Scalable Volume Visualization for Big Scientific Data Modeled by Functional Approximation

Jianxin Sun[1], David Lenz[3], Hongfeng Yu[1,2], Tom Peterka[3]
[1]*School of Computing, University of Nebraska-Lincoln, Lincoln, NE, USA*
[2]*Holland Computing Center, University of Nebraska-Lincoln, Lincoln, NE, USA*
[3]*Argonne National Laboratory, Lemont, IL, USA*

***Abstract*—Considering the challenges posed by the space and time complexities in handling extensive scientific volumetric data, various data representations have been developed for the analysis of large-scale scientific data. Multivariate functional approximation (MFA) is an innovative data model designed to tackle substantial challenges in scientific data analysis. It computes values and derivatives with high-order accuracy throughout the spatial domain, mitigating artifacts associated with zero- or first-order interpolation. However, the slow query time through MFA makes it less suitable for interactively visualizing a large MFA model. In this work, we develop the first scalable interactive volume visualization pipeline, MFA-DVV, for the MFA model encoded from large-scale datasets. Our method achieves low input latency through distributed architecture, and its performance can be further enhanced by utilizing a compressed MFA model while still maintaining a high-quality rendering result for scientific datasets. We conduct comprehensive experiments to show that MFA-DVV can decrease the input latency and achieve superior visualization results for big scientific data compared with existing approaches.**

***Index Terms*—volume visualization, functional approximation, big scientific dataset, distributed computing**

I. Introduction

Advancement in instrumentation and technology has enabled scientists in diverse fields, such as medical imaging, meteorology, materials science, and physical simulations, to generate or obtain 3D volumetric datasets at an increasingly rapid rate, which necessitate scalable volume visualization solutions to facilitate scientists to explore and gain discoveries from their datasets [12], [30]. However, creating volumetric visualizations for large-scale scientific datasets presents distinctive performance and quality challenges that require comprehensive solutions to address them holistically.

First, managing substantial scientific datasets on hardware systems with constrained memory resources and I/O bandwidths presents significant space and time complexities. These complexities can easily incur performance bottlenecks and further complicate the task of delivering a smooth user experience for interactively visualizing large-scale datasets. Researchers have developed various techniques to address this performance challenge. Out-of-core techniques [31] partition the dataset into manageable chunks that can be rendered independently, ensuring memory usage remains constant. Multi-resolution volume rendering methods [34] decrease the volume of data sent to the rendering pipeline based on zoom levels, rendering data at higher resolutions only when required. Data compression techniques [35] aim to reduce the size of datasets generated in scientific research while preserving the essential information for processing and visualization. Data streaming strategies [4] facilitate an incremental rendering of a dataset as its availability progresses by employing push and pull models. However, these methods often compromise rendering quality (data compression and multi-resolution methods) or input latency (out-of-core and streaming methods).

Second, accurately retrieving values and gradients from arbitrary positions within a 3D volumetric space is challenging but necessary for volume visualization algorithms to generate high-quality images. Popular interpolation methods for volume visualization include low-order filters, such as nearest neighbor search (NNS) and trilinear interpolation, and high-order filters, such as tricubic and Catmull-Rom. However, these filters can relatively easily generate artifacts in the interpolated values and gradients. Functional data analysis [13], [28] uses functional approximation to represent the original discrete dataset for querying off-grid value and gradient with higher accuracy through the computing of geometric bases. An important consideration in functional data analysis involves selecting the appropriate family of basis functions. Common choices include Fourier [6], wavelet [15], and geometric splines [10] bases. More recently, Austin et al. [1] introduced the Tucker decomposition as a low-rank alternative. Majdisova and Skala also suggested the use of radial basis functions for particle data [21]. Multivariate functional approximation (MFA) [26] is a new function approximation model with advantages of high-order evaluation of both value and derivative anywhere in the spatial domain, compact representation for large-scale volumetric data, and uniform representation of both structured and unstructured data. The initial discrete data require an initial preprocessing step involving encoding or prefiltering [29], [38] to obtain the sequence of coefficients of the interpolating functions. However, the primary challenge when employing functional approximation lies in its computational complexity, making it less suitable for real-time applications with stringent demands for information query latency. This latency becomes more severe when handling large-scale datasets.

In this work, we aim to tackle both quality and performance challenges, and propose a scalable volume visualization solution for big scientific datasets. First, we select functional approximation, specifically MFA, in our pipeline for its evaluation accuracy and better rendering quality. Second, we utilize distributed computing to accelerate both data fetching latency and rendering latency, so that the overall input latency of visualizing a large-scale MFA model can satisfy the requirements of interactive visualization. We name our proposed pipeline as MFA-based distributed volume visualization (MFA-DVV). To the best of our knowledge, this is the first work utilizing MFA as a data representation with distributed computing to speed up the rendering process for responsive user exploration of large-scale scientific datasets. The main contributions of this work include:

- A novel interactive volume visualization framework (MFA-DVV) using the MFA model with distributed computational architecture to visualize big scientific datasets with high quality and low input latency.
- Detailed performance analysis of critical components of the proposed pipeline.
- Comprehensive experiments studying the effect of using the compressed MFA model to optimize the performance.

The subsequent sections of this paper are organized as follows: we first provide an overview of related work in Section II and the background of MFA in Section III. Section IV outlines the architecture and implementation of the proposed MFA-DVV. Experimental results and evaluation are presented in Section V. We draw our conclusions in Section VI.

II. Related Work

A. Large-scale Volume Visualization

Researchers have developed various algorithms to visualize large-scale scientific data, encompassing tasks such as isosurface computation, streamline computation, and I/O-efficient volume rendering [5], [23]. Through the consideration of the distance between the camera view and individual data segments within the current view, multi-resolution techniques [34], [37] intelligently load data segments at different levels of detail. This approach reduces the volume of data loaded for rendering while preserving a comparable level of rendering quality. Efficiently accessing raw data in real-time visualization tasks such as progressive slicing and particle traces can be facilitated by employing strategies like an optimized disk data layout [25], or by utilizing pre-computed lookup tables [8]. Cox and Ellsworth [9] introduce a framework for out-of-core scientific visualization systems. This framework involves modifying the I/O subsystem to implement application-controlled demand paging. The approach capitalizes on the observation that many crucial visualization tasks only require access to a small portion of extensive datasets at a given time. Similarly, Ueng et al. [33], and Leutenegger and Ma [18] adopt a comparable strategy to load data on demand for visualization tasks by reorganizing the physical data on disk and employing structures like octrees or R-tree partitions to handle both structured and unstructured data.

B. Data Compression

The substantial size of large-scale scientific dataset results in prolonged data retrieval times due to constraints in bandwidths of various I/O types. Lossy compression [16] can be used to mitigate issues arising from I/O intensive workloads during data storage and transfer. TTHRESH [2] (using Tucker decomposition [3]), TAMRESH [37] (using global tensor approximation [17] factor matrices), ZFP [19] (a floating-point compressor using custom transform matrices), SZ [11] (using best fit curve-fitting compression) and SQ [14] (preserving connected and coherent regions) are popular choices. Although some compression algorithms, like ZFP, support random-access decompression [32], none have the ability to query at arbitrary locations away from the sample locations of the original discrete dataset, meaning they are subject to interpolation artifacts. Multivariate functional approximation (MFA) can model large-scale raw volumetric data into an MFA model with a compact size. Various volume visualization techniques can directly render results from the MFA model without referencing the original data, enabling large-scale volume visualization under limited memory resources.

C. Acceleration Techniques

Visualizing large datasets can be accelerated by leveraging environments and platforms like multi-core CPUs, GPUs, and high-performance computing (HPC) clusters [40]. Piringer et al. [27] present a generic multithreaded visualization architecture that helps avoid pitfalls related to multithreading with visual feedback. Piringer et al. [20] develop POIViz using radial representation to compute a 2D layout of the multidimensional dataset in parallel on CPU and GPU to improve the visualization of large datasets on a single computer. GPU-based large-scale volume visualization [5] combines the parallel processing power with out-of-core methods and data streaming to improve interactivity. In this work, we exploit distributed computing power to develop a scalable volume visualization solution.

III. Multivariate Functional Approximation

A. Background

Multivariate Functional Approximation (MFA) [26] represents discrete high-dimensional scientific data using a functional basis representation derived from the tensor product of B-spline functions. The geometric characteristics of the field and the values of the scientific data are captured and efficiently represented through a collection of control points and knots. Figure 1 illustrates the key steps to construct an MFA model from input data. Initially, the system computes parameterization and establishes an initial knot distribution with the minimum required number of control points. Additional control points and knots are dynamically introduced until all evaluated points within each knot span meet the specified maximum allowable relative error compared to the original points. During this iterative process, knot spans exceeding the tolerance threshold are subdivided, and the functional approximation is recalculated. From the input data to its MFA model, various levels of compression can be achieved by adjusting the number of control points to generate a compact MFA model.

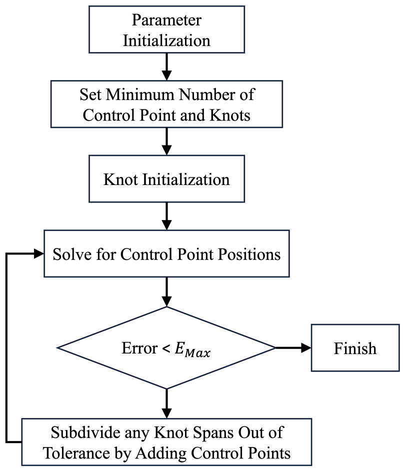

Parameter Initialization
Set Minimum Number of Control Point and Knots
Knot Initialization
Solve for Control Point Positions
Error < E_{Max}
Finish
Subdivide any Knot Spans Out of Tolerance by Adding Control Points

Fig. 1: Key steps to construct an MFA model from raw input data. E_{Max} is the maximum allowable relative error.

B. Advantages of MFA

In contrast to other local filters that aim to address numerous small local optimization problems, MFA represents the resolution of a solitary global optimization spanning the entire domain. As a result, MFA gives more accurate approximation than other high-order local filters like tricubic and Catmull-Rom [36]. The detailed advantages of MFA are:

- While local filters either interpolate or precisely match input points, MFA approximates them, which permits the smoothing of high-frequency discretization artifacts while retaining information about underlying low-frequency patterns.
- Local filters tend to reduce continuity between successive filter applications, whereas MFA, functioning as a global model, maintains high-order continuity consistently across the entire domain.
- Local filters determine their size and position by adapting to the distribution of input points, effectively behaving like a sliding window over these points. Conversely, MFA distributes its piecewise polynomials based on predetermined knot positions, which remain independent of the input point distribution. These knot locations can accurately reflect the data's complexity rather than relying on the input point distribution.
- Local filters may rely on finite approximations for gradient calculations, resulting in approximate high-order derivatives, especially at cell boundaries. On the other hand, MFA provides analytical high-order derivatives, ensuring precision in its gradient computations.
- Although the model is an approximation, its derived values, including gradients, are analytically accurate. Additionally, it offers analytical availability for higher-order derivatives, extending up to the polynomial degree.

C. Opportunities for MFA

To achieve effective volume visualization, existing techniques necessitate searching or retrieving values and gradients from arbitrary locations within the volume. As a result, the latency at which this information is retrieved determines the overall performance of a volume visualization system. This retrieval latency is particularly pronounced when handling large datasets. While MFA excels at modeling large-scale scientific data in situ and provides more accurate evaluations than many local filters, its query time becomes the primary performance bottleneck for visualizing the MFA model. Although MFA's query time outperforms popular high-order local filters such as tricubic and Catmull-Rom, its query performance falls short when compared to straightforward local filters like nearest neighbor and trilinear interpolation. We aim to improve the performance of volume visualization of the MFA model encoded from large-scale datasets. We build a distributed volume visualization architecture for rendering the MFA model with improved overall input latency. Our work also utilizes the compressed MFA model to achieve a better performance and rendering quality than the general distributed volume rendering using the raw discrete data through trilinear interpolation [39].

IV. Methods

A. Architecture

Figure 2 shows the architecture of our developed MFA-based distributed volume visualization pipeline, MFA-DVV, for handling large-scale scientific datasets using multiple processors of a computing cluster or supercomputer. There are six key steps to generate the final visualization result.

1) Volume Partitioning: The input large-scale volume is first partitioned into blocks that will be distributed among processors. This step is a recursive binary partitioning until the target number of partitions is achieved. The partition follows the round-robin order among the volume's x, y, and z directions. If n recursions of partitioning are executed, then the total number of partitioned blocks is 2^n. The partitioned data block will be the input of the MFA encoder to generate the MFA model.

2) MFA Encoding: Each partitioned block is encoded using the MFA encoder with a configuration where the key encoding parameters are the number of control points and the polynomial degree. Control points serve as coefficients that scale piecewise-continuous basis functions within the hypervolume of B-splines. Typically, the quantity of control points is equal to or fewer than the number of the input sample points. Meanwhile, the polynomial degree represents the degree of the basis function employed for modeling. The number of control points is the key parameter to determine the actual size of the encoded MFA model, which will be further evaluated in Section V-D. Encoding is done following the previous partition step, and partitioned blocks are processed together for encoding on corresponding nodes in parallel.

3) Data Fetching: When the visualization starts, MFA models are first fetched to the memory of each target node for rendering. Each processing node only retrieves the MFA model file encoded from the corresponding partitioned block. The fetching time, determined by the I/O bandwidth, is one of the key contributors to the overall input latency.

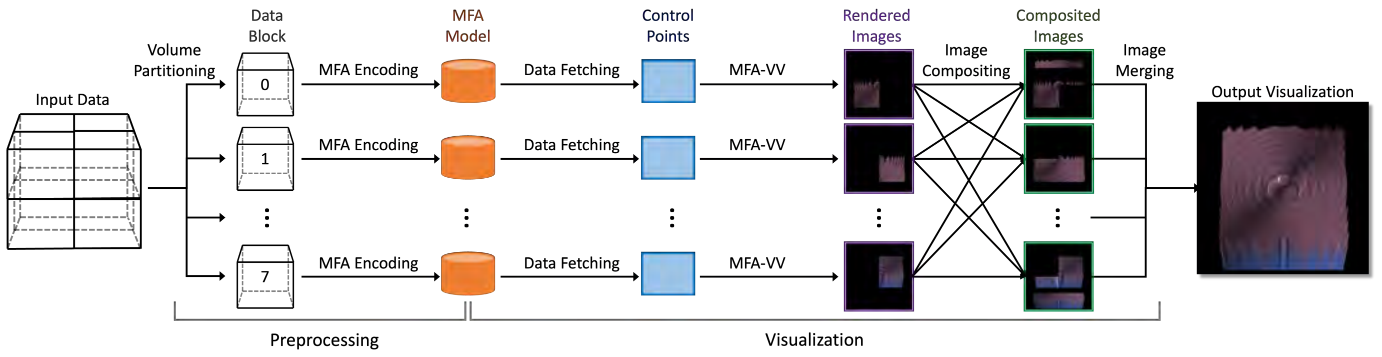


Fig. 2: Architecture of our MFA-DVV. In this example, the input data is partitioned into eight octants with identical size.

4) MFA-VV: An MFA-based volume visualization (MFA-VV) pipeline generates the required visualizations, like volume rendering and isosurface rendering, by considering the user configurations. Various volume visualization methods can utilize the query interfaces to directly query value and gradients from the MFA model to construct the visual representation. In this paper, the MFA-VV is ray-casting-based volume visualization utilizing the over operation between the neighboring samples on the ray segment. The rendering time is the main contributor to the final input latency. The complexity of rendering time is $O(n)$ where n is the number of queries for value and gradient within the volume space. Since such queries of a functional approximation are expensive, as discussed in Section III-C, we parallelize the rendering pipeline on m computing cores such that each node only needs to perform n/m queries to finish its rendering of the local block. This approach significantly accelerates the rendering time on multiprocessor systems, thereby reducing the overall input latency of visualizing large-scale MFA models. The output of MFA-VV from each node is a partially-rendered image that must be aggregated with other images for the final resulting image.

5) Image Compositing: Partially-rendered images of partitioned blocks are combined together to construct the final image. Our image compositing utilizes the associative property of the over operation. As long as the distances of the two images to the viewpoint are comparable, the correct composited image can be obtained through the over operation in the pixel level of the two images. Image compositing requires communications between nodes for exchanging pixels, and the compositing time is determined by the resolution of the final image and how many nodes are involved in the compositing process. We utilize binary-swap to schedule the communication between nodes efficiently [39]. After the compositing, only a subset of the total pixels of each composited image are correct. Section V-C will investigate the compositing time with respect to the number of nodes employed for this task.

6) Image Merging: Correct pixels from each node are sent to the master node for concatenation into the final volume visualization image. Section V-C will also reveal how the image merging time scales when the number of nodes increases.

B. Implementation

During the course of the MFA-DVV pipeline, Steps 1 and 2 (i.e., volume partitioning and MFA encoding) are preprocesses done before the visualization. The visualization pipeline is from Step 3 (i.e., data fetching) to 6 (i.e., image merging) and executes in response to every data-dependent (e.g., changing transfer functions) or view-dependent (e.g., change view parameters) operation by the user. The parallel library we used to execute MFA encoding, data fetching, MFA-VV, and image compositing is DIY ("Do-It-Yourself" Parallel Analysis) [24], which is a block-parallel library for implementing scalable distributed- and shared-memory parallel algorithms that can run both in- and out-of-core. DIY supports data distribution in parallel through its write and read I/O functions, which are utilized for an MFA model writing to storage after encoding and data fetching from storage to system memory of each node. The binary-swap image compositing algorithm can be readily integrated by leveraging DIY's merge-reduce mechanism for managing communication. In addition, our MFA-DVV provides a user interface so that users can freely adjust data-dependent (e.g., setting for transfer functions and shading parameters) and view-dependent (e.g., exploring view parameters) operations. The setting of all operations will be passed to MFA-DVV for generating the corresponding visualization as desired.

V. Results and Evaluation

We design and conduct comprehensive experiments to evaluate the scalability and performance of the MFA-DVV. Additionally, we explore the impact of essential MFA encoding parameters on these factors.

A. Datasets

1) Synthetic Datasets: We use the Marschner-Lobb synthetic function to gain an accurate ground-truth reference for volume rendering. Marschner-Lobb is a function initially introduced for evaluating 3D resampling filters when applied to the conventional Cartesian cubic lattice [22]. We create corresponding discrete datasets derived from the Marschner-Lobb function to serve as the input for MFA-DVV and other discrete volume rendering algorithms. Accurately reconstructing a Marschner-Lobb signal from discrete samples is challenging due to its complex amplitude distribution across various frequencies. This complexity makes it a valuable benchmark for evaluating reconstruction quality. The Marschner-Lobb function F_{ML} used in this paper is defined as:

$$F_{ML}(x,y,z) = \frac{1-\sin(\frac{\pi z}{2}) + \alpha(1+\rho_r(\sqrt{x^2+y^2}))}{2(1+\alpha)} \quad (1)$$

where

$$\rho_r(r) = \cos(2\pi f_M \cos(\frac{\pi r}{2}))$$

We use $f_M = 6$ and $\alpha = 0.25$ to generate discrete Marschner-Lobb datasets with different resolutions. The spatial boundaries on each dimension are $[0,7]$.

2) Regular Real Datasets: We further evaluate the rendering quality of MFA-DVV using real discrete datasets. These datasets are rendered using MFA models encoded at various compression levels. Specifically, we showcase the rendering quality on small datasets, such as Fuel and Nucleon, with sizes of 41^3 and 64^3, respectively. Fuel represents a simulation of fuel injection into a combustion chamber, while Nucleon simulates the two-body distribution probability of a nucleon in an atomic nucleus. Additionally, we examine larger datasets, Aneurysm and Bonsai, both with sizes of 256^3. Aneurysm represents a rotational angiography scan of a head with an aneurysm, while Bonsai is a CT scan of a bonsai tree. The spatial boundaries for all the real datasets in each dimension range from 0 to 255.

3) Large-scale Real Dataset: The large-scale scientific dataset used to test the capability of MFA-DVV is the Richtmyer-Meshkov Instability (RMI) simulation [7]. The Richtmyer-Meshkov instability arises when a shock wave interacts with an interface separating two different fluids. The original time-varying RMI dataset is over 2TB. The input volume we use is the 160th time step with a spatial resolution of $1024 \times 1024 \times 1024$.

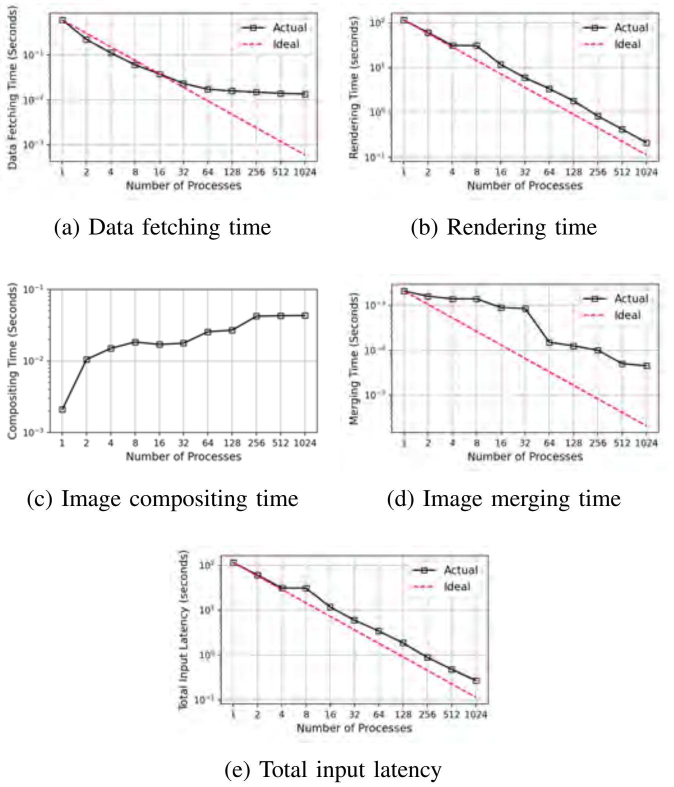
B. Experiment Setup

We run experiments using MFA-DVV on the Swan cluster at the Holland Computing Center (HCC) of the University of Nebraska-Lincoln. Swan is a massively parallel processing system with 9408 computing cores. There are 168 nodes in total, and each node has an Intel Xeon Gold 6348 CPU (56 cores) with 256GB RAM. Swan provides high-performance, low-latency communication for MPI jobs. We run our job in parallel using different processor numbers ranging from 1 to 1024. Due to the nature of the binary partition of the input data, the number of processors used is 2^n, where n is the times of partitions executed. Our test uses n from 1 to 10.

C. Scalability of MFA-DVV

We perform the scalability test of MFA-DVV using the Marschner-Lobb dataset. The testing volume is a scalar field with a spatial resolution of $256 \times 256 \times 256$ with the float32 data type. We first preprocess the input discrete data with data partitioning and MFA encoding. The input data is encoded by MFA also using 256x256x256 control points, which is the same as the size of the input discrete sample of the dataset. The polynomial degree of the MFA encoder is set to 2 for non-linear approximation. Table I shows the partitioned block size, the number of control points for MFA encoding, and the parallel core counts with respect to the partition patterns. For high-quality visualization, a fine sample step is used for the ray-casting rendering step of MFA-DVV. The resolution of the final rendering image is 768×768. The total input latency of MFA-DVV on a large-scale dataset depends on the visualization time in its pipeline, including data fetching latency, MFA-VV rendering time, image compositing time, and image merging time. We summarize the time measurement for each of the four components and total input latency in Figure 3. From the results, we can see that, for a general size volume, the main contributor to the input latency of MFA-DVV is the rendering time, as the times used in other states are comparably much smaller. The scalability results when handling large-scale data will be investigated in the next section.

The data fetching time depends on the size of the MFA model encoded for each partitioned block. Since the number of control points used for encoding the MFA model is the same as the size of the samples of input partitioned blocks, the MFA model has the same size as the input block. As the size of the blocks goes down together with the MFA model when using more processes, data prefetching time also decreases.



(a) Data fetching time (b) Rendering time

(c) Image compositing time (d) Image merging time

(e) Total input latency

Fig. 3: Time breakdown of MFA-DVV rendering pipeline and total input latency.

TABLE I: Setting of volume partition and number of MFA control points with respect to the partition patterns.

Partition	$1\times1\times1$	$2\times1\times1$	$2\times2\times1$	$2\times2\times2$	$4\times2\times2$	$4\times4\times2$	$4\times4\times4$	$8\times4\times4$	$8\times8\times4$	$8\times8\times8$	$16\times8\times8$
Block Size	$256\times256\times256$	$128\times256\times256$	$128\times128\times256$	$128\times128\times128$	$64\times128\times128$	$64\times64\times128$	$64\times64\times64$	$32\times64\times64$	$32\times32\times64$	$32\times32\times32$	$16\times32\times32$
Ctrl Points	$256\times256\times256$	$128\times256\times256$	$128\times128\times256$	$128\times128\times128$	$64\times128\times128$	$64\times64\times128$	$64\times64\times64$	$32\times64\times64$	$32\times32\times64$	$32\times32\times32$	$16\times32\times32$
Cores	1	2	4	8	16	32	64	128	256	512	1024

Because the dataset used here is relatively small in size, and the partitioned blocks can become very small (e.g., as small as 64KB) as more processes are used, it reaches the saturation points of the I/O interfaces, as shown in Figure 3a. For a large-scale dataset, the data fetching time still scales well because the partitioned data block will not reach such a small limit in file size.

The rendering time scales with the number of processes as shown in Figure 3b for its computational cost are distributed evenly across all parallel processing nodes.

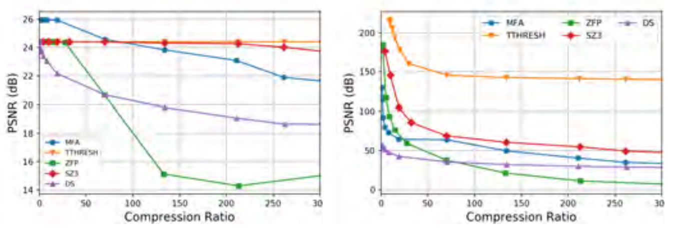
Figure 3c shows the image compositing time. More communications are involved when more processes are used, as more images are needed for compositing. However, the compositing time increases marginally as the number of process increases, and each compositing operation between two images in each round only perform one over operation compared to many over operations executed in the MFA-VV step. As a result, the compositing time is normally not comparable with the rendering time.

The final image is merged in the master node by concatenating pixels sent from all the working nodes. As shown in Figure 3d, the image merging time also scales with the number of processes. More processes mean a smaller number of pixels are transmitted to the master node for image merging. Compared to the image compositing step that requires multiple rounds of communication among all working nodes, image merging only requires one communication between the working nodes to the master node. Thus, the image merging time is normally the fastest step of the pipeline.

By summing up all the time used in each step, Figure 3e shows that the total input latency of MFA-DVV scales well with respect to the number of processes, where the decreasing rate of the measured total input latency is almost the same as the ideal rate.

D. *MFA-DVV using Compressed MFA Model*

1) Quality Evaluation using Compressed Data: We quantitatively evaluate the rendering quality of MFA-DVV using compressed MFA models compared with other popular compression algorithms, including TTHRESH, ZFP, SZ3, and downsampling (DS), in both image and volume domains. MFA-DVV generates volume rendering results directly from the MFA model, while the employment of other compression algorithms needs to decompress data first and then render the decompressed discrete data. The Marschner-Lobb dataset of the size 256^3 is used as the original data for compression. Figure 4a shows the image quality PSNR scores on volume-rendered images with respect to compression ratio. We observe that MFA-DVV always gives better rendering quality than ZFP and DS. Its rendering



(a) Image space quality (b) Volume space quality

Fig. 4: Compression evaluation using MFA and other compression algorithms.

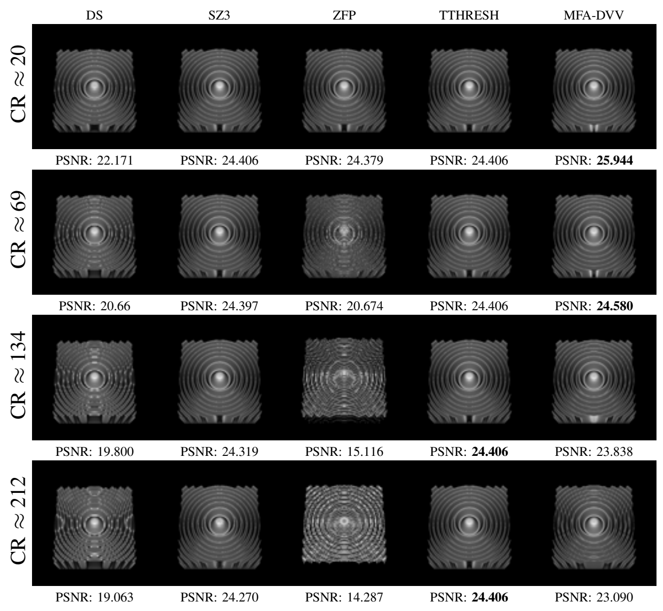


Fig. 5: Volume rendering results under different compression ratio (CR) using proposed MFA-DVV with compressed MFA model and other volume compression algorithms.

result is better than SZ3 and TTHRESH for a compression ratio of less than 70. For aggressive compression, SZ3 and TTHRESH give better results than MFA-DVV. The volume-rendered images with their PSNR scores under various levels of compression ratio are shown in Figure 5. We can see that MFA-DVV does not have high-frequency artifacts on the rings of the ripple seen in other compression algorithms. This shows the effectiveness of high-order value and gradient estimation for off-grid locations, while other algorithms rely on linear interpolation. For compression evaluation in the volume domain, the reconstruction error is calculated by comparing values on original discrete sample locations. As shown in Figure 4b, MFA-

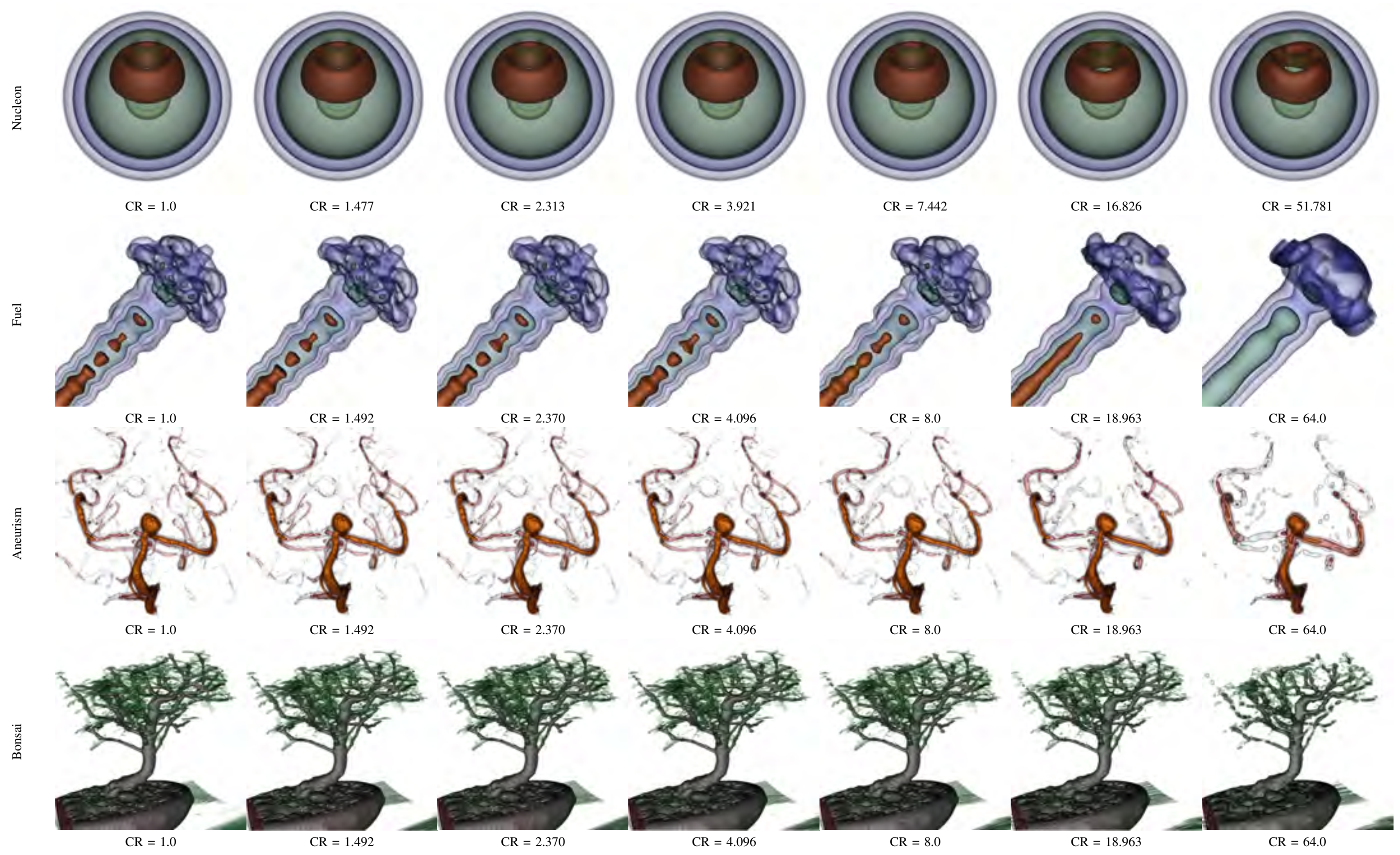


Fig. 6: MFA-DVV volume rendering results of MFA model with different compression ratio (CR) encoded using various numbers of control points.

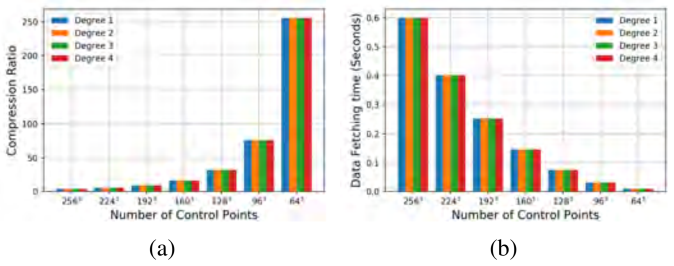
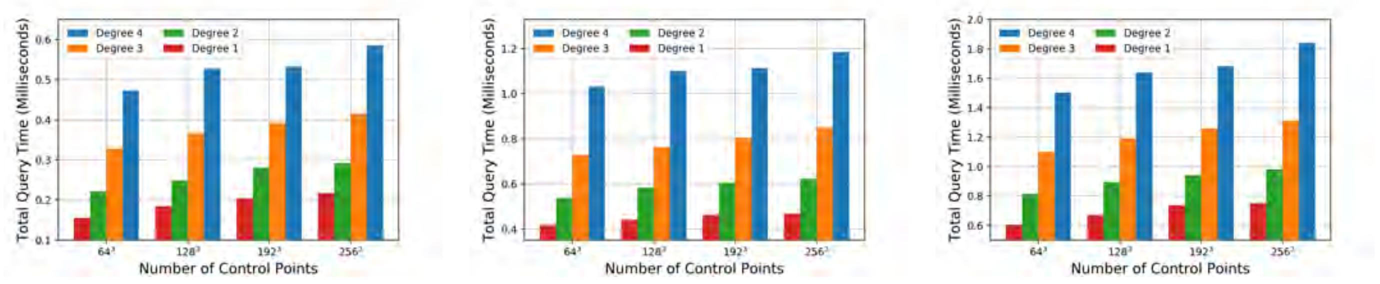


Fig. 7: MFA compression ratio and data fetching time using different numbers of control points and polynomial degree.

DVV always performs better than ZFP and better than DS for higher compression ranges. However, SZ3 and TTRESH are generally better than MFA-DVV. This means the MFA model is not the best data compressor targeting reconstruction of data values on-grid locations, which is what lossy floating-point compressors are designed to do. However, MFA's accurate query of off-grid locations improves its volume rendering quality in the image domain, as shown in Figure 4a, compensating for its accuracy in the volume domain.

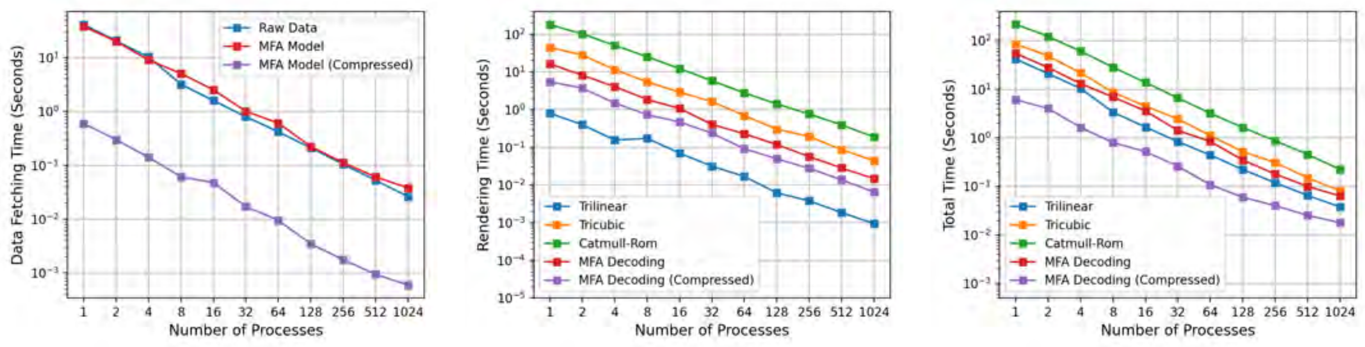
We also qualitatively evaluate the volume rendering using real datasets under various levels of compression. The encoded MFA model of the input discrete raw dataset is compressed by adjusting the number of control points as shown in Figure 6. The first column of the figure is the volume rendering result using MFA-DVV without compression, where the number of the control points is equal to the number of points of the input discrete dataset. As the compression ratio (CR) increases, the rendering quality decreases. For sparse datasets like Nucleon and Fuel, although MFA-DVV using moderate compression can capture the main structure of the data, the results have noticeable errors compared to using an uncompressed model. Because the Nucleon dataset is simpler than the Fuel dataset, its result with a similar level of compression gives more faithful results. However, due to the limited number of control points, an extremely compressed MFA model results in inaccurate volume rendering results, as shown in their right images. For datasets with high resolutions, like Aneurysm and Bonsai, the compression ratio can be more aggressive because there are more sample points to describe the dynamics of the data. As shown in the middle figures, the rendering error of using moderate compression is not very noticeable. For high-resolution data, the maximal compression ratio the MFA-DVV can achieve is determined by the complexity of the specific data.

2) MFA Parameter Study on Data Fetching Time: The number of control points is the main MFA parameter that determines



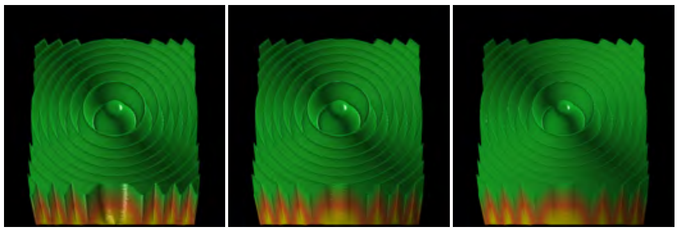
(a) Value query time (b) Gradient query time (c) Overall query time

Fig. 8: Query performance from MFA model for value and gradient using various number of control points. The overall query time is the sum of the query times for value and gradient.



(a) Input loading I/O time (b) Rendering time (c) Total time

Fig. 9: Scalability of data fetching time, rendering time, and total time using various ways of information query.



(a) Trilinear (b) MFA-DVV (compressed MFA model) (c) Ground Truth

Fig. 10: Rendering results using trilinear interpolation, MFA-DVV of compressed MFA model, and the ground truth.

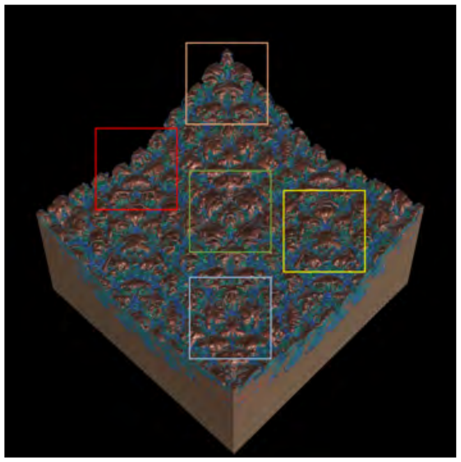
TABLE II: Quantitative comparison of rendering quality between trilinear and MFA-DVV using compressed MFA model.

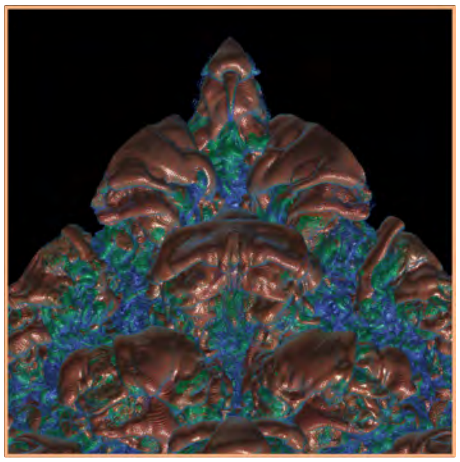
Methods	MSE	PSNR (dB)	SSIM
Trilinear	205.46	25.00	0.89
MFA-DVV (Compressed)	**50.76**	**31.08**	**0.96**

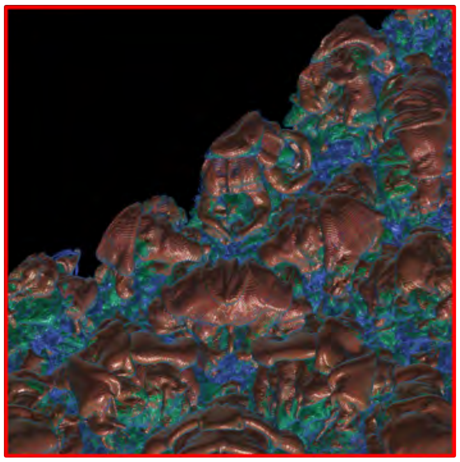
the compression ratio of the MFA model. Specifically, as shown in Figure 7a, the size of the compressed MFA model is proportional to the number of control points used for encoding, while the polynomial degree almost has no influence on the size of the MFA model. As the MFA model decreases due to compression, its data fetching time also decreases, as shown in Figure 7b. This time-saving becomes more significant when handling large-scale datasets.

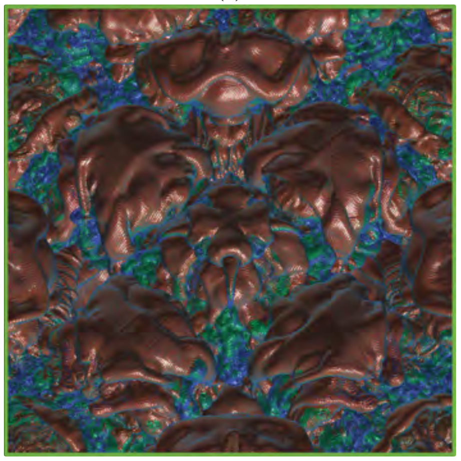
3) MFA Parameter Study on Rendering Time: MFA model encoded using fewer control points will give faster query times for both value and gradient because fewer items (i.e., basis functions and control points) are involved in the calculation. General volume visualization needs to query both value and gradient for better visualization with a shading effect. Thus, faster queries with a compressed MFA model encoded with fewer control points will reduce the rendering time. Figure 8 shows how the number of control points affects the query time used to retrieve the value and gradient for a location of interest within the domain.

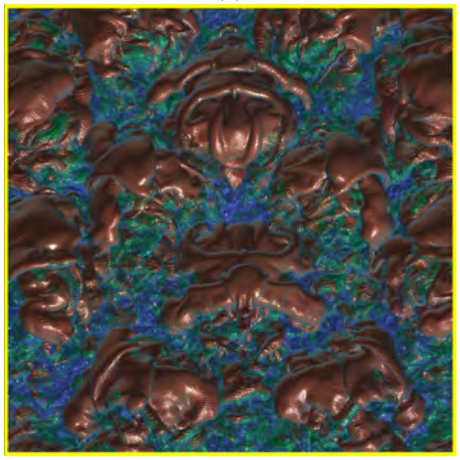
4) Performance Evaluation: We now evaluate the performance of MFA-DVV using a compressed MFA model encoded from a large-scale dataset. The large-scale dataset is derived from the Marschner-Lobb function with a spatial resolution of $1024 \times 1024 \times 1024$. We generate two MFA models, with and without compression, respectively. The non-compressed MFA model uses $1024 \times 1024 \times 1024$ control points while the compressed MFA model uses $256 \times 256 \times 256$ control points. Thus, the size of the compressed MFA model is $1/64$ of the original dataset. We used various query methods for


(a)


(b)


(c)


(d)


(e)

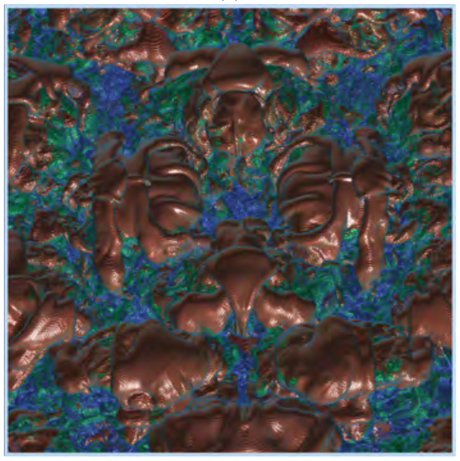

(f)

Fig. 11: Rendering of RMI large-scale dataset using MFA-DVV, where (a) is an overview of the data with an image resolution of 4096×4096, and (b) to (f) show 5 zoom-in view with a 512×512 image resolution. Detail of the dataset can be clearly observed, enhancing the comprehension of the large dataset.

performance comparison on rendering time. We consider the most commonly used trilinear interpolation and two high-order local filters which are tricubic and Catmull-Rom. Figure 9a shows the data fetching scalability that depends on the size of the input file, raw data, MFA model, and the compressed MFA model. We can see that the compressed MFA model saves considerable time on data fetching. Figure 9b shows the rendering scalabity. All high-order approximations, two MFA models, tricubic and Catmull-Rom, have longer rendering time than trilinear interpolation. The MFA model is generally faster than the other two high-order filters. Due to the computation reduction, the compressed MFA model performs better than the non-compressed version. As shown in Figure 9c, with the help of improved data fetching latency and rendering latency, MFA-DVV gives the best overall performance when rendering a compressed MFA model. Figure 10 and Table II clearly show that MFA-DVV not only surpasses the already fast trilinear interpolation but also achieves more accurate rendering results.

E. *Visualization Example*

In Figure 11, we showcase the visualization result of the MFA model encoded from a large-scale real dataset, RMI, with an overall view and several zoom-in views. This dataset boasts a high image resolution of 4096×4096. The results highlight the effectiveness of our high-resolution, high-precision distributed solution of MFA-DVV, empowering scientists to discern intricate details from vast amounts of data.

VI. Conclusions

We present MFA-DVV, a distributed volume visualization framework to render large-scale scientific datasets modeled by functional approximation. The proposed framework leverages Multivariate Functional Approximation (MFA) to improve rendering accuracy and achieve low input latency through a distributed architecture. Compared to existing compression methods, our MFA-DVV can balance better between image space quality and volume space quality. We examine the impact of essential MFA encoding parameters on both data fetching latency and rendering latency. Experimental results demonstrate that MFA-DVV showcases strong scalability, and its performance can be significantly enhanced by utilizing a compressed MFA model while still maintaining a high-quality rendering result for scientific datasets. Our MFA-DVV has provided scientists with high-quality and high-performance visualization to gain more detailed observations from their big scientific datasets. In the future, we would like to integrate new data models like implicit neural representations into our pipeline. Additionally, we are also interested in exploring alternative volume visualization techniques, such as inverse rendering, to offer diverse approaches for effectively visualizing large-scale volumes in parallel.

VII. Acknowledgement

This work is supported by Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contracts DE-AC02-06CH11357, program manager Margaret Lentz. This work was completed utilizing the Holland Computing Center of the University of Nebraska, which receives support from the UNL Office of Research and Economic Development, and the Nebraska Research Initiative.

References

[1] W. Austin, G. Ballard, and T. G. Kolda. Parallel tensor compression for large-scale scientific data. In *2016 IEEE international parallel and distributed processing symposium (IPDPS)*, pages 912–922. IEEE, 2016.

[2] R. Ballester-Ripoll, P. Lindstrom, and R. Pajarola. Tthresh: Tensor compression for multidimensional visual data. *IEEE Transactions on Visualization and Computer Graphics*, 26(9):2891–2903, 2020.

[3] R. Ballester-Ripoll and R. Pajarola. Lossy volume compression using tucker truncation and thresholding. *Vis. Comput.*, 32(11):1433–1446, nov 2016.

[4] J. Beyer, M. Hadwiger, A. Al-Awami, W.-K. Jeong, N. Kasthuri, J. W. Lichtman, and H. Pfister. Exploring the connectome: Petascale volume visualization of microscopy data streams. *IEEE computer graphics and applications*, 33(4):50–61, 2013.

[5] J. Beyer, M. Hadwiger, and H. Pfister. State-of-the-art in gpu-based large-scale volume visualization. *Computer Graphics Forum*, 34(8):13–37, 2015.

[6] B. Boashash. *Time-frequency signal analysis and processing: a comprehensive reference*. Academic press, 2015.

[7] M. Brouillette. The richtmyer-meshkov instability. *Annual Review of Fluid Mechanics*, 34(1):445–468, 2002.

[8] R. Bruckschen, F. Kuester, B. Hamann, and K. I. Joy. Real-time out-of-core visualization of particle traces. In *Proceedings of the IEEE 2001 Symposium on Parallel and Large-Data Visualization and Graphics*, PVG '01, page 45–50. IEEE Press, 2001.

[9] M. Cox and D. Ellsworth. Application-controlled demand paging for out-of-core visualization. In *Proceedings of the 8th Conference on Visualization '97*, VIS '97, page 235–ff., 1997.

[10] C. De Boor and C. De Boor. *A practical guide to splines*, volume 27. springer-verlag New York, 1978.

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

[12] D. El-Torky, M. N. Al-Berry, M. A.-M. Salem, and M. I. Roushdy. 3d visualization of brain tumors using mr images: a survey. *Current Medical Imaging*, 15(4):353–361, 2019.

[13] F. Ferraty. *Nonparametric functional data analysis*. Springer, 2006.

[14] J. Iverson, C. Kamath, and G. Karypis. Fast and effective lossy compression algorithms for scientific datasets. In C. Kaklamanis, T. Papatheodorou, and P. G. Spirakis, editors, *Euro-Par 2012 Parallel Processing*, pages 843–856, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[15] M. H. Jansen and P. J. Oonincx. *Second generation wavelets and applications*. Springer Science & Business Media, 2005.

[16] U. Jayasankar, V. Thirumal, and D. Ponnurangam. A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications. *Journal of King Saud University - Computer and Information Sciences*, 33(2):119–140, 2021.

[17] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[18] S. Leutenegger and K.-L. Ma. Fast retrieval of disk-resident unstructured volume data for visualization. *External Memory Algorithms and Visualization*, 50, 1999.

[19] P. Lindstrom. Fixed-rate compressed floating-point arrays. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2674–2683, 2014.

[20] T. Liu, F. Bouali, and G. Venturini. On visualizing large multidimensional datasets with a multi-threaded radial approach. *Distributed and Parallel Databases*, 34:321–345, 2015.

[21] Z. Majdisova and V. Skala. Radial basis function approximations: comparison and applications. *Applied Mathematical Modelling*, 51:728–743, 2017.

[22] S. Marschner and R. Lobb. An evaluation of reconstruction filters for volume rendering. In *Proceedings Visualization '94*, pages 100–107, 1994.

[23] F. Marton, M. Agus, and E. Gobbetti. A framework for gpu-accelerated exploration of massive time-varying rectilinear scalar volumes. *Computer Graphics Forum*, 38(3):53–66, 2019.

[24] D. Morozov and T. Peterka. Block-Parallel Data Analysis with DIY2. In *Proceedings of the 2016 IEEE Large Data Analysis and Visualization Symposium LDAV'16*, Baltimore, MD, 2016.

[25] V. Pascucci and R. J. Frank. Global static indexing for real-time exploration of very large regular grids. In *Proceedings of the 2001 ACM/IEEE Conference on Supercomputing*, 2001.

[26] T. Peterka, Y. Nashed, I. Grindeanu, V. Mahadevan, R. Yeh, and X. Trixoche. Foundations of Multivariate Functional Approximation for Scientific Data. In *Proceedings of 2018 IEEE Symposium on Large Data Analysis and Visualization*, 2018.

[27] H. Piringer, C. Tominski, P. Muigg, and W. Berger. A multi-threading architecture to support interactive visual exploration. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1113–1120, 2009.

[28] J. O. Ramsay and B. W. Silverman. *Applied functional data analysis: methods and case studies*. Springer, 2002.

[29] D. Ruijters and P. Thévenaz. Gpu prefilter for accurate cubic b-spline interpolation. *The Computer Journal*, 55(1):15–20, 2012.

[30] I. H. Sarker. Data science and analytics: an overview from data-driven smart computing, decision-making and applications perspective. *SN Computer Science*, 2(5):377, 2021.

[31] J. Sarton, N. Courilleau, Y. Remion, and L. Lucas. Interactive visualization and on-demand processing of large volume data: A fully gpu-based out-of-core approach. *IEEE Transactions on Visualization and Computer Graphics*, 26(10):3008–3021, 2020.

[32] J. Schneider and R. Westermann. Compression domain volume rendering. In *IEEE Visualization, 2003. VIS 2003.*, pages 293–300, 2003.

[33] Shyh-Kuang Ueng, C. Sikorski, and Kwan-Liu Ma. Out-of-core streamline visualization on large unstructured meshes. *IEEE Transactions on Visualization and Computer Graphics*, 3(4):370–380, 1997.

[34] 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*, 20(12):2417–2426, 2014.

[35] S. W. Son, Z. Chen, W. Hendrix, A. Agrawal, W.-k. Liao, and A. Choudhary. Data compression for the exascale computing era-survey. *Supercomputing frontiers and innovations*, 1(2):76–88, 2014.

[36] J. Sun, D. Lenz, H. Yu, and T. Peterka. MFA-DVR: direct volume rendering of mfa models. *Journal of Visualization*, pages 1–18, 2023.

[37] S. Suter, M. Makhynia, and R. Pajarola. Tamresh – tensor approximation multiresolution hierarchy for interactive volume visualization. *Computer Graphics Forum*, 32(3pt2):151–160, 2013.

[38] P. Thevenaz, T. Blu, and M. Unser. Interpolation revisited [medical images application]. *IEEE Transactions on Medical Imaging*, 19(7):739–758, 2000.

[39] H. Yu, C. Wang, and K.-L. Ma. Massively parallel volume rendering using 2–3 swap image compositing. In *SC '08: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, pages 1–11, 2008.

[40] J. Zhang, J. Sun, Z. Jin, Y. Zhang, and Q. Zhai. Survey of parallel and distributed volume rendering: revisited. In *Computational Science and Its Applications–ICCSA 2005: International Conference, Singapore, May 9-12, 2005, Proceedings, Part III 5*, pages 435–444. Springer, 2005.