A Distributed-Memory Parallel Approach for Volume Rendering with Shadows

Manish Mathai* University of Oregon
Matthew Larsen† Luminary Cloud
Hank Childs‡ University of Oregon

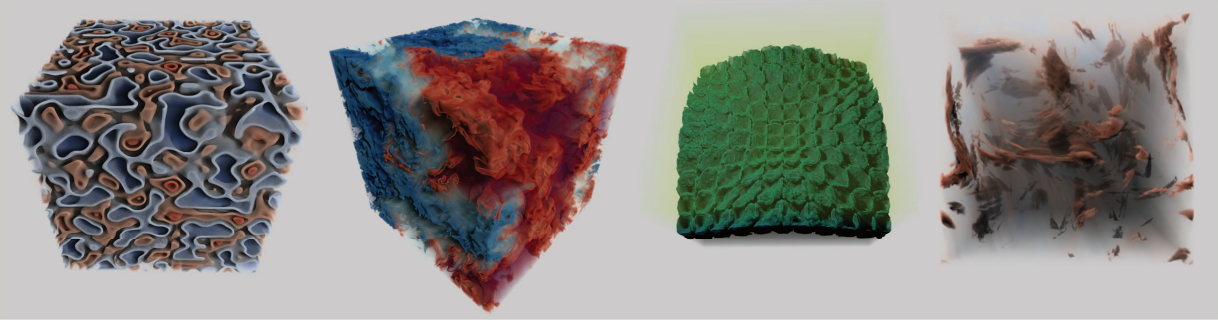


Figure 1: Volume renderings of four data sets using shadows. From left to right, the data sets are Perlin Noise [34], Rayleigh-Taylor Instability [11], Richtmyer-Meshhkov Instability [10], and Rotating Stratified Turbulence [20].

ABSTRACT

We present a parallel, distributed-memory technique that enhances traditional ray-casting volume rendering of large data sets to highlight the depth and perception of interesting volumetric features. The technique introduces a lighting system that accounts for global shadows across distributed MPI nodes while using shared-memory parallelism within each node to compute shading information efficiently. The first stage of the approach involves estimating energy attenuation from a point light source through the global volume, using a reduced spatial resolution representation of the volume, with minimal global communication between nodes. It is then used in the second stage during volume rendering to shade sample points captured during ray-casting, generating a high-quality image. In this work, we study the technique's performance across varying spatial resolutions of the estimated light attenuation using synthetic and real-world volumetric data sets on distributed systems.

1 INTRODUCTION

Scientific visualization is instrumental in enabling scientists to uncover complex phenomena. Using innovative graphic methods, scientists can traverse and examine volumetric data, retrieving vital information and unveiling potentially obscured patterns. Among the techniques available, volume rendering has proven to be very popular for visualizing volumetric data sets, as it permits the portrayal of complex structures and the description of data attributes in an intuitively comprehensible manner. However, depth and perception of interesting volumetric attributes are often inadequately captured by traditional volume rendering techniques.

Research in the graphics community has led to advancements in photo-realistic rendering techniques that produce visually compelling and immersive experiences. Path tracing and global illumination, for instance, have effectively simulated realistic lighting, shadows, and reflections. These advancements have significantly heightened the visual fidelity and realism of computer-generated imagery across a variety of sectors, including entertainment, virtual reality, and architectural visualization.

However, implementing photo-realistic rendering techniques on large volumetric data sets introduces a significant challenge. As data size and complexity expand, the computational demands become too excessive for a single processing unit to handle. Supercomputers are a promising environment, but many approaches were designed only for a desktop machine and need to be rethought to work in a distributed-memory setting. As a result, scientists' ability to uncover crucial insights it limited on large-scale volumetric data.

Addressing the challenges accompanying large volumetric data sets, we propose a parallel, distributed-memory method that heightens traditional distributed ray-casting volume rendering by incorporating shadows from a point light source. Our technique employs distributed-memory parallelism by utilizing multiple processing units interconnected through MPI. The approach allows the efficient distribution of the computational workload across multiple nodes, thus enabling the rendering of large volumes, considering global lighting effects and shadows.

The initial phase of our approach encompasses a method for estimating energy attenuation from point light source(s) through the distributed volume. We generate a lower-resolution representation of the input volume, called the "shadow volume," to minimize the inter-node communication overhead. This estimate is utilized in the second phase, the actual volume rendering process, to shade sample points gathered through ray-casting by estimating light contributions at sample locations, culminating in high-quality images that emphasize depth and perception of interesting volumetric features.

In our research, we focus on investigating the performance of our distributed-memory parallel technique across varying spatial resolutions of the shadow volume. To validate the effectiveness and applicability of our technique, we conduct experiments using synthetic and real-world volumetric data sets on distributed systems. We evaluate the performance and scalability of our approach to showcase its potential in facilitating the exploration and analysis of large-scale volumetric data, which in turn can enable scientists to gain valuable insights into their data.

*e-mail: mmathai@uoregon.edu
†e-mail: matt.larsen@luminarycloud.com
‡e-mail: hank@uoregon.edu

2023 IEEE 13th Symposium on Large Data Analysis and Visualization (LDAV) | 979-8-3503-3018-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/LDAV60332.2023.00010

2832-6512/23/$31.00 ©2023 IEEE
DOI 10.1109/LDAV60332.2023.00010

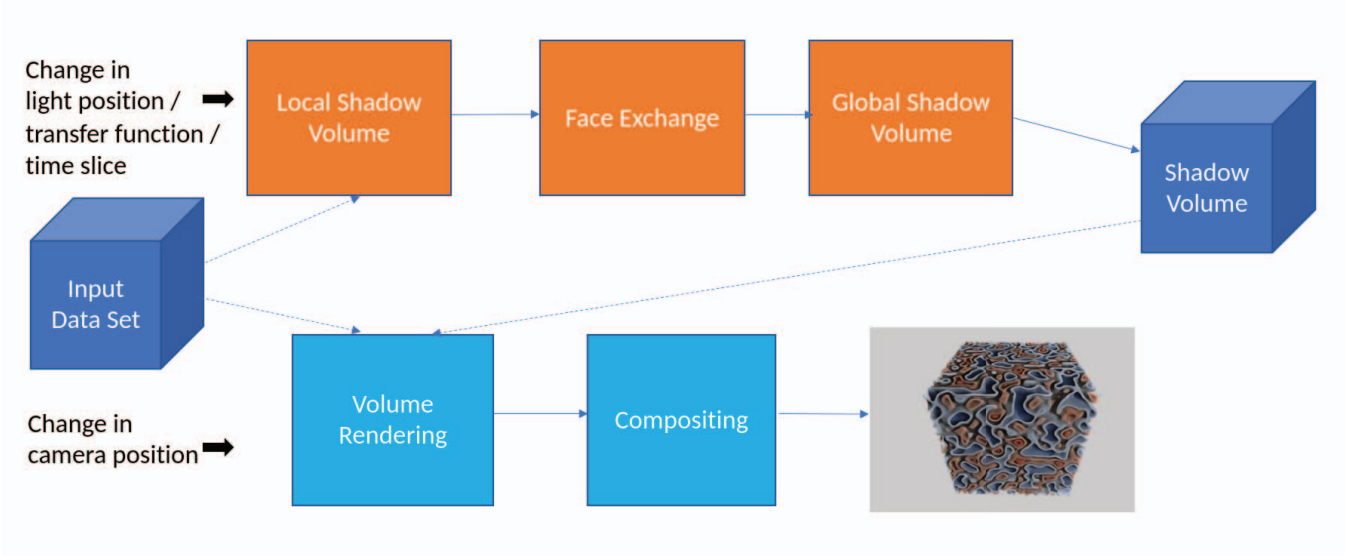


Figure 2: Illustration of our proposed algorithm. Section 3 contains more detail about the individual phases.

2 Related Work

There are two main areas of related work: illumination models for volume rendering and parallel volume rendering on supercomputers. Our work considers the intersection of these two topics, i.e., how to put shadows into the parallel volume rendering process.

2.1 Illumination Models for Volume Rendering

Illumination models for volume rendering and techniques are well studied. Since its inception, researchers strove to improve the visual quality of volume rendering. Volumetric path tracing [22], is the gold standard in visual quality, but it is a brute force algorithm that leverages Monte Carlo techniques to solve the rendering equation [29]. Even with modern hardware and variance reduction techniques, volumetric path tracing can be cost-prohibitive (i.e., interactivity is only achieved through progressive rendering) and complicated to implement in a distributed-memory setting.

Shadow maps [42] have long been used to approximate direct lighting contributions from light sources by rendering the scene from the point of the light into a depth buffer. Direct lighting calculations use values from the shadow map to determine if the point is in shadow or not. Deep shadow maps [23] extended shadow maps with a per-pixel fractional visibility function that enabled support for shadows from semi-transparent surfaces and volumes. Deep shadow maps were first applied to direct volume rendering by Hadwiger et al. [16] in the context of scientific visualization. To our knowledge, no distributed-memory implementations have been studied.

Behrens et al. [3] used discrete space shadow volumes in the context of a slice-based [6] volume rendering technique. At each point in the grid, the light attenuation along the path to the light source is stored. In sliced-based approaches, shadows are calculated by progressively attenuating the lighting, slice by slice, within a designated shadow volume. The closest work to our own, by Domonkos and Csébfalvi [12], extends the slice-based approach to a distributed-memory setting. Each rank locally renders both the volume slices and the shadow slices, then proceeds with a distributed-memory compositing phase that gathers the global information for each rank. The slice-based shadow map approach needs to be calculated for every frame. Both the texture slices and the shadow map must be composited for each frame. Support for multiple lights is possible, but the per-frame shadow overhead is a multiple of the number of lights. To our knowledge, the this work is the only other work that uses any shadow approximation technique in a distributed-memory setting. Their approach, however, has some fundamental disadvantages to our own. First, they calculate shadows along a series of planes; achieving effects similar to our approach would require compositing hundreds of planes and thus be quite slow. Second, their shadowing is calculated on a per-viewpoint basis, in contrast with our approach which can amortize the shadow volume calculation over many renderings.

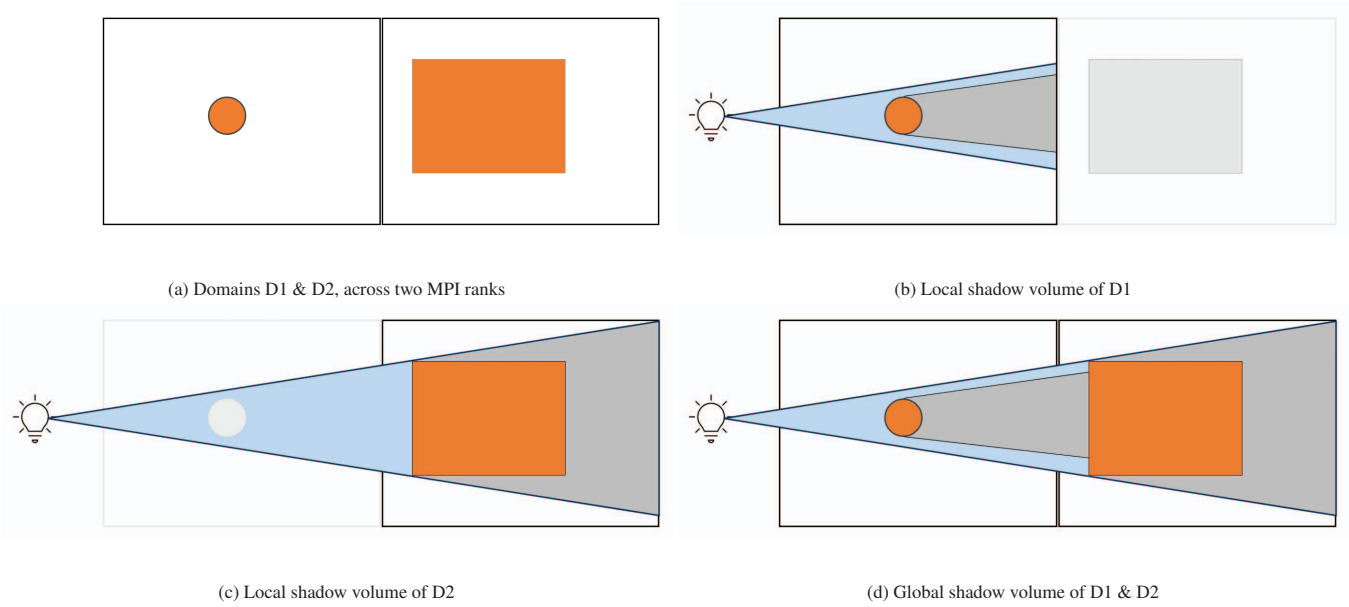
2.2 Parallel Volume Rendering on Supercomputers

Distributed-memory parallel volume rendering can be image order (over pixels), object order (over data), or a hybrid [5, 8, 30]. While some work has considered image order [14, 39], the majority has considered object order, as is our proposed technique. We assume the data is partitioned into blocks, and every block is available in the main memory of one MPI rank. These assumptions are consistent with in situ processing. Of note, other works have considered repartitioning data blocks, in order to facilitate data compression [2, 41], multi-resolution and out-of-core processing [4], and equal workload [24, 25, 28, 33, 40]. Other non-repartitioning works have focused on parallel exchanges, including binary swap [26, 43] and Radix-K [35, 37]. These latter works have often looked at massive scale, as have some additional works focusing strictly on scalability and barriers to scalability [9, 15, 17, 18, 36, 38].

3 Algorithm Description

3.1 Overview

Our algorithm augments ray-casted volume rendering with light attenuation information. For each sample along a ray, our algorithm evaluates the light attenuation at that sample – how much light makes it through volumetric data (incorporating opacity information from a transfer function) – and shades the sample color appropriately based on the attenuation value. The light attenuation is effectively a three-dimensional field, i.e., at every location in the three-dimensional data set, the field describes how much light makes it to that location. One possible approach for evaluating this field for a given sample is dynamic evaluation. In such an approach, evaluating each sample location would spawn a new calculation from the light source, through the volume, to the sample location. That said, this dynamic approach would be inefficient in a distributed-memory parallel setting since each calculation would likely involve coordination between



(a) Domains D1 & D2, across two MPI ranks

(b) Local shadow volume of D1

(c) Local shadow volume of D2

(d) Global shadow volume of D1 & D2

Figure 3: A conceptual example of a data set with two domains, D1 & D2, both containing dense regions. Subfigure (a) shows the positions of the dense regions, with a small sphere in D1 and a large cube in D2. Subfigure (b) introduces a light source and shows the shadow volume with respect to D1. Importantly, this conceptual example is overly simple, as it treats the dense volumes as fully opaque; the approach introduced in this paper handles semi-opaque volumes as well. Subfigure (c) shows the shadow volume with respect to only D2, ignoring any shadow effects from D1. Finally, Subfigure (d) shows what our proposed approach would do for this configuration, i.e., the combination of the two. The result is that the left face of the cube in D2 is in shadow due to the effects from D1.

many compute ranks. Instead, our algorithm uses a combination of discretization with pre-processing. The discretization involves representing the spatial volume as a grid of L x W x H vertices. The pre-processing involves calculating the light attenuation at each vertex of the grid. We refer to this combination of grid and light attenuation field as the "shadow volume." This approach allows for an alternate way to calculate light attenuation at each sample along a ray, namely identifying the shadow volume cell a sample lies within and then interpolating a light attenuation value from the eight vertices of the cell.

Our algorithm is exchanging accuracy for potentially improved performance. The accuracy is reduced since our light attenuation calculation is an approximation from the shadow volume as opposed to an exact calculation. In terms of improved performance, the number of grid points in the shadow volume can be significantly smaller than the number of samples along the ray and the mesh resolution of the input data. The reduced size compared to the input data is particularly attractive for volume rendering large data sets on supercomputers, minimizing the computational effort and memory usage associated with the light ray tracing process.

Our approach occurs in two complementary phases: Shadow Volume Generation and Rendering, with the Rendering phase incorporates the shadow volume from the Generation phase. Shadow Volume Generation executes each time the light position changes or each time the transfer function changes. In the case of time-varying data, Shadow Volume Generation would also need to occur for each change in time slice as well. Rendering executes each time the camera position changes. We envision the most common use case will be that a light is placed at a fixed position, and then the camera position is updated repeatedly. In this use case, the shadow volume is calculated only once, and then it is used repeatedly during rendering. This effectively amortizes the cost of shadow volume generation over the rendering. Figure 2 shows how the phases interact visually and also includes sub-phases within the two phases.

3.2 Shadow Volume Generation

Similar to a standard distributed simulation, the input volume is partitioned into domains and distributed across multiple MPI ranks. Each MPI rank generates a shadow volume representing the locally available domain. These ranks then exchange the attenuation values and vertex locations at each of the six faces of the shadow volume with every other MPI rank, to update the local shadow volumes. The update involves accounting for light energy loss due to all other intervening domains that lie between the light source and domain being considered at each MPI rank, as shown in Figure 3.

The Shadow Volume Generation phase divides the creation of the shadow volume into three sub-phases: Local Shadow Volume Generation, Face Exchange, and Global Shadow Volume Generation.

3.2.1 Local Shadow Volume Generation

During the initial phase, each MPI rank creates a shadow volume by tracing light rays solely within its locally available domain of the input volume. This tracing process focuses on the domain's specific contribution to the attenuation of light rays and only considers the vertices of the shadow volume grid within its own domain. As all other domains are distributed across other MPI ranks and thus unavailable, they do not influence the attenuation calculations at this stage. To accomplish this, the light rays originate from the designated light source location and traverse through the input volume, considering only the cells and vertices within the domain's bounds, until they reach the vertices of the shadow volume grid. Along each light ray, equidistant points are sampled within the input domain. An assigned transfer function is utilized at these sample points to obtain the opacity value. The light ray begins with an initial intensity of α_s, typically set to 1, and the attenuation α_n at the nth sample point is then adjusted using the following equation:

$$\alpha_n = \alpha_{n-1} + (1 - \alpha_{n-1}) * \alpha \quad (1)$$

where α_n is the accumulated attenuation due to n light ray sample points, α_{n-1} is the previous value, and α is the opacity at the nth sample point. Note that the attenuation values at the destination points do not account for the rest of the volume distributed at the other MPI ranks and, thus, not the final value.

This shadow volume approach can be used in a serial setting as well, and Figure 4 provides an illustration to clarify how it operates, as well as looking ahead to our distributed-memory approach.

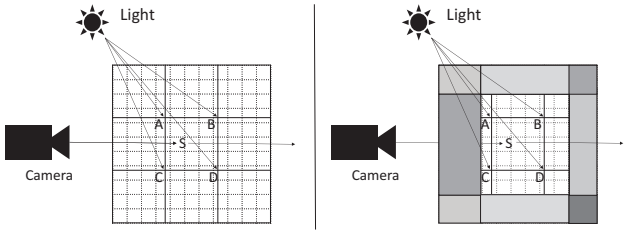


Figure 4: Illustrating our shadow volume approach in both a serial setting (left) and a distributed-memory setting (right). For ease of illustration, the figure is showing a two-dimensional scene. The underlying data set is 12x12, and its grid is rendered with dotted lines. The example shadow volume is 4x4, and its grid is rendered with solid lines. When volume rendering, a ray is cast from the camera through the data set for every pixel in the image. For this example, only one of those rays is shown. There are many samples evaluated along this ray, and this figure shows one of those samples, labeled S. Consider the serial setting on the left. To calculate the shadow volume, a ray is cast for each of its 16 vertices. This figure shows four of these rays and labels their corresponding vertices as A, B, C, and D. The attenuation at each of these vertices is calculated according to Equation 1. For our shadow volume approach, the shadowing for S will be calculated by interpolating the attenuation values from A, B, C, and D. Of course, this figure is showing a two-dimensional scene, and our usage is for three-dimensional scenes. In particular, our approach would actually have a three-dimensional shadow volume, a sample would lie in a voxel and not a pixel, and the interpolation would involve eight values from the shadow volume, not four. Now consider the distributed-memory setting on the right. This figure has nine domains, eight of which are shades of gray, and the perspective of this figure is from the MPI rank that owns the ninth domain, located in the center and not grayed out. This MPI rank will only calculate the attenuation for the vertices within its domain (again A, B, C, and D). Further, it will only calculate the attenuation occurring due to cells within its own domain. This takes less work than the serial setting and gives a different value. That said, the other MPI ranks will calculate their own attenuation values, and the final attenuation values for A, B, C, and D will be updated using those values. In the end, the attenuation values for A, B, C, and D will only differ due to approximation error.

3.2.2 Face Exchange

During the subsequent phase, every MPI rank exchanges computed attenuation values at the faces of the shadow volume grid with all other MPI ranks. To facilitate this exchange, each MPI rank duplicates the vertex locations and corresponding attenuation values on each of the six faces (left, right, bottom, top, front, back) and stores them in two separate arrays. These arrays are then collectively exchanged with all other MPI ranks utilizing an MPI_Allreduce invocation, ensuring comprehensive data sharing across the distributed simulation. Additionally, the ranks exchange information regarding the boundaries of their respective domains.

For a given shadow volume with the resolution L x W x H, the number of attenuation values exchanged per MPI rank can be calculated by:

$$N = 2 * (L * W + W * H + H * L) \quad (2)$$

Additionally, N vertex locations, each with (x, y, z) coordinates, are also exchanged.

Finally, a reviewer pointed out that MPI_Allgather could have been more efficient than MPI_Allreduce. We agree with this, but were unable to make the change and re-run the experiments within the revision period.

3.2.3 Global Shadow Volume Generation

During this phase, each MPI rank updates the attenuation values within its shadow volume by incorporating potential contributions from domains across all other MPI ranks. This is accomplished by tracing light rays in a manner similar to the phase described in 3.2.1. The light rays are tested for intersections against the faces of the other domains, and these intersections are then sorted in front-to-back order based on the distance to the light source. The attenuation values at the intersections are interpolated using the values exchanged as described in 3.2.2 and alpha composited using Equation 1 in the sorted order. The final shadow volume attenuation values are then updated, again using Equation 1.

3.3 Rendering

The Rendering phase involves two sub-phases: Volume Rendering and Compositing. During these sub-phases, the algorithm utilizes the previously generated shadow volume to achieve the desired visual rendering.

3.3.1 Volume Rendering

Upon the completion of shadow volume generation, parallel direct volume rendering occurs simultaneously across all MPI ranks, employing the conventional technique of image order sampling. During this process, camera rays are traced through the input domain, sampling the field at equidistant locations along each ray. The provided transfer function is utilized at each sample location to compute the corresponding RGB and alpha values based on the field value at that specific location. Simultaneously, the shadow volume is sampled at the same location to acquire the corresponding attenuation value. The RGB values are then shaded using a multiplication operation in conjunction with the obtained attenuation value.

3.3.2 Compositing

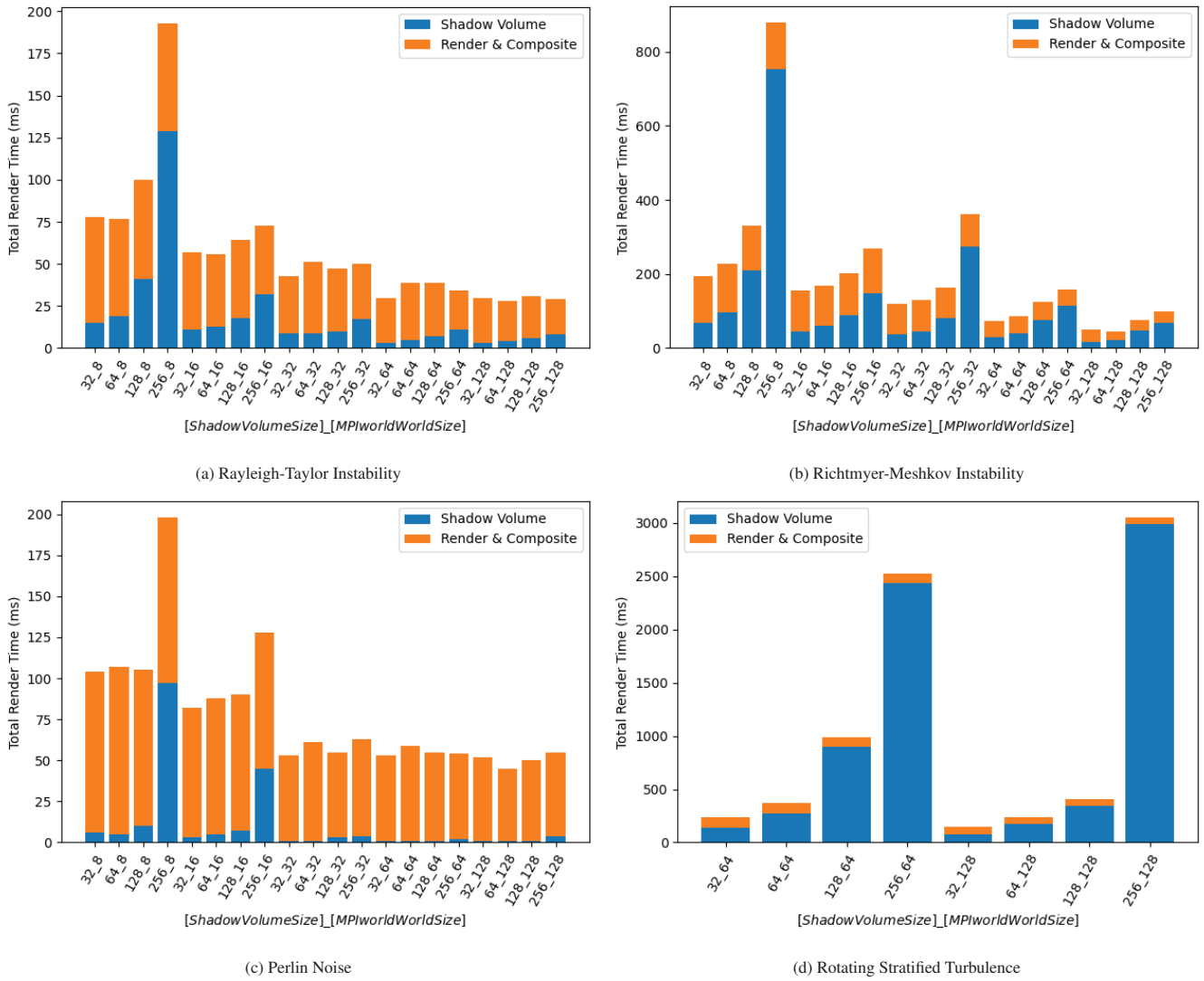
In the final phase, the rendered images obtained through direct volume rendering are composited using the Direct Send compositing algorithm as described by Eilmann and Pajarola [13]. The partial composites that were generated in section 3.3.1 are indexed and sorted based on the distance of each domain's spatial center to the camera location. The final image is divided into tiles and distributed across all the MPI ranks, with each rank performing alpha compositing based on this order and then finally collected at the root MPI rank.

4 Study Overview

Our study is designed to investigate the performance impacts of our algorithm for calculating the attenuation of light through the volume and the use of the attenuation data during traditional volume rendering. This section describes details about the factors used for the study and the architectures used for testing.

4.1 Study Factors

This study has two parts. The first part considers the runtime performance of shadow volume creation using three factors. The second part considers the visual effects from shadow volumes, both compared to other techniques and when the resolution of the shadow volume changes. The factors used for the first part are:



(a) Rayleigh-Taylor Instability

(b) Richtmyer-Meshkov Instability

(c) Perlin Noise

(d) Rotating Stratified Turbulence

Figure 5: A comparison of computation costs of shadow volume generation across four data sets. The X-Axis lists the configuration (shadow volume resolution and number of MPI ranks used), and the Y-Axis represents the runtime in milliseconds. The general trend is that the time taken to generate shadow volumes at lower resolutions is less than the time to render.

- Data sets (4)
- Shadow volume resolutions (4)
- Number of MPI ranks (6)

4.1.1 Data Set

The time to create a shadow volume depends on the complexity of the combination of the data set and transfer function used for the volume rendering. In order to account for that, we test the technique on different data sets that represent typical scenarios. The data sets used in this study are:

- **Rayleigh-Taylor Instability**: This is a time step of a density field in a simulation of the mixing transition in Rayleigh-Taylor instability [11], with a resolution of 1024x1024x1024.
- **Richtmyer-Meshkov Instability**: This is a time step of the entropy field of Richtmyer-Meshkov instability simulation [10], with a resolution of 2048x2048x1920.
- **Perlin Noise**: This is a procedurally generated data set using gradient noise with a resolution of 2048x2048x2048 [34].
- **Rotating Stratified Turbulence**: This is a time step of the temperature field of a direct numerical simulation of rotating stratified turbulence [20], with a resolution of 4096x4096x4096.

4.1.2 Shadow Volume Resolution

The shadow volume resolution determines the number of light rays that are traced during the generation of the volume. We consider the impact of the volume resolution by testing the following resolutions: 32^3, 64^3, 128^3, and 256^3.

4.1.3 Number of MPI Ranks

The shadow volumes are updated by having all MPI ranks exchange border cells of the volume with every other rank. We consider the scalability of our algorithm by increasing the number of MPI ranks, using 8, 16, 32, 64, 128, and 256 ranks. The data set size is fixed regardless of the number of MPI ranks, so scalability evaluations consider strong scaling.

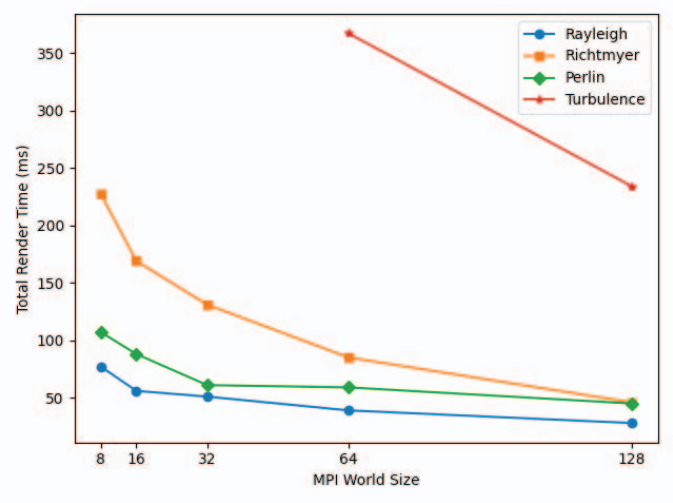


Figure 6: Scaling results of total render time at a constant shadow volume resolution for the four data sets. The X-Axis enumerates the number of MPI ranks utilized, whereas the Y-Axis signifies the elapsed time in milliseconds.

4.2 Implementation Details and Machine Configuration

Our software was implemented using VTK-m [32]. VTK-m has native many-core performance, which has been demonstrated to be effective across many architectures [31]. Most phases of the algorithm (see Section 3) utilize GPUs for shared-memory parallelism, although data exchanges between MPI ranks are performed on the CPU.

All of our experiments were run on the Oak Ridge Summit supercomputer. This machine uses NVIDIA GPUs, so our VTK-m implementation uses a CUDA backend.

5 Results

The results are organized into the two parts of the study: Section 5.1 details the runtime performance of the algorithm, while Section 5.1 considers the visual results of the algorithm.

5.1 Runtime Performance

The performance results for the Rayleigh-Taylor Instability, Richtmyer-Meshkov Instability, Perlin Noise, and Rotating Stratified Turbulence data sets are located in Figure 5. In the initial three data sets, we observe a consistent pattern of the shadow volume generation time escalating exponentially. This aligns with the understanding that every subsequent doubling of the shadow volume resolution on each axis results in an eight-fold surge in the total cell count within the shadow volume grid. For the most part, the runtimes are quite fast, capable of 10 to 20 frames per second. Further, when shadow volume creation is amortized over many frames, which happens when the light source and transfer function do not change, then the effective runtime is the runtime for just rendering and compositing. In this case, the frame rates jump up to as much as 50 frames per second.

The most notable poorly performing experiments involve large grid resolutions in the shadow volume (256^3) with large data sets. For example, the Rotating Stratified Turbulence data set (4096^3 mesh resolution) with the largest shadow volume takes almost 3 seconds to compute the shadow volume. That said, the rendering and compositing time are mostly unaffected by the finer resolution in the shadow volume. Therefore a scenario with repeated rendering with the same shadow volume could still achieve interactive frame rates (10-15 FPS depending on the number of MPI ranks), but frequent changes in light position or transfer function would not be acceptable for interactive use. Of course, our proposed approach has utility beyond interactive use cases since it can be used to create imagery for presentation graphics.

Many factors play a role in runtime, including data set, transfer function, concurrency (i.e., number of MPI ranks), and resolution of the shadow volume. Consider the experiments with the Richtmyer-Meshkov and Perlin Noise, 8 MPI ranks and a 256^3 shadow volume resolution. Although both data sets are 2048^3, shadow volume calculation takes 97ms for Perlin Noise and 753ms for Richtmyer Meshkov. This is because of the interaction between data set and transfer function — Perlin Noise is more opaque and benefits from early ray termination, while Richtmyer Meshkov is more transparent and does not benefit. A secondary concern with the data set is the overall mesh resolution, as clearly, there is more work to do in our "strong scaling" study format when the mesh resolution size grows. With respect to concurrency, the expected behavior of improved runtime when there are more MPI ranks mostly holds, although there are some variations. The most notable outlier is for the Rotating Stratified Turbulence data set, where the 256^3 resolution shadow volume gets slower from 64 MPI ranks to 128 MPI ranks. The smaller shadow volume resolutions for this data set behave more normally. Finally, with respect to the resolution of the shadow volume, the results follow expectations — the higher the resolution, the more time it takes to calculate. That said, rendering and compositing time are largely unaffected by shadow volume resolution.

Figure 6 shows the 64^3 shadow volume results from Figure 5 in a different way, demonstrating that total render time settles asymptotically as the number of MPI ranks used increases. This is because there are fewer cells per rank to process when the number of MPI ranks increases and the input volume remains fixed.

5.2 Visual Performance

This part of the study considers visual performance. One aspect of this part of the study is visual comparison between shadow volumes with volume renderings using the Phong shading model or no shading at all. Another aspect is considering the effect of shadow volume resolution.

5.2.1 Comparing With Phong Shading and No Shading

Figure 7 shows the data sets used in the performance study, and Figure 8 shows some additional data sets. Both figures consider the same scene (camera, transfer function, data set) with different shading — no shading in the left column, our shadow volume approach in the middle column, and Phong shading in the right column. The Phong shading model utilizes gradient estimation with central differences to determine the normal at each sample point for lighting calculations.

The images rendered using shadow volumes offer several distinct advantages over those generated using Phong shading, particularly in the context of distinguishing areas of interest within the volume across most of the data sets. Moreover, there is a notable improvement in accurately following the contour and shape of these regions within the volume. Additionally, regions cast in shadow provide a heightened sense of depth perception, as our visual system relies on shadows to determine spatial relationships between objects. Importantly, the shadows introduced by the shadow volumes do not significantly hinder the visibility of various parts of the volume by causing excessive occlusion.

However, it is worth mentioning that the use of the Phong shading model can lead to a misrepresentation of the image, especially in scenarios where the data exhibits clear, distinct patterns. In such cases, the application of shadow volumes may prove particularly beneficial, especially when compared to smaller data sets. The absence of clutter and the ease of identifying the layout of the image contribute to the enhanced perceptual experience offered by the

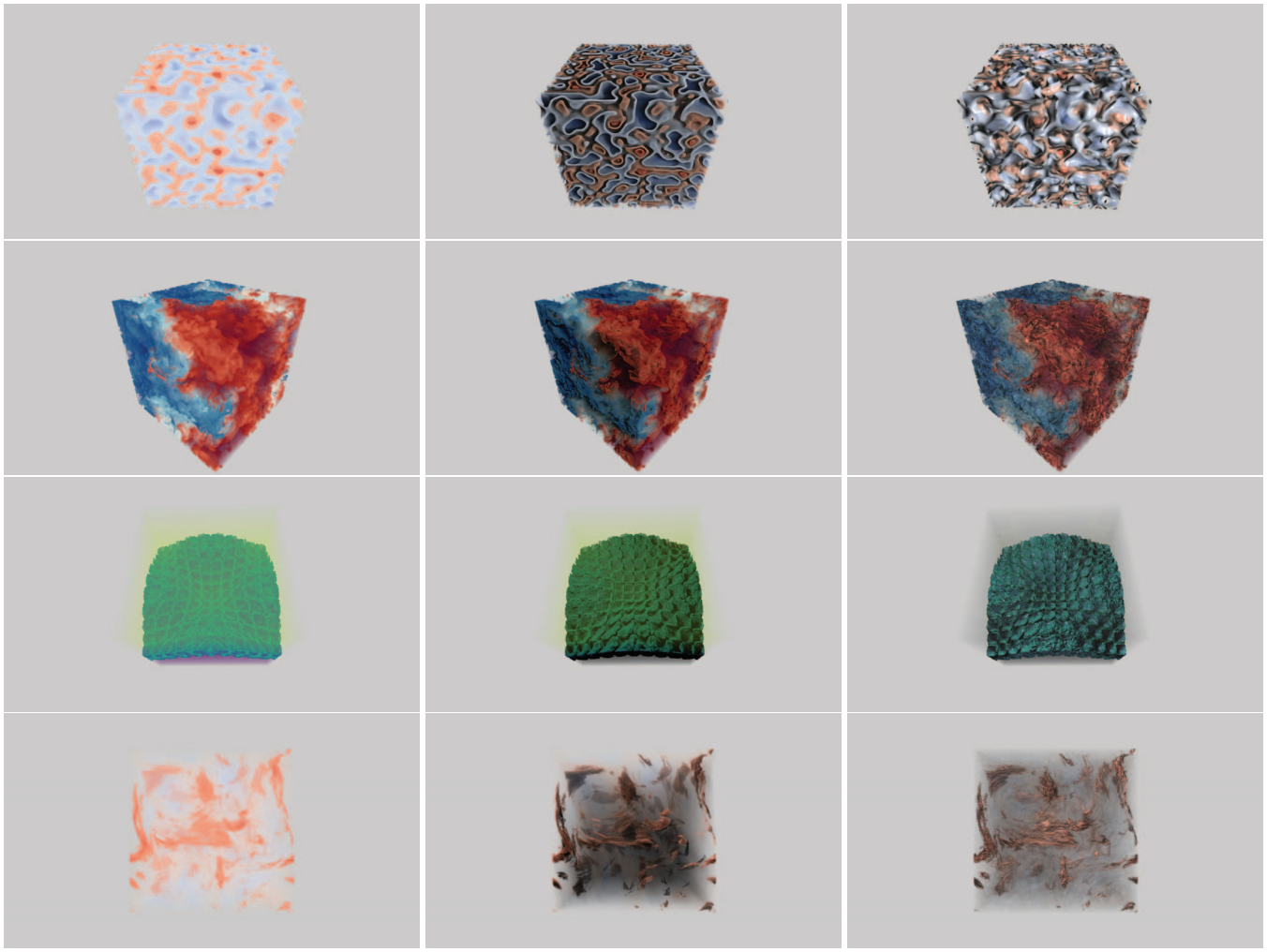


Figure 7: Images informing visual performance of our approach. The images are arranged in a 4x3 layout, with columns representing different rendering approaches and rows representing different data sets. For the columns, the left column is direct volume rendering (i.e., no lighting), the middle column is our shadow volume approach, and the right column uses Phong shading. For the rows, the first row shows the Perlin data set, the second row shows a Rayleigh-Taylor data set, the third row shows a Richtmyer-Meshkov data set, and the fourth row shows the Rotating Stratified Turbulence data set

shadow volumes. These advantages do not have a big impact on the images rendered for Richtmyer-Meshkov data set, suggesting that the complexity introduced by the shadow volumes can be distracting when the underlying data itself possesses well-defined patterns, as the shadow volumes may not add significant structural information.

In summary, the use of shadow volumes in volume rendering provides notable advantages over the standard direct rendering or the one using the Phong shading model. The ability to distinguish areas of interest, follow contour and shape, and leverage the depth cues offered by shadows contributes to an improved perceptual understanding of the volumetric data. However, it is essential to consider the nature of the data and the potential for distracting complexity introduced by the shadow volumes, particularly when the data exhibits clear, distinct patterns.

5.2.2 The Effect of Shadow Volume Resolution

Figure 9 shows the impact of shadow volume resolution on the quality of the shadows generated. This figure shows both renderings that inform the difference and metrics (PSNR and SSIM) that quantify the differences. It shows that, 32^3 shadow volumes, a resolution significantly lower than that of the input volume, can still improve depth perception (with the benefit of being much faster to compute). However, the examples show that it does not capture the finer details of the volume. It is also important to note that higher resolutions can cause the image to appear darker, as each cell of the shadow volume represents light attenuation spread over a comparatively larger area of the input. This effect is particularly evident at the boundaries of the input volume, as the light attenuation values at the faces start at 0.0 and accumulate deeper into the volume.

6 Conclusion and Future Work

This work introduced a new approach for calculating shadow volumes when volume rendering large data sets in a distributed-memory setting. Our results focused on two aspects: runtime performance and visual performance. Runtime was generally good, although high-resolution shadow volumes led to penalties of several seconds in some cases. Further, visual performance was generally good, with images using our technique often looking for more informative than approaches with no lighting and less cluttered than a traditional lighting approach using Phong shading. Finally, our approach is simple enough that it can be easily adopted in practice. In short, we

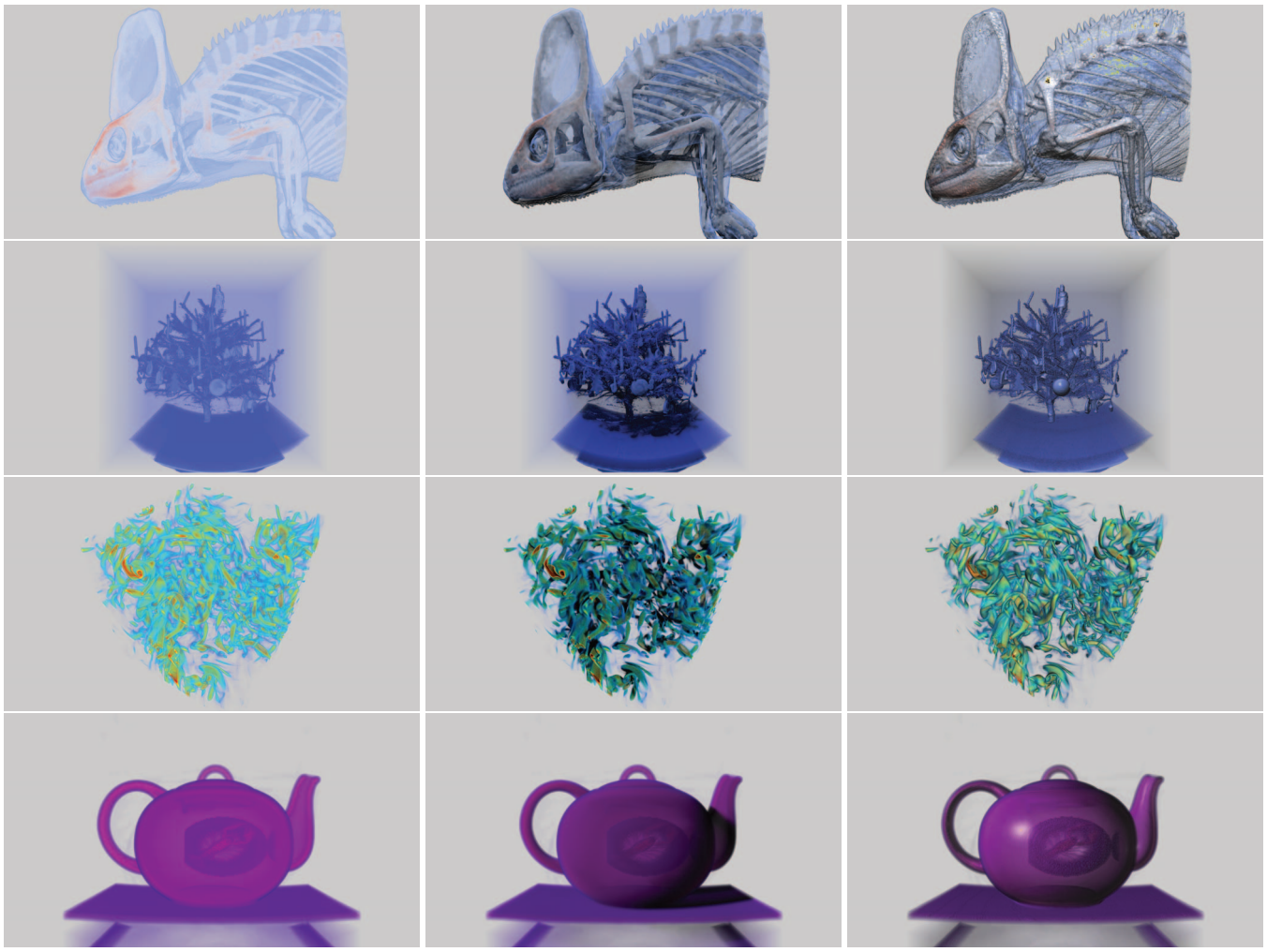
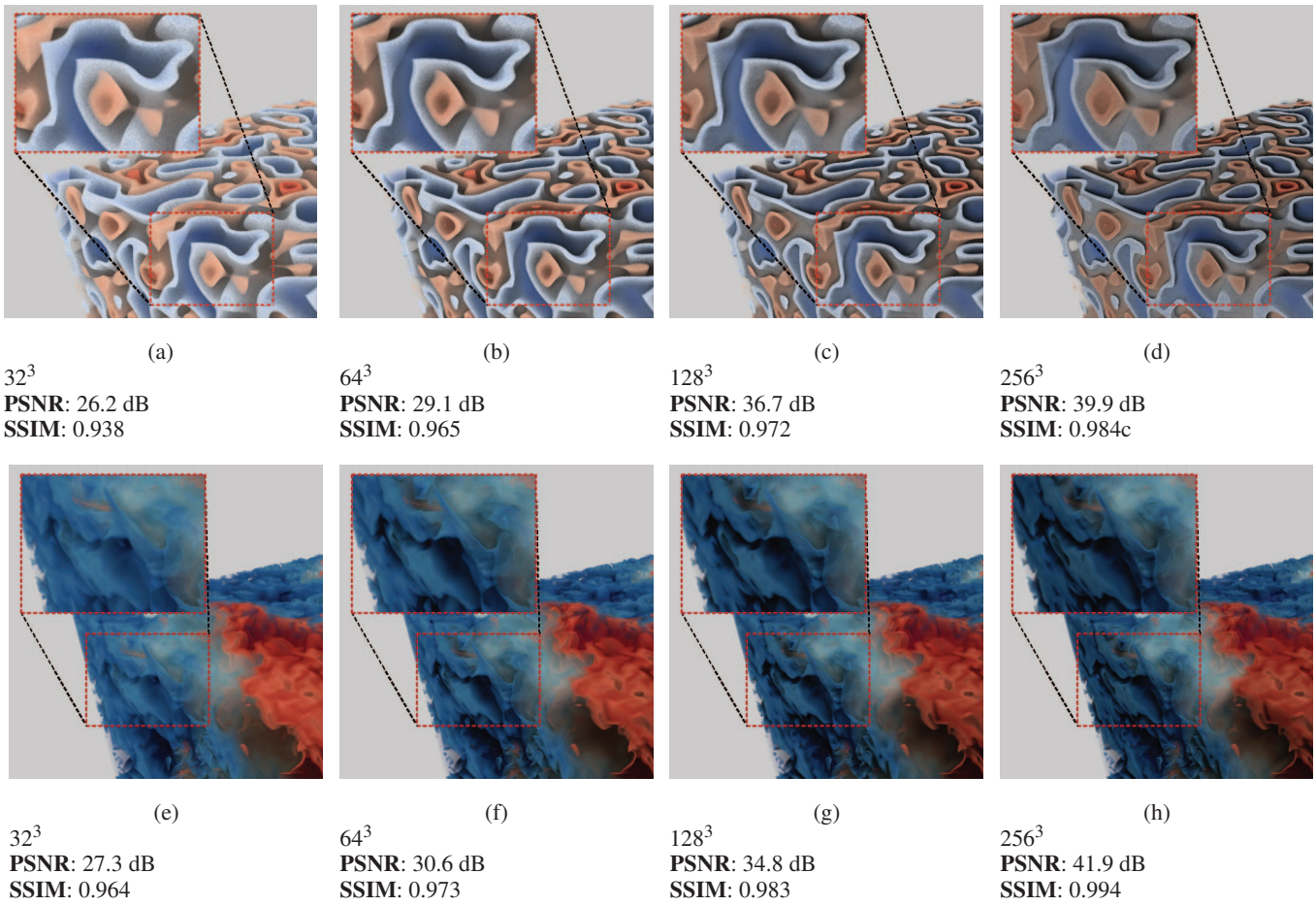


Figure 8: Additional images informing visual performance of our approach, complementing Figure 7. Where the data sets in Figure 7 also appear in our performance study, these data sets are smaller and only used for visual performance. Like with Figure 7, the images are arranged in a 4x3 layout, with columns representing different rendering approaches and rows representing different data sets. Once again, the left column is direct volume rendering (i.e., no lighting), the middle column is our shadow volume approach, and the right column uses Phong shading. For the rows, the first row shows CT scan of a chameleon [27], the second row shows a model for a Christmas tree [19], the third row shows an isotropic turbulence simulation, and the fourth row shows the Boston teapot.

believe this approach is worthy of inclusion in production software since it has smaller penalties for rendering time and developer time and significant benefit in terms of improved graphics. Our implementation in VTK-m is a significant step in this direction, since VTK-m is incorporated into both ParaView [1] and VisIt [7].

We feel this direction suggests many lines of inquiry for future work. First, our algorithm assumes the data can be decomposed into blocks with axis-aligned faces. Unstructured meshes and AMR meshes do not currently fit with this algorithm and would have to be resampled onto a block-decomposed regular grid. Second, we have traded one type of lighting (Phong) for another (shadowing). Another possibility is to use both, which would more closely mirror what happens in the real world. That said, initial attempts retained the cluttered look seen in the Phong-only images in this paper. Future work would need to consider how these two lighting effects should be combined in a more intelligent manner. The current work only supports shadowing due to a point light source, which can be limiting for volumes that are highly dense and may cause large sections to be self-shadowed. Adding support for other kinds of light sources, like area lights, can help in better illumination in these cases. Next, this work used limited shading comparators (Phong shading and no lighting). Comparing our approach with other volume rendering approaches that include shading (such as Exposure Render [21]) would be useful. A limitation of the current approach is that algorithm is acutely sensitive to any changes in the input volume and requires the shadow volume to be recreated. This can be a hindrance to using it for dynamic time-varying data sets. Future work could enhance the algorithm to iteratively update the shadow volume. Finally, this work only considered shadow volumes up to 256^3. We found that experiments with low numbers of MPI ranks ran out of memory. More experiments exploring tradeoffs between cost and picture quality for larger shadow volumes would be worthwhile. Of course, going to higher resolutions may result in unacceptable runtimes — a grid of 256^3 has 16.7M vertices and calculating the attenuation for each of these vertices requires significant computation.



(a) 32^3 **PSNR**: 26.2 dB **SSIM**: 0.938

(b) 64^3 **PSNR**: 29.1 dB **SSIM**: 0.965

(c) 128^3 **PSNR**: 36.7 dB **SSIM**: 0.972

(d) 256^3 **PSNR**: 39.9 dB **SSIM**: 0.984c

(e) 32^3 **PSNR**: 27.3 dB **SSIM**: 0.964

(f) 64^3 **PSNR**: 30.6 dB **SSIM**: 0.973

(g) 128^3 **PSNR**: 34.8 dB **SSIM**: 0.983

(h) 256^3 **PSNR**: 41.9 dB **SSIM**: 0.994

Figure 9: A two-by-four configuration of Perlin Noise and Rayleigh-Taylor data sets. In each row, all four renderings have the same data resolution, the same transfer function, and the same camera position. The only difference between them is the resolution of the shadow volume, with the specific resolution for each in a sub-figure caption. All these images are compared against an image generated using a shadow volume resolution of 512^3, which is considered the reference image, and their PSNR and SSIM metrics are reported as quality metrics. Further, the upper left of each figure shows a close-up of a specific region within the volume. This close-up shows that the shadow quality improves somewhat as the resolution improves. That said, the main takeaway from these images is not that the shadow quality improves, but rather that the images get brighter with lower resolution. This is because the shadow volume is being interpolated over a relatively larger region, and the shadow value at each face is 0.0, i.e., the lack of shadowing on grid points along the face is interpolated deeper into the volume.

Acknowledgments

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

References

[1] J. Ahrens, B. Geveci, and C. Law. ParaView: An End-User Tool for Large Data Visualization. In *Visualization Handbook*. Elesvier, 2005.

[2] M. Balsa Rodríguez, E. Gobbetti, J. A. Iglesias Guitian, M. Makhinya, F. Marton, R. Pajarola, and S. K. Suter. State-of-the-Art in Compressed GPU-Based Direct Volume Rendering. In *Computer Graphics Forum*, vol. 33, pp. 77–100. Wiley Online Library, Sept. 2014.

[3] U. Behrens and R. Ratering. Adding Shadows To A Texture-Based Volume Renderer. In *IEEE Symposium on Volume Visualization*, pp. 39–46, Oct. 1998.

[4] J. Beyer, M. Hadwiger, and H. Pfister. State-of-the-Art in GPU-Based Large-scale Volume Visualization. In *Computer Graphics Forum*, vol. 34, pp. 13–37, Dec. 2015.

[5] R. Binyahib, T. Peterka, M. Larsen, K.-L. Ma, and H. Childs. A Scalable Hybrid Scheme for Ray-Casting of Unstructured Volume Data. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 25(7):2349–2361, July 2019.

[6] B. Cabral, N. Cam, and J. Foran. Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware. In *Proceedings of the 1994 Symposium on Volume Visualization*, p. 91–98. Association for Computing Machinery, Oct. 1994.

[7] H. Childs, E. Brugger, B. Whitlock, J. Meredith, S. Ahern, K. Bonnell, M. Miller, G. H. Weber, C. Harrison, D. Pugmire, T. Fogal, C. Garth, A. Sanderson, E. W. Bethel, M. Durant, D. Camp, J. M. Favre, O. Rübel, P. Navrátil, M. Wheeler, P. Selby, and F. Vivodtzev. VisIt: An End-User Tool for Visualizing and Analyzing Very Large Data. In *Proceedings of SciDAC 2011*. Denver, CO, July 2011.

[8] H. Childs, M. Duchaineau, and K.-L. Ma. A Scalable, Hybrid Scheme for Volume Rendering Massive Data Sets. In *Proceedings of Eurographics Symposium on Parallel Graphics and Visualization (EGPGV)*, pp. 153–162. Braga, Portugal, May 2006.

[9] H. Childs, D. Pugmire, S. Ahern, B. Whitlock, M. Howison, Prabhat,

G. Weber, and E. W. Bethel. Extreme Scaling of Production Visualization Software on Diverse Architectures. *IEEE Computer Graphics and Applications (CG&A)*, 30(3):22–31, May/June 2010.

[10] R. Cohen, W. Dannevik, A. Dimits, D. Eliason, A. Mirin, Y. Zhou, D. Porter, and P. Woodward. Three-Dimensional Simulation of a Richtmyer-Meshkov Instability with A Two-scale Initial Perturbation. *Physics of Fluids*, 14:3692–3709, Oct. 2002.

[11] A. W. Cook, W. Cabot, and P. L. Miller. The Mixing Transition in Rayleigh-Taylor Instability. *Journal of Fluid Mechanics*, 511:333–362, July 2004.

[12] B. Domonkos and B. Csébfalvi. Interactive Distributed Translucent Volume Rendering. In *The 15th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, pp. 153 – 160, Feb 2007.

[13] S. Eilemann and R. Pajarola. Direct Send Compositing for Parallel Sort-Last Rendering. In J. M. Favre, L. P. Santos, and D. Reiners, eds., *Eurographics Symposium on Parallel Graphics and Visualization*. The Eurographics Association, May 2007.

[14] F. Erol, S. Eilemann, and R. Pajarola. Cross-Segment Load Balancing in Parallel Rendering. In *EGPGV@ Eurographics*, pp. 41–50, Apr. 2011.

[15] T. Fogal, H. Childs, S. Shankar, J. Krüger, R. D. Bergeron, and P. Hatcher. Large Data Visualization on Distributed Memory Multi-GPU Clusters. In *Proceedings of High Performance Graphics (HPG)*, pp. 57–66. Saarbrucken, Germany, June 2010.

[16] M. Hadwiger, A. Kratz, C. Sigg, and K. Bühler. GPU-Accelerated Deep Shadow Maps for Direct Volume Rendering. In *Proceedings of the 21st ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware*, GH '06, p. 49–52. New York, NY, USA, Sept. 2006.

[17] M. Howison, E. W. Bethel, and H. Childs. MPI-hybrid Parallelism for Volume Rendering on Large, Multi-core Systems. In *Eurographics Symposium on Parallel Graphics and Visualization (EGPGV)*, pp. 1–10. Norrköping, Sweden, Apr. 2010.

[18] M. Howison, E. W. Bethel, and H. Childs. Hybrid Parallelism for Volume Rendering on Large-, Multi-, and Many-Core Systems. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 18(1):17–29, Jan. 2012.

[19] A. Kanitsar, T. Theußl, L. Mroz, M. Šrámek, A. V. Bartrolí, B. Csébfalvi, J. Hladůvka, D. Fleischmann, M. Knapp, R. Wegenkittl, P. Felkel, S. Röttger, S. Guthe, W. Purgathofer, and M. E. Gröller. Christmas Tree Case Study: Computed Tomography as a Tool for Mastering Complex Real World Objects with Applications in Computer Graphics. In *IEEE Visualization, 2002. VIS 2002.*, pp. 489–492, 2002.

[20] K. Kanov, R. Burns, C. Lalescu, and G. Eyink. The Johns Hopkins Turbulence Databases: An Open Simulation Laboratory for Turbulence Research. *Computing in Science and Engineering*, 17(5):10–17, Sept./Oct. 2015.

[21] T. Kroes, F. H. Post, and C. P. Botha. Exposure Render: An Interactive Photo-Realistic Volume Rendering Framework. *PloS one*, 7(7):e38586, July 2012.

[22] E. P. Lafortune and Y. D. Willems. Rendering Participating Media with Bidirectional Path Tracing. In *Proceedings of the Eurographics Workshop on Rendering Techniques '96*, pp. 91–100. Springer, Dec. 1996.

[23] T. Lokovic and E. Veach. Deep Shadow Maps. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 385–392, July 2000.

[24] K.-L. Ma. Parallel Volume Ray-Casting for Unstructured-Grid Data on Distributed-Memory Architectures. In *Proceedings of the IEEE Symposium on Parallel Rendering*, pp. 23–30, Dec. 1995.

[25] K.-L. Ma and T. W. Crockett. A Scalable Parallel Cell-Projection VolumeRrendering Algorithm for Three-Dimensional Unstructured Data. In *Proceedings IEEE Symposium on Parallel Rendering*, pp. 95–104, Oct. 1997.

[26] K.-L. Ma, J. S. Painter, C. D. Hansen, and M. F. Krogh. Parallel Volume Rendering Using Binary-Swap Compositing. *IEEE Computer Graphics and Applications*, 14(4):59–68, July 1994.

[27] J. Maisano. Chamaeleo calyptratus, Aug. 2003. (On-line), Digital Morphology.

[28] S. Marchesin, C. Mongenet, J.-M. Dischler, et al. Dynamic Load Balancing for Parallel Volume Rendering. In *Proceedings of the 6th Eurographics Conference on Parallel Graphics and Visualization (EGPGV)*, pp. 43–50, May 2006.

[29] N. Max. Optical Models for Direct Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, June 1995.

[30] C. Montani, R. Perego, and R. Scopigno. Parallel Rendering of Volumetric Data Set on Distributed-Memory Architectures. *Concurrency: Practice and Experience*, 5(2):153–167, Apr. 1993.

[31] K. Moreland, R. Maynard, D. Pugmire, A. Yenpure, A. Vacanti, M. Larsen, and H. Childs. Minimizing Development Costs for Efficient Many-Core Visualization Using MCD^3. *Parallel Computing*, 108:102834, Dec. 2021.

[32] K. Moreland, C. Sewell, W. Usher, L. Lo, J. Meredith, D. Pugmire, J. Kress, H. Schroots, K.-L. Ma, H. Childs, M. Larsen, C.-M. Chen, R. Maynard, and B. Geveci. VTK-m: Accelerating the Visualization Toolkit for Massively Threaded Architectures. *IEEE Computer Graphics and Applications (CG&A)*, 36(3):48–58, May/June 2016.

[33] C. Müller, M. Strengert, and T. Ertl. Optimized Volume Raycasting for Graphics-Hardware-Based Cluster Systems. In *Proceedings of the 6th Eurographics Conference on Parallel Graphics and Visualization*, pp. 59–66. Eurographics Association, May 2006.

[34] K. Perlin. Improving Noise. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 681–682, July 2002.

[35] T. Peterka, D. Goodell, R. Ross, H.-W. Shen, and R. Thakur. A Configurable Algorithm for Parallel Image-Compositing Applications. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, pp. 1–10, Nov. 2009.

[36] T. Peterka, R. Ross, H. Yu, K.-L. Ma, W. Kendall, and J. Huang. Assessing Improvements in the Parallel Volume Rendering Pipeline at Large Scale. In *Proceedings of SC 08 Ultrascale Visualization Workshop*. Austin TX, Nov. 2008.

[37] T. Peterka, H. Yu, R. Ross, and K.-L. Ma. Parallel Volume Rendering on the IBM Blue Gene/P. In *Proceedings of Eurographics Parallel Graphics and Visualization Symposium 2008*. Crete, Greece, Apr. 2008.

[38] T. Peterka, H. Yu, R. Ross, K.-L. Ma, and R. Latham. End-to-End Study of Parallel Volume Rendering on the IBM Blue Gene/P. In *Proceedings of ICPP 09*. Vienna, Austria, Sept. 2009.

[39] R. Samanta, J. Zheng, T. Funkhouser, K. Li, and J. P. Singh. Load Balancing for Multi-Projector Rendering Systems. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pp. 107–116, July 1999.

[40] D. Steiner, E. G. Paredes, S. Eilemann, and R. Pajarola. Dynamic Work Packages in Parallel Rendering. In *Proceedings of the 16th Eurographics Symposium on Parallel Graphics and Visualization*, pp. 89–98, June 2016.

[41] M. Strengert, M. Magallón, D. Weiskopf, S. Guthe, and T. Ertl. Hierarchical Visualization and Compression of Large Volume Datasets Using GPU Clusters. In *Proceedings of the 5th Eurographics Conference on Parallel Graphics and Visualization*, vol. 4, pp. 41–48, June 2004.

[42] L. Williams. Casting Curved Shadows on Curved Surfaces. In *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 270–274, Aug. 1978.

[43] 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*, pp. 1–11. IEEE, Nov. 2008.