair comparison on reconstruction error, we adjust the parameters of these four approaches to generate data proxies whose size are as close as possible to our approach. Comparing with raw data size, our approach outputs 0.44%. For naive down-sampling approach, each 6^3 data block is represented by one average value and this consumes 0.46% of the raw size. The statistical down-sampling without prior knowledge also down-samples a 16^3 data block to a GMM and consumes 0.42% of the raw size. We adjust the error tolerance and the number of stored coefficients to minimize the size from ISABELA. The smallest size of compressed data we can achieve is approximately 24% of the raw data, around 55 times more than our approach. For SZ compression, we adjust the PSNR error bound to 12 dB to generate a compressed data with an average around 2.44% of the raw size.

We randomly draw 250 simulation parameter inputs to run the simulation and compare the error of the ground truth to the reconstructed data in each case. The RMSE of one reconstructed volume is calculated by Equation 4, where *h* and *r* are the whole raw volume and reconstructed volume, respectively. $Max(h)$ and $Min(h)$ are the maximum and minimum values of the raw volume. Figure 6 shows the average Root Mean Square Error (RMSE) of the reconstructed data among 250 simulation runs. Since we create the prior knowledge at every 10 times steps and use the closest prior knowledge in the temporal domain to reconstruct data, the error curves for our approach have periodic bumps. Our approach has the best reconstruction RSME, except for several time steps of the *Temp* quantity. By examining the data, we found the simulated universe initially has a uniform temperature over the whole space. Therefore, the denominator of Equation 4 is extremely small, greatly amplifying errors due to the re-sampling process from GMM. In general, our approach shows lower reconstruction error than all other approaches for all 7 quantities.

We used a domain tool, Gimlet [13], to compute the power spectrum from the raw data and reconstructed data. The power spectrum of the density fluctuations is a frequently used cosmological statistical measure. The error of the power spectrum computed from raw data and reconstructed data is plotted in Figure 6(h). The error from our technique is lower than other alternatives.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 6: (a) - (g) are average reconstruction RMSE of 7 quantities over 250 simulation runs. (h) is the average error of the power spectrum. (i) is the average *in situ* down-sampling time. X-axis is the time steps. Y-axis in (a) - (h) is the RMSE in log scale.

7.1.3 Performance

The computation time of the *in situ* statistical down-sampling consists of two parts, time for the EM algorithm to compute GMMs and time for writing down-sampled data to disk. The average time to write the down-sampled data for one time step to disk is 0.8 seconds among all time steps and simulation runs. In contrast, the time to write full resolution raw data to disk is 11.2 seconds. The statistical down-sampling runs the EM algorithm to compute GMMs and the average time to compute GMMs for one step is 9.4 seconds. The time to compute GMMs at different time steps is not constant. Because the data values usually have higher variation in later time steps in the Nyx simulation, the EM algorithm requires more time to converge for the later time steps. Figure 6(i) shows the time of *in situ* down-sampling over time steps. The average of the *in situ* statistical down-sampling time per simulated time step is 10.19 seconds. Therefore, our approach significantly reduces the stored data size without spending more supercomputer time and reduces the data movement cost to post-analysis computers.

In this experiment, we created 64 * 7 * 21 prior knowledge data sets, because we have 64 prior knowledge blocks per volume, 7 quantities, and 21 time steps for prior knowledge creation. We subdivided the prior knowledge creation tasks into 140 non-overlapping sets, where each set takes 1.37 hours on average. Note that the prior knowledge creation is a one-time task from the 5 simulation runs. Once the prior knowledge is generated, it can be used for reconstructing down-sampled data of all subsequent simulation runs. The wall time can be reduced by subdividing the tasks into more sets if more computational resources are available. The time to reconstruct a statistical down-sampled volume to full resolution is 1.28 seconds, using an Intel 8-Core i-7-4770 3.40GHz CPU with 16GB memory. Each local data block reconstruction is an independent task, and the time can be further reduced if more computational resources are available.

7.2 Qualitative Evaluation

We use direct volume rendering, isosurfaces, streamlines, and Gimlet [13] to show the quality of reconstruction. Figure 7 shows rendering images of the reconstructed data for *density* using one selected time step and input parameters. Our approach shows the most similar image compared to the raw data.

Figure 8 shows an isosurface from the reconstructed data of *phi_grav* quantity. Our approach most accurately preserves the shape of the isosurface. We circle several regions where other alternatives show incorrect shapes, such as the pieces at the left upper corner, the connection at the bottom left corner, and the hold at the middle.

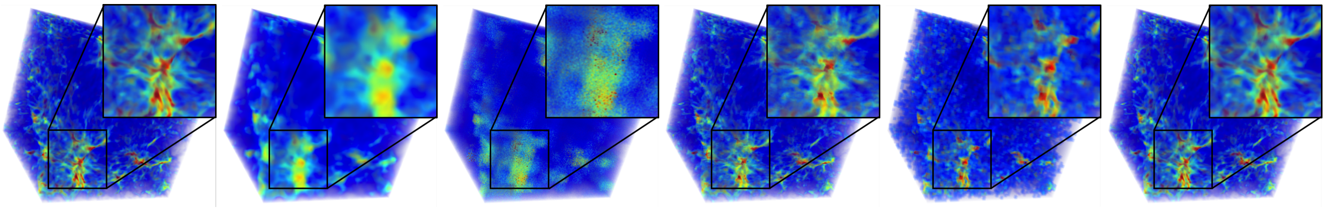
Figure 9 shows streamlines using the vector field from *xmom*, *ymom*, and *zmom* quantities. We highlight and zoom-in two regions, a vortex and a swirl. Our approach reproduces the left vortex. Naive down-sampling also reproduces it, but the vortex shifts in space. All methods cannot reproduce the right swirl, but our approach approximates this feature the best.

Figure 10 shows the power spectrum at time step 200 for two simulation runs. We observe that the power spectrum computed from our approach is the most similar to the raw data. A low error power spectrum is also computed from reconstructed data using ISABELA, but ISABELA requires 55 times storage than our approach.

8 Discussion

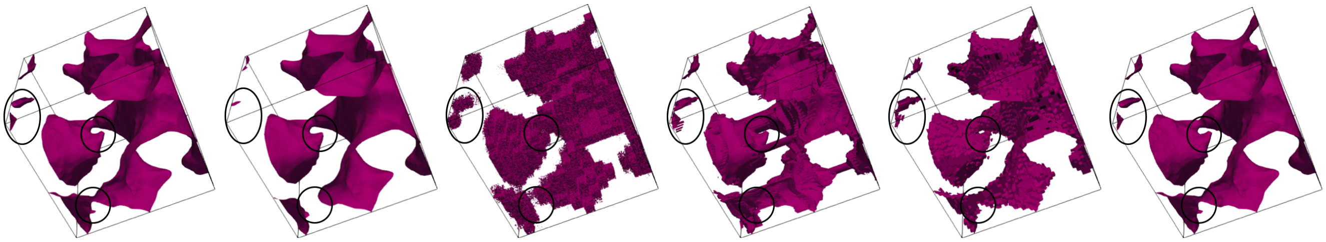
8.1 Parameter Turning

In our system, several parameters have to be selected. They are the number of Gaussian components of each GMM, the number of prior simulation runs, the size of a down-sampled block, and the number of samples in each local data block for the EM algorithm. The size of a down-sampled block is mainly determined by the affordable size of statistical down-sampled data. The selection of the number of samples in each local data block for the EM algorithm determines *in situ* time. A very small number simulation runs with sparse *in situ* calls in the temporal domain can be used to determine the number of sub-sampling data samples to run the EM algorithm. The number



(a) Raw data (100%) (b) Naive D.S. (0.46%) (c) Statistical D.S. (0.42%) (d) ISABELA (24%) (e) SZ (2.44%) (f) Our approach (0.44%)

Figure 7: We display volume rendering of *density* quantity at time step 180 from a simulation run with input parameters, *comoving_h = 0.61666*, *comoving_OmB = 0.02216* and *comoving_OmM = 0.14328*. D.S. stands for down-sampling. The numbers in sub-figure captions are the ratio of the storage consumption to the raw data size.



(a) Raw data (100%) (b) Naive D.S. (0.46%) (c) Statistical D.S. (0.42%) (d) ISABELA (24%) (e) SZ (2.44%) (f) Our approach (0.44%)

Figure 8: We display isosurface of *phi_grav*=0 at time step 153 from a simulation run with input parameters, *comoving_h = 0.55*, *comoving_OmB = 0.02326* and *comoving_OmM = 0.1394*. D.S. stands for down-sampling. The numbers in sub-figure captions are the ratio of the storage consumption to the raw data size.

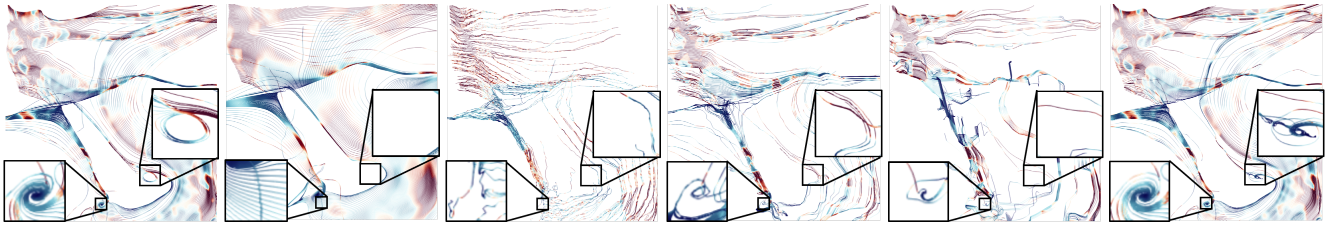
of prior simulation runs impacts the reconstruction quality. To study the reconstruction quality, we compared the reconstruction results when parameters are changed. Keeping all high-resolution data for this study is not feasible. Therefore, data generated from fewer qualities, and smaller spatial and temporal resolution are used to study and select the parameters. We have run this study by 64^3 resolution, 3 quantities (*"density"*, *"zmom"* and *"Temp"*), 4 time steps (50, 100, 150, 200), and a down-sampled block size 8^3. The raw data of all 1000 simulation runs, defined in the simulation parameter space in Section 7, are saved for this study using 12 GB. We use Latin hypercube sampling to sample 1, 3, 5, 7, 9 simulation runs for prior knowledge creation. Figure 11(a) shows the normalized RMSE. When only one simulation run is used to create the prior knowledge, the reconstruction error is high. The error decreases when the number of simulation runs for prior knowledge creation increases. We see diminishing returns when the number of simulation runs is more than 5. In addition, we also varied the number of Gaussian components to study the impact on reconstruction quality. In Figure 11(b), we observe similar diminishing returns if the number of Gaussian components is greater than 4. Figure 11(c)-(f) show the quality of reconstructed data with fewer Gaussian components and prior simulation runs. With fewer prior simulation runs, the GMM samples are more likely to be assigned incorrect locations. If fewer Gaussian components are used, the GMM may not accurately summarize the data, and a large error may be introduced to re-sampled data.

8.2 Data Variation in Parameter Space of Nyx

We will discuss the variation produced from Nyx simulation. Density, at time step 200, is used as an example in this discussion. We examine all simulation runs in Section 7 and find the maximum density value, $3.8*10^{13}$, and the minimum value in the same simulation run is $4.8*10^8$. For the smallest valued simulation, the maximum density value is $1.2*10^{11}$. So, Nyx can produce datasets where the maximum value of a simulation run is 300 times larger than that of another. In addition, the standard deviation of the maximal values among simulation runs is $1.7*10^{13}$. We show how data values change among different simulation runs in Figure 12(a). We compared values along raycast segments from simulation runs as an example. The ray is cast from the center of the x-y plane with z=0 to the center of the x-y plane with z=255 on the density quantity of 5 prior simulation runs in Section 7 and 20 randomly sampled test simulation runs to collect data values on the ray and plot. It shows that data values from different simulation runs have large differences and the trends of data value change among simulation runs are also different. For example, the curves circled by the purple ellipse have a concave in the region, but the curves circled by the red and green ellipses are peaks. On the other hand, we also observe that multiple curves may have similar trends of data value change in a local region, but these curves still have quite different data values. In our approach, GMM accurately captures the data values without keeping location information. The trends of value change from the prior knowledge, the colored solid lines in Figure 12(a), is used to place values into space. We show the isosurfaces from a prior simulation run and a test simulation run, which are marked in Figure 12(a). Although they have relatively close curves in Figure 12(a), the isosurfaces in Figure 12 (b) and (c) are significantly different.

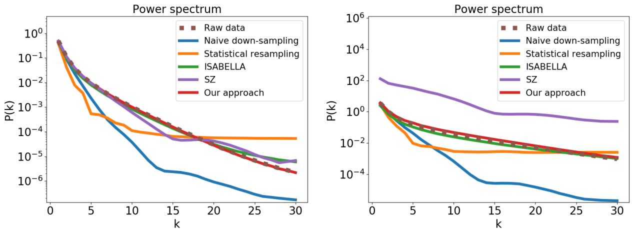
8.3 Domain Expert Feedback

We have interviewed a cosmologist, who is one of the authors of Nyx simulation, and summarized his comments. We discuss the applicability of our technique in their data analysis workflow. Using the setting in Section 7, it requires approximately 2 hours to finish one simulation. As we have noted before, it is not feasible to retain data from the high-resolution simulations permanently, due to the size. Re-running the simulation every time cosmologists need a data will require waiting hours to continue their analysis tasks. Our technique does not significantly reduce the overall simulation time, but it provides a high data reduction rate and reconstruction quality to retain data. Thus, the cosmologists are able to keep simulation runs and access data without re-running the simulation. The domain expert also commented, "These simulations are often very expensive,



(a) Raw data (100%) (b) Naive D.S. (0.46%) (c) Statistical D.S. (0.42%) (d) ISABELA (24%) (e) SZ (2.44%) (f) Our approach (0.44%)

Figure 9: We display streamlines of the vector field from *xmom*, *ymom* and *zmom* quantities at time step 200 with input parameters, *comoving_h = 0.58333*, *comoving_OmB = 0.02304*, and *comoving_OmM = 0.12776*. 250 seeds are put on the line between [0,0,0] and [255,255,255] to compute streamlines. We zoom-in to sub-domains of the data to show the details. The red and blue color on the streamlines indicate the higher and lower vector magnitudes, respectively. D.S. stands for down-sampling. The percentage in the sub-figure captions are the ratio of the storage consumption to the raw data size.



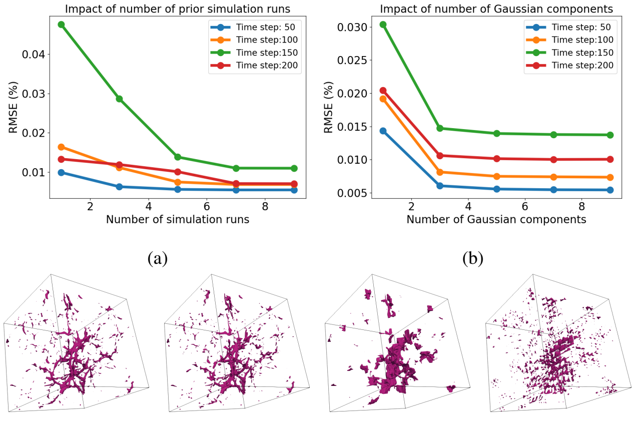
(a) (b)

Figure 10: (a) and (b) are power spectra computed from data at time step 200 with input parameters, *comoving_h = 0.84997*, *comoving_OmB = 0.02304* and *comoving_OmM = 0.13552*, and *comoving_h = 0.61666*, *comoving_OmB = 0.02216* and *comoving_OmM = 0.14328*, respectively.

so you don't want to run and re-run them every time you want to do some analysis". The cosmologist commented that the visual quality looks good and it's good that our method has the lowest RMSE.

9 Conclusion and Future Work

This paper presents a novel *in situ* technique for cosmological data analysis and visualization. The data from a few simulation runs are selected by Latin hypercube sampling and used to create prior knowledge which captures the relation between low- and high-resolution. Data from simulation runs with other simulation parameter inputs are down-sampled *in situ* to reduce the requirements of I/O bandwidth and disk storage. We implement data parallel statistical down-sampling and data sub-sampling to remain the simulation time of new workflow at the same scale of the traditional workflow. Because our approach hugely reduces the data from the cosmological simulation, the data from more simulation runs is affordable to save for data analysis. We qualitatively and quantitatively demonstrate that our technique outperforms other alternative approaches. In future work, we will explore using error quantification and machine learning technique. To quantify the point-wise error and convey that to scientists are valuable because knowing the error range in the visualization could change scientists' decision and avoid that the data analysis task is misled by the region with a higher error. Machine learning techniques, such as neural network based approaches, could improve the prior knowledge to capture more precise feature shapes and allow our technique to be used on more diverse types of simulations.



(a) (b)

(c) (d) (e) (f)

Figure 11: We show the impact of the reconstruction quality using different numbers of simulation runs for prior knowledge creation and a different number of Gaussian components. Every point on each curve is the average RMSE of quantities, *"density"*, *"zmom"* and *"Temp"*. (c) is an isosurface from raw data. (d), (e) and (f) are isosurfaces from the reconstructed data with 5 prior simulation runs and 5 Gaussian components, 5 prior simulation runs and 1 Gaussian component, and 1 prior simulation run and 5 Gaussian components, respectively.

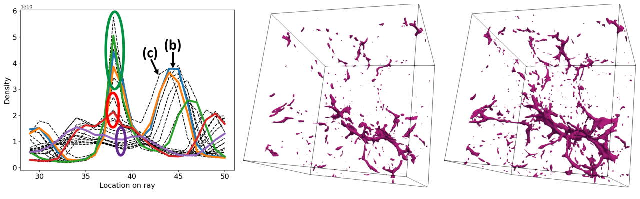
Acknowledgments

This work was supported in part by UT-Battelle LLC 4000159557, Los Alamos National Laboratory Contract 471415, NSF grant SBE-1738502 and Department of Energy Office of Science Advanced Scientific Computing Research, Program Manager Dr. Laura Biven. We also thank the domain scientists, Zarija Lukić, from Lawrence Berkeley National Laboratory to provide the insight of Nyx simulation and the feedback of our work.

References

[1] J. Ahrens, S. Jourdain, P. O'Leary, J. Patchett, D. H. Rogers, and M. Petersen. An image-based approach to extreme scale in situ visualization and analysis. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 424–434. IEEE Press, 2014.

[2] J. Ahrens, J. Patchett, A. Bauer, S. Jourdain, D. H. Rogers, M. Petersen, B. Boeckel, P. OLeary, P. Fasel, and F. Samsel. In situ mpas-ocean image-based visualization. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Visualization & Data Analytics Showcase*, 2014.



(a) (b) (c)

Figure 12: (a) Data values on raycast segments from 5 prior simulation runs and 20 randomly sampled test simulation runs. Data values on rays from prior simulation runs and sampled test simulation runs are plotted by colored solid lines and black dashed lines, respectively. Two simulation runs, a prior simulation run marked by (b) and a test simulation run marked by (c), show isosurfaces with isovalue 10^{10} in (b) and (c), respectively.

[3] A. S. Almgren, J. B. Bell, M. J. Lijewski, Z. Lukić, and E. Van Andel. Nyx: A massively parallel amr code for computational cosmology. *The Astrophysical Journal*, 765(1):39, 2013.

[4] J. C. Bennett, H. Abbasi, P.-T. Bremer, R. Grout, A. Gyulassy, T. Jin, S. Klasky, H. Kolla, M. Parashar, V. Pascucci, et al. Combining in-situ and in-transit processing to enable extreme-scale scientific analysis. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, p. 49. IEEE Computer Society Press, 2012.

[5] A. P. D. Binotto, J. L. Comba, and C. M. Freitas. Real-time volume rendering of time-varying data using a fragment-shader compression approach. In *Parallel and Large-Data Visualization and Graphics, 2003. PVG 2003. IEEE Symposium on*, pp. 69–75. IEEE, 2003.

[6] C.-M. Chen, A. Biswas, and H.-W. Shen. Uncertainty modeling and error reduction for pathline computation in time-varying flow fields. In *2015 IEEE Pacific Visualization Symposium (PacificVis)*, pp. 215–222. IEEE, 2015.

[7] M. Davis, G. Efstathiou, C. S. Frenk, and S. D. White. The evolution of large-scale structure in a universe dominated by cold dark matter. *The Astrophysical Journal*, 292:371–394, 1985.

[8] S. Di and F. Cappello. Fast error-bounded lossy hpc data compression with sz. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 730–739. IEEE, 2016.

[9] D. Donovan, K. Burrage, P. Burrage, T. McCourt, B. Thompson, and E. Yazici. Estimates of the coverage of parameter space by latin hypercube and orthogonal array-based sampling. *Applied Mathematical Modelling*, 57:553–564, 2018.

[10] 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*, 23(1):811–820, 2017.

[11] S. Dutta, H.-W. Shen, and J.-P. Chen. In situ prediction driven feature analysis in jet engine simulations. In *Pacific Visualization Symposium (PacificVis), 2018 IEEE*, pp. 66–75. IEEE, 2018.

[12] S. Dutta, J. Woodring, H.-W. Shen, J.-P. Chen, and J. Ahrens. Homogeneity guided probabilistic data summaries for analysis and visualization of large-scale data sets. In *Pacific Visualization Symposium (PacificVis), 2017 IEEE*, pp. 111–120. IEEE, 2017.

[13] B. Friesen, A. Almgren, Z. Lukić, G. Weber, D. Morozov, V. Beckner, and M. Day. In situ and in-transit analysis of cosmological simulations. *Computational Astrophysics and Cosmology*, 3(1):4, 2016.

[14] J. Gao, H.-W. Shen, J. Huang, and J. A. Kohl. Visibility culling for time-varying volume rendering using temporal occlusion coherence. In *Proceedings of the conference on Visualization'04*, pp. 147–154. IEEE Computer Society, 2004.

[15] S. Guthe and W. Strasser. Advanced techniques for high-quality multiresolution volume rendering. *Computers & Graphics*, 28(1):51–58, 2004.

[16] 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*, 2018.

[17] S. Lakshminarasimhan, N. Shah, S. Ethier, S.-H. Ku, C.-S. Chang, S. Klasky, R. Latham, R. Ross, and N. F. Samatova. Isabela for effective in situ compression of scientific data. *Concurrency and Computation: Practice and Experience*, 25(4):524–540, 2013.

[18] J. Li, J. Wu, S. Yang, and J. Liu. Dictionary learning for image superresolution. In *Control Conference (CCC), 2014 33rd Chinese*, pp. 7195–7199. IEEE, 2014.

[19] 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, pp. 422–447. Wiley Online Library, 2018.

[20] S. Liu, J. A. Levine, P. Bremer, and V. Pascucci. Gaussian mixture model based volume visualization. In *Large Data Analysis and Visualization (LDAV), 2012 IEEE Symposium on*, pp. 73–77. IEEE, 2012.

[21] E. B. Lum, K. L. Ma, and J. Clyne. Texture hardware assisted rendering of time-varying volume data. In *Proceedings of the conference on Visualization'01*, pp. 263–270. IEEE Computer Society, 2001.

[22] K.-L. Ma. In situ visualization at extreme scale: Challenges and opportunities. *IEEE Computer Graphics and Applications*, 29(6):14–19, 2009.

[23] G. McLachlan. *Discriminant analysis and statistical pattern recognition*, vol. 544. John Wiley & Sons, 2004.

[24] M. Meyer, S. Takahashi, and A. Vilanova. Data reduction techniques for scientific visualization and data analysis. *STAR*, 36(3), 2017.

[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*, 36(3):48–58, 2016.

[26] P. OLeary, 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*, 55:43–48, 2016.

[27] C. Sewell, K. Heitmann, H. Finkel, G. Zagaris, S. T. Parete-Koon, P. K. Fasel, A. Pope, N. Frontiere, L.-t. Lo, B. Messer, et al. Large-scale compute-intensive analysis via a combined in-situ and co-scheduling workflow approach. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, p. 50. ACM, 2015.

[28] R. Sicat, J. Kruger, T. Möller, and M. Hadwiger. Sparse pdf volumes for consistent multi-resolution volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2417–2426, 2014.

[29] D. Tao, S. Di, X. Liang, Z. Chen, and F. Cappello. Fixed-psnr lossy compression for scientific data. *arXiv preprint arXiv:1805.07384*, 2018.

[30] A. Tikhonova, C. Correa, and K.-L. Ma. Visualization by proxy: A novel framework for deferred interaction with volume data. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1551–1559, 2010.

[31] A. Tikhonova, C. D. Correa, and K.-L. Ma. An exploratory technique for coherent visualization of time-varying volume data. In *Computer Graphics Forum*, vol. 29, pp. 783–792. Wiley Online Library, 2010.

[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*, 2018.

[33] 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 *Pacific Visualization Symposium (PacificVis), 2017 IEEE*, pp. 161–170. IEEE, 2017.

[34] K.-C. Wang, N. Shareef, and H.-W. Shen. Image and distribution based volume rendering for large data sets. In *Pacific Visualization Symposium (PacificVis), 2018 IEEE*, pp. 26–35. IEEE, 2018.

[35] T.-H. Wei, S. Dutta, and H.-W. Shen. Information guided data sampling and recovery using bitmap indexing. In *Pacific Visualization Symposium (PacificVis), 2018 IEEE*, pp. 56–65. IEEE, 2018.

[36] J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image super-resolution via sparse representation. *IEEE transactions on image processing*, 19(11):2861–2873, 2010.

[37] H. Younesy, T. Möller, and H. Carr. Improving the quality of multiresolution volume rendering. In *EuroVis*, pp. 251–258. Citeseer, 2006.