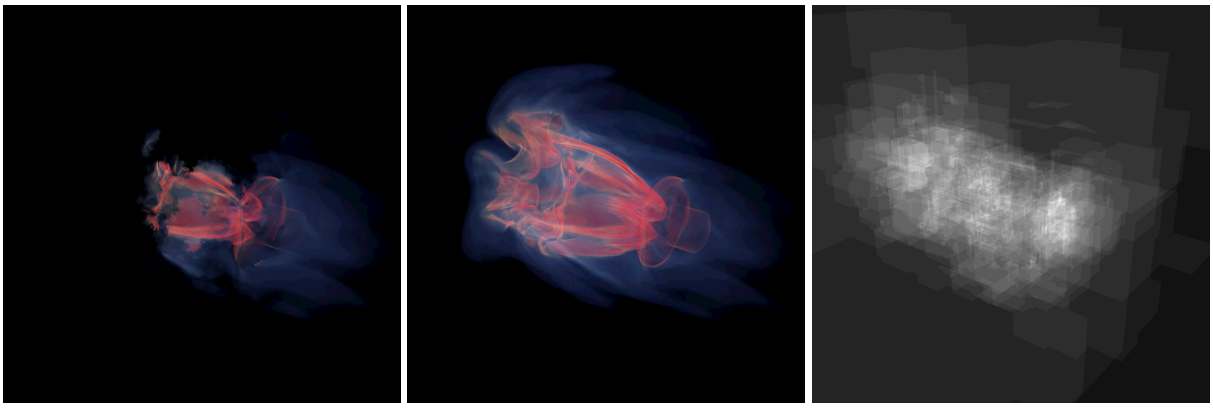
This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2025.3567124

Approximate Puzzlepiece Compositing

Xuan Huang, Will Usher and Valerio Pascucci



(a) 16 Distributed Domains (15.3s/frame) (b) All 72 Distrib. Domains (15.6s/frame) (c) Segment Density Heatmap (max=28)

Fig. 1: Large-scale moment-based order-independent (MBOIT) distributed transparency rendering with the FUN3D Mars Lander A/143M dataset, consisting of 72 subdomains and 798M elements. Figures (a) and (b) are rendered at 2560×2560 using TACC Frontera Intel Xeon Platinum 8280 ("Cascade Lake") nodes with 192GB memory. (c) A heatmap of the per-pixel segment counts with a range of [0, 28]. The segment lists must be individually sorted and blended in sort-last compositing due to the overlapping boundaries of data on the ranks, resulting in large data transfers and bottlenecks. Our approach ensures a constant, fixed, and small amount of communication for compositing arbitrary data distributions.

Abstract—The increasing demand for larger and higher fidelity simulations has made Adaptive Mesh Refinement (AMR) and unstructured mesh techniques essential to focus compute effort and memory cost on just the areas of interest in the simulation domain. The distribution of these meshes over the compute nodes is often determined by balancing compute, memory, and network costs, leading to distributions with jagged nonconvex boundaries that fit together much like puzzle pieces. It is expensive, and sometimes impossible, to re-partition the data posing a challenge for in situ and post hoc visualization as the data cannot be rendered using standard sort-last compositing techniques that require a convex and disjoint data partitioning. We present a new distributed volume rendering and compositing algorithm, Approximate Puzzlepiece Compositing, that enables fast and high-accuracy in-place rendering of AMR and unstructured meshes. Our approach builds on Moment-Based Ordered-Independent Transparency to achieve a scalable, order-independent compositing algorithm that requires little communication and does not impose requirements on the data partitioning. We evaluate the image quality and scalability of our approach on synthetic data and two large-scale unstructured meshes on HPC systems by comparing to state-of-the-art sort-last compositing techniques, highlighting our approach's minimal overhead at higher core counts. We demonstrate that Approximate Puzzlepiece Compositing provides a scalable, high-performance, and high-quality distributed rendering approach applicable to the complex data distributions encountered in large-scale CFD simulations.

Index Terms—Volume Rendering, Distributed Rendering, Compositing, Order-Independent Transparency

1 Introduction

Volume visualization is a crucial part of the analysis pipeline, and is used by domain scientists to analyze their data in fields ranging from biology and medicine to engineering and geoscience. Although continuing advances in data acquisition, simulation, and computation power provide ever more accurate data this comes at the cost of single-node computation and storage space. As scientists solve ever larger and more complex problems, it becomes necessary to render the data in situ because it can no longer be saved out frequently enough. High performance computing systems offer massive amounts of computing power for simulation and visualization; however, scalable distributed volume rendering is non-trivial for simulations with non-convex data distributions. Such distributions are common in unstructured or octree AMR mesh simulations, e.g., using fill-reducing partitioners such as ParMETIS [13] or p4est's [5] Morton index partitioning.

Although sort-last data-parallel rendering is a standard approach for distributed volume rendering, it requires that the data partition on each node be convex and disjoint to ensure each node can produce a single depth-sortable partial image for compositing. This restriction is typically not satisfied in distributed simulations on unstructured [1, 1, 8, 25, 26] or octree AMR [2, 5] grids. Elements in such simulations are distributed to optimize compute and networking costs, e.g., through fill reducting orderings [13], resulting in data partitions with jagged boundaries where the bounding box of each rank's partition overlaps somewhat those of its neighbors (Figure 2b). The partial images from each rank can no longer be sorted since the jagged boundaries cause them to overlap in depth along the view axis. Although sort-last compositing could be extended to produce and blend individual fragments for each nonoverlapping segment of the volume on each rank, such an approach would be prohibitively expensive in both bandwidth and computation costs.

We propose a novel, scalable compositing approach that works on arbitrary data distributions, does not require sorting, and minimizes

- *Xuan Huang and Valerio Pascucci are with the SCI Institute at the University of Utah. E-mail: xuanhuang@sci.utah.edu*
- *Will Usher is with Luminary Cloud.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2025.3567124

network communication to two parallel reductions. We achieve this by adapting an order-independent transparency technique, Moment-Based Order-Independent Transparency [23], to data-parallel compositing. Our final compositing pipeline consists of two stages, a moments generation phase, where communication consists of a single all-reduce add, and a final gather stage where the final image is produced through a reduce add to the display rank. We describe our proposed Approximate Puzzlepiece Compositing (APC) pipeline and evaluate its performance on the NASA FUN3D Mars Lander datasets [1], the NASA Exajet dataset [21], and a worst-case synthetic scaling test case on the TACC Frontera HPC system. We compare our proposed method to sort-last compositing on segments as the state of the art and demonstrate that APC is fast, applicable to any large-scale volumetric data, and produces accurate images. Our contributions are:

- A highly scalable two-stage distributed volume rendering method that imposes no restrictions on the data distribution;
- Evaluation of the image quality of Moment-Based Order-Independent Transparency for compositing distributed volumetric data; and
- A thorough performance study on a worst-case synthetic test volume and two real-world large-scale datasets, the FUN3D and the Exajet, demonstrating our method's minimal communication costs and high scalability.

2 Related Work

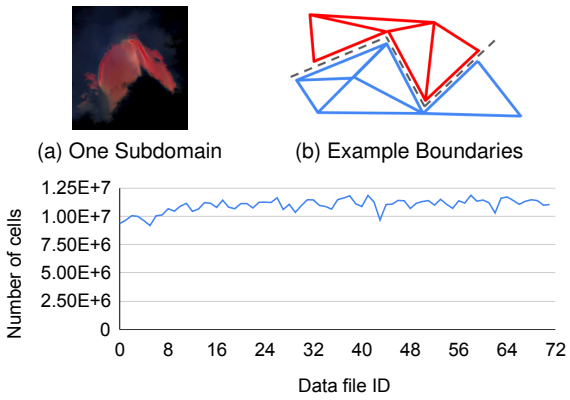
We review related work in distributed volume rendering and order-independent rendering of transparent objects. Sort-last compositing is a widely studied technique for distributed volume rendering (Section 2.1) The problem of real-time rendering of complex transparent objects is frequently encountered in real-time graphics and games, and has been the subject of extensive study (Section 2.2). We further review order-independent transparency methods and their potential application to distributed volume rendering in Section 3.

2.1 Distributed Volume Rendering

A common way to render large volumes is to parallelize rendering over a cluster of machines in image-, object-, or hybrid-order. Distributing the rendering workload allows accelerating rendering or rendering data sets that cannot fit on a single machine. Each machine now has just a subpiece of the data, and independently produces a partial image of whole dataset. The fundamental scaling challenge in distributed rendering is combining these partial images into a final single image of the entire dataset [4, 11, 12, 14, 19, 20, 27, 33, 35–37].

Based on the Porter and Duff over operation [31], Molnar et al. [20] summarized a theoretical model of distributed rendering based on where the sorting happens: sort-first, sort-middle, and sort-last. Sort-last is a practical and scalable object-order approach used for large volumes where the data is distributed and rendered fragments are exchanged between each node [4, 11, 12, 14, 27, 33, 35, 36]. Hsu [12] proposed segmented raycasting, where data is distributed and fragments are sent to the node that owns a given pixel. Tree-based methods, such as binary swap [14] and Radix-k [27], introduce more structured and scalable fragment exchange patterns to improve parallelism. The IceT library [22] provides practical implementations of a number of sort-last algorithms and is widely used in the scientific visualization community. OSPRay [34] provides a distributed rendering facility that can integrate with IceT [35] or leverage its own scalable Distributed FrameBuffer [33] to allow for more flexible data distributions. However, these techniques are all based on sort-last compositing and require that the partial image produced on each node can be uniquely ordered in depth relative to other node's partial images so that they can be composited to produce the final image. This requirement does not hold for the data distributions we consider in this paper.

Layer and deeper fragment buffer approaches have been proposed to handle some level of depth order overlap. The A-buffer algorithm enables unordered rendering by storing and sorting fragments afterwards [6]. Then, to avoid sorting an arbitrarily long list, the k-buffer allows merging extra pixel segments heuristically [3]. However, even though these layer-based approaches accumulate only a fixed number of layers, the compositing pipeline still needs to be executed in order.



(c) Number of Cells for Each Subdomain.

Fig. 2: The FUN3D data. Note the uneven and jagged data boundary of an individual subdomain's data in (a). An illustration of the volume boundaries of two subdomains is shown in (b).

2.2 Order-Independent Transparency for Rendering

Enabling order-independent transparency (OIT) the need to sort fragments to produce the final image, which is especially costly in modern highly parallel renderers. The key is to find a reasonable heuristic that balances approximation accuracy with any additional computational overhead. Developing such techniques has been the focus of a substantial body of work in real-time rendering. Early OIT work on depth peeling [10] utilized multiple rendering passes to render and peel away layers of surfaces in depth order without explicitly sorting geometry before rendering. Although depth peeling accurately resolves the transparency of objects, its performance is highly dependent on scene complexity, resulting in unpredictable compute and memory costs.

To reduce the impact of scene complexity, OIT works heuristically merge or discard fragments [9, 15–18, 23, 32] to achieve fast, scene-independent OIT, at the cost of image quality. However, OIT methods are primarily targeted at real-time applications such as games, incorporating assumptions about limited depth-complexity and smoothness that do not hold for visualization applications. Scenes with high depth or color complexity, as can be common in visualization, break these assumptions and are especially challenging for such methods. Subsampling based approaches such as hybrid transparency [15], which tries to pick the k most important colors, and stochastic transparency [9], which stochastically discards fragments, can encounter missing surfaces. Single-layer heuristic techniques, such as phenomenological transparency [17], sort-independent alpha blending [18], and weighted-blended OIT [16] operate in a fixed memory budget can produce incorrect occlusion and other visual artifacts. Although these methods achieve fast and scene-independent OIT, the artifacts introduced can make them less suited to visualization applications, where accuracy is more important compared to real-time applications such as games.

In this work, we leverage Moment-Based Order-Independent-Transparency (MBOIT) [23] as an off-the-shelf and efficient solution for OIT that is well suited to use in a distributed visualization environment due to its low data requirements and high image quality. MBOIT is based on moment-based shadow map approximation [28, 29], which offers compact, filterable, closed-form representation that is able to capture sparse signals accurately. Similar to OIT through Fourier opacity mapping, which views transmittance as a function of depth in logarithmic space to enable an additive accumulation, this approximation provides a more continuous and faithful representation of complex transparent scenes [23]. MBOIT requires little data to be transferred, does not require sorting or redistributing the mesh data, and provides high image quality, making it well suited to scalable rendering of unstructured and AMR datasets. Figure 3 compares rendering a synthetic volume using MBOIT with Hybrid Transparency and Weighted Blended OIT to illustrate MBOIT's improved image quality.

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2025.3567124

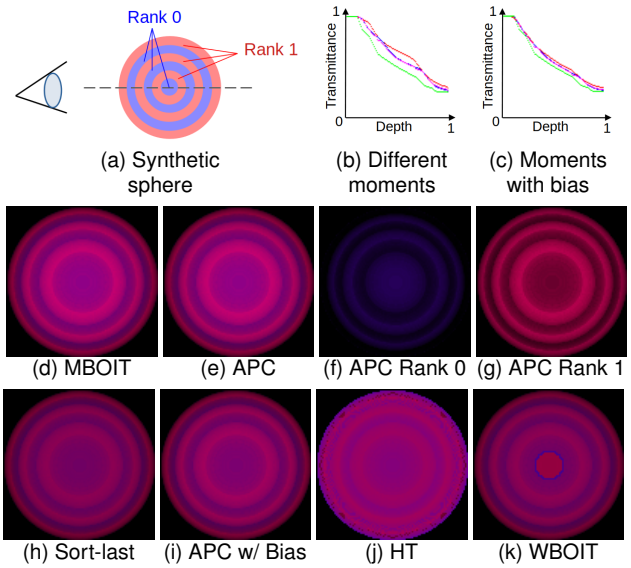


Fig. 3: A comparison of approximate OIT techniques on a synthetic red and blue concentric circles example. This configuration produces sharp changes in color, challenging approximate OIT methods. (b,c) compare different MBOIT configurations with sort-last (green line): 4 power moments (red), 6 power moments (blue) and trigonometric moments (purple). We observe that APC achieves the same rendering quality as single-node MBOIT, providing a high-quality approximation. This is in contrast to the color artifacts of Hybrid Transparency (j) or occlusion errors from Weight-blended OIT (k). Furthermore, with 4 power moments and a bias (c), the APC image closely approximates the ground truth sort-last. The respective image similarity measurements are (h) vs (i): SSIM=0.99, MSE=38.18, PSNR=32.34 (h) vs (j) SSIM=0.87, MSE=618.76, PSNR=20.24 (h) vs (k) SSIM=0.98, MSE=21.55, PSNR=34.82

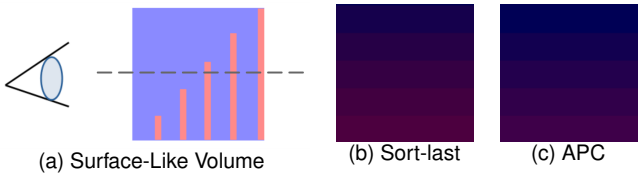


Fig. 4: A synthetic case where APC exhibits noticeable image quality loss. With thin, near-opaque red structures in the middle of a blue volume, we can see that MBOIT does not handle abrupt transmittance changes well such that (c) looks more blended, i.e., more purple, than (b).

3 Background

Scalable sort-last compositing algorithms require that the data be partitioned among the ranks in convex, nonoverlapping pieces, but this requirement is typically not satisfied by unstructured [1, 1, 8, 25, 26] or octree AMR [2, 5] simulations that use nonspatial data distribution methods to accelerate the solver, e.g. ParMETIS for fill-reduction [13], or Morton-order [5]. Such simulations are common in computational fluid dynamics (CFD) [1, 8, 25, 26], medicine [24], and geoscience [2]

We evaluate our approach on two motivating CFD datasets that represent these mesh configurations. The FUN3D [1] Mars Lander uses an unstructured mesh, whereas the Exajet [7] uses a Cartesian AMR mesh. The FUN3D data were generated on Summit at Oak Ridge Leadership Computing Facility (OLCF) using a CFD code developed at the NASA Langley Research Center. The mesh consists of a mix of tetrahedra, pyramid, and prism cells, and were written in a total of 72 subdomains, each storing part of the mesh. In our benchmarks we use the smaller version with 798M total elements that was run at Mach 2.4. Fun3D uses ParMETIS [13] for data decomposition, resulting in mesh cells being distributed among ranks with nonuniform, jagged boundaries that fit together much like puzzle pieces and served as the initial motivation for our work (Figure 2).

MBOIT is a high-accuracy two-pass OIT solution that operates on a moment-based representation of transparency. To avoid collecting transparency by alpha blending along the depth with the Porter and Duff compositing operator, MBOIT looks into approximating the transmittance as a function of depth by two order-independent operations: one to construct the function, the other to recover the transmittance value through the generated function. Defined by $[z, z^2, z^3, z^4]$ for a 4 moments implementation where z represents the depth, the power moments serve as a collection of measures that record transmittance behaviors when traversing through transparent layers. Converted to logarithmic space operations, this representation can be additively constructed for global transmittance information, allowing for accurate, order-independent approximation reconstruction of the transmittance value given any depth. Therefore, the first rendering pass accumulates per-pixel power moments that encode powers of transparency needed to generate the function, and the second rendering pass uses the moments to solve for transmittance value along depth to produce final colors that can also be additively blended. Two data summation phases are required in image space, and thus the scene complexity does not become the bottleneck of the rendering process.

We summarize the algorithm in Algorithm 1 and the two key components in Algorithm 2 and Algorithm 3. For additional details on the MBOIT computation, we refer to the paper by Münstermann et al. [23] and a more detailed description in the preceding paper [29].

Compared to prior OIT methods, MBOIT has been shown to perform well for different signal frequencies, in that it is both truly order-independent throughout the computation, and well approximates reconstruction of the object occlusions in their true depth order. The key steps of moments construction and moments-based transmittance reconstruction, namely the summation loops with *GeneratePowerMoments* and *getTransmittance*, can be executed in an arbitrary order, allowing for an order-independent rendering pipeline. This characteristic sets the foundation of our scalable distributed compositing method.

Algorithm 1 The *RenderMBOIT* algorithm that returns a pixel color rendered with Moment-Based Order-Independent Transparency

```
function RenderMBOIT(volume v, bias β)
    col ← (0,0,0,0), b ← (0,0,...0)
    for each sample s in v do
        GeneratePowerMoments(b, s.depth, s.transmittance)
    for each sample s in v do
        col ← col + getTransmittance(b, s.depth, β) · s.col
    return col
```

Algorithm 2 The *GeneratePowerMoments* function that computes the moments at a given sample point

```
function GeneratePowerMoments(moments b, depth d, transmittance t)
    d ← logDepthWarp(d)                          ▷ rescale logged depth value to [-1, 1]
    absorbance ← −log(transmittance)             ▷ get logged absorbance
    absorbance ← min(absorbance, ABSORBANCE_MAX_VALUE)
    for moment b_i in b do
        b_i ← b_i + pow(s.depth, i) · absorbance ▷ store moments by powers of depth
```

Algorithm 3 The *getTransmittance* function that reconstructs the final transmittance at a given depth by the moments

```
function getTransmittance(moments b, depth d, transmittance t)
    m ← len(b)                                   ▷ get number of moments
    d ← logDepthWarp(d)                          ▷ rescale logged depth value to [-1, 1]
    b_tmp ← (b_1, ...b_m)
    b_tmp ← mix(b_tmp, β.bias_vector)            ▷ bias input data to avoid artifacts
    q ← SolveMat(b_tmp, d)                       ▷ compute Cholesky factorization of the Hankel matrix
    z ← SolvePowerEquation(q, d)                 ▷ get roots of the power equation
    weights_0 ← β.overstimation                  ▷ adjust weight factors by overestimation
    for i < m do
        weights_i ← (z_i < z_0)? 1.0 : 0.0
    p ← SolvePolynomial(weights, z)              ▷ solve for final absorbance vector
    absorbance ← p · vector(1.0, b_tmp)          ▷ compute absorbance value
    return clamp(exp(−b_0 · absorbance))         ▷ return transmittance in the original depth range
```

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2025.3567124

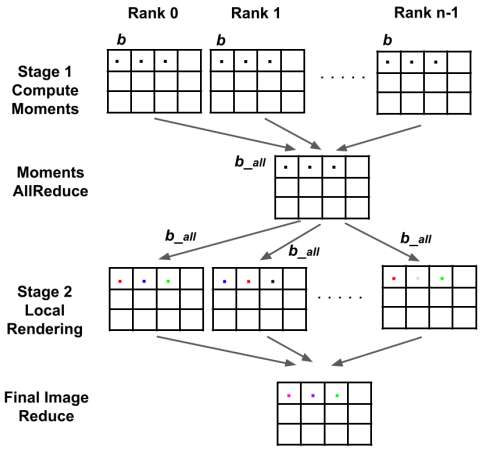


Fig. 5: An illustration of the APC pipeline. Stage one computes a local moments vector per pixel on each rank, adding them up in the moments AllReduce step to form the global moments. The global moments are used in stage two on each rank to approximate transmittance when rendering their local volume to produce final subimages. Finally, all subimages are added using a Reduce onto the display rank.

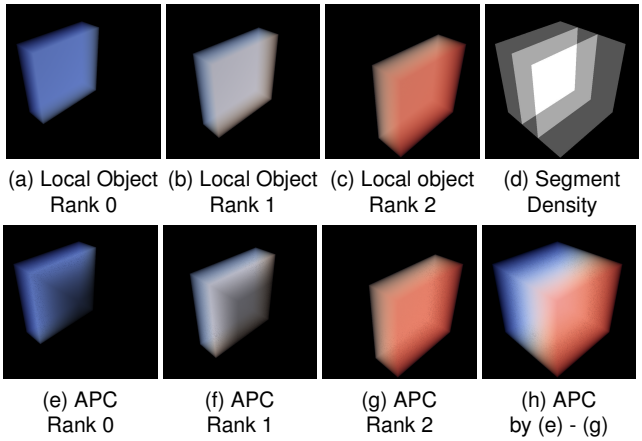


Fig. 6: An example image produced from distributed rendering of a sandwiched volume over three ranks. Each rank has two interleaved slices of a cube. The occluded part has its opacity corrected with MBOIT's approximation, i.e., each rank here has its local objects rendered with the "correct alpha" as it would appear in the final image, which can thus be produced by a simple image space summation.

4 Method

Our rendering pipeline extends the single-node MBOIT pipline to a distributed computing environment to enable highly scalable approximate order-independent compositing. Our Approximate Puzzlepiece Compositing pipeline consists of two main stages: stage one renders the local moments on each rank and computes global moments through an `MPI_Allreduce` add (Section 4.1); stage two uses the global moments to approximate transmittance on each rank to render local partial images (Section 4.1). The partial images are then combined to form the final image through an `MPI_Reduce` add. The advantage of our proposed method for distributed rendering is that each rank can independently produce its part of the moments and final image, with global results produced through single optimized MPI operations (AllReduce and Reduce). An illustration of our pipeline is shown in Figure 5.

4.1 Stage One: Computing Local and Global Moments

The first step is the same as the first stage in Section 3. We must produce a local moments image on each rank that we can combine to produce a global moments image to use for representing absorbance. Each rank traces rays through its local data; however, instead of rendering out final colors as in a typical rendering pass, we compute the moment for the pixel by summing up moments along the ray. These moments are written out to an image to produce a local moments image on each rank. After all ranks have computed their local moments image, we compute the global moments image by performing an `MPI_Allreduce` add on the local moments images. As moments in MBOIT are computed through addition, which is commutative, it makes no difference whether the addition occurs locally on a node or globally over MPI. Thus, the moments computed in our distributed pipeline match those produced by single-node rendering.

4.2 Stage Two: Using Moments to Approximate Transmittance

After the Allreduce add, each rank has the same global moments image, and we can now approximate the global transmittance along each ray as we traverse the volume. In stage 2, each rank produces a local partial image of its data by tracing rays through its local data again, using the global moments image to approximate transmittance. Our rendering algorithm is similar to standard ray-tracing; however, instead of getting the standard RGBA at the voxel position, we drop alpha when blending the sample into the final color and replace the transmittance value with what we get from the MBOIT function. By combining the sampled color and the approximated transparency, we are able to blend the pixel segments in any order.

When all ranks have completed rendering their data using the approximate transmittance data, we have the local images ready with the correct global transmittance. As discussed in Section 3, each local image is completely order-independent because the reconstruction function calculates the transmittance with correct occlusion. Therefore, a `MPI_Reduce` add in image space will produce the result with the correct object ordering.

A three-rank distributed rendering using this pipeline is shown in Figure 6. Figures 6a to 6c are the subvolumes each local renderer owns, rendered in traditional alpha blending. Each rank consists of two interleaved slices that are occluded by either other rank's subvolume or partially by themselves. Figures 6e to 6g show the local rendering of our method after stage 2, where the global transmittance has been approximated to render each rank's local data with the global transmittance. At the final image reduce step in our method, all that needs to be done is to add all the local images together, whereas alpha blending requires sorting for each pixel to determine a correct sequence of operations to produce the final color.

4.3 Implementation

We implement our method within OSPRay [34] by modifying its distributed rendering framework [33] to take advantage of OSPRay's high-fidelity ray-tracing engine for fast volume rendering on modern CPU architectures. To boost performance, we group pixels into tiles for locality, exchange only nonempty tiles, and utilize ISPC [30] to leverage SIMD hardware to accelerate moment computation and transmittance estimation.

MBOIT can also be customized by using different sets of moments, which require slightly different mathematical operations for *GeneratePowerMoments* and *GetTranmittance*, enabling finer adjustments of local rendering computation, memory overheads, and numeric precision. However, the pipeline we describe here remains the same regardless of particular moments of choice. Figure 3 shows a sample sphere volume and comparisons between different configurations. The distributed computation with APC produces a faithful image of the single-node computation quality due to the order-independent nature of MBOIT. Compared to popular alternatives, the MBOIT images provide a more faithful depth perception, presenting a smooth transition for low-frequency volume intervals while preserving high-frequency occlusion details, whereas the method still suffers from inaccurate blending when facing surface-like, extremely thin structures in volume rendering as in Figure 4. A closer look at image-quality evaluation with real-world datasets will be presented in Section 5.2.

As shown in the transmittance curves, the MBOIT method is able to provide an accurate and smooth approximation to the sort-last techniques but tends to overestimate the current transmittance, resulting

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2025.3567124

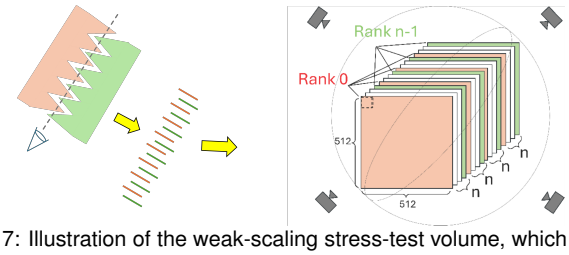


Fig. 7: Illustration of the weak-scaling stress-test volume, which represents a worst-case "gear teeth" boundary. Each rank owns four 512×512 hexahedra layers, interleaved with $n-1$ layers from other ranks in between. Traditional sort-last rendering would require producing and blending individual color fragments for each slice to produce a correct image. Images are captured around a sphere camera orbit in our benchmarks.

Dataset	Num of Subdomains	Num of Cells	Cell Type(s)
Synthetic	64	8388608	hexahedra
FUN3D	72	788841511	tetrahedra, pyramids, prisms
Exajet	128	656444884	hexahedra

Table 1: All unstructured mesh datasets used in the evaluation, with total number of subdomains and total cell numbers and cell types. For the performance experiments, each subdomain is assigned to one MPI rank run by a single compute node.

in slightly brighter pixels in higher-density regions. Different from mainstream compositing techniques, the MBOIT method is not guaranteed to be energy conserving, which is compensated for by additional renormalization operations and bias vectors, as described in section 3 in the original paper [23]. Thus, a constant bias parameter β (as shown in Algorithm 1) is set in the implementation to offset the overestimation, allowing more convincing rendering outcomes. Under this adjusted configuration, the choice of moments functions does not introduce significant image difference as seen in Figure 3c. Therefore, we adopt the most compact representation, i.e., the 4 power moments computation, for maximum performance.

5 Evaluation

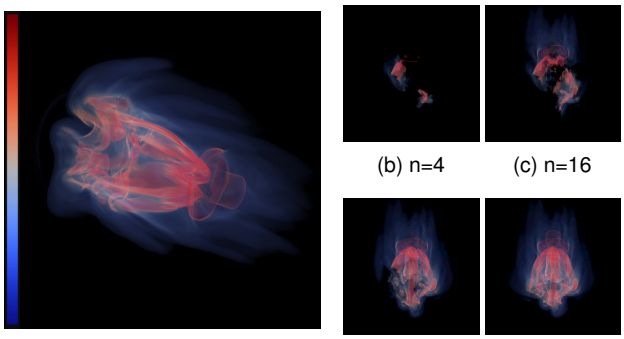
We evaluate our method's image quality and rendering performance through distributed rendering benchmarks performed on the TACC Frontera HPC system, with 56 cores and 192 GB of memory on each Intel Xeon Platinum 8280 compute node. We perform runs on up to 128 ranks with one rank per compute node. For the MBOIT renderer implementation, we use 4 power moments with an overestimation bias $\beta = 0.3$. All images are rendered at a 2560×2560 resolution.

We note that aggregating the entire dataset to a single node during in situ rendering is sometimes not possible due to the memory restriction on the compute nodes, or the heavy data transfer that is order-of-magnitude slower than the rendering. To compare our work to the state-of-the-art sort-last distributed rendering algorithm, we implemented a segment layer-based distributed rendering method, which alpha blends the fragments on every continuous sample interval into a single segment. The local render will thus render into order-based layers, leading to less total data transfer traffic and a reduced sorting workload compared to working with all fragments as in the full individual segment pipeline.

We present all used datasets and the corresponding settings in Section 5.1. The image quality and performance results are shown in Section 5.2 and Section 5.3. In addition, we perform an algorithmic level comparison of communication scaling against standard sort-last compositing methods adapted to support these jigsaw puzzlepiece data boundaries in Sec. 5.4.

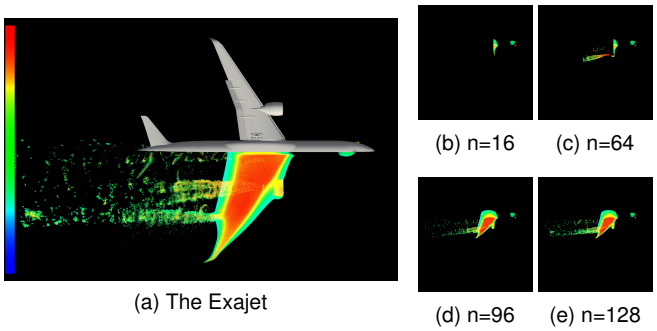
5.1 Evaluation Datasets

The synthetic volume dataset is created based on the worst-case scenario where the camera looks through a zigzagging boundary between



(a) The FUN3D Mars Lander (b) n=4 (c) n=16 (d) n=32 (e) n=72

Fig. 8: The FUN3D from partial domain to full domain. Resulting from a CFD simulation, the subdomains are spatially sparse and consist of nonuniform boundaries, as mentioned before in Figure 2. (b-e) The dataset rendered with 4, 16, 32, and 72 (all) subdomains combined. Even though the number of cells in each subvolume stays roughly the same, the shapes and spatial locations can be unpredictable.



(a) The Exajet (b) n=16 (c) n=64 (d) n=96 (e) n=128

Fig. 9: The Exajet. (a) A bottom-up view of the semi-span Exajet data with the half-plane model on the symmetric side of the axis. (b-e) show Exajet rendered with 16, 64, 96, and 128 (all) subdomains. The data are distributed by splitting the simulation into equal numbers of cells on each rank. The Exajet has a more irregular spatial distribution and complex boundaries than the FUN3D.

different ranks' distributions. This situation produces an extreme case of segment overlap, where each rank generates four local volume slices of $512 \times 512 \times 2$, which are separated by $n-1$ slices from other ranks in between. As seen in Figure 7, the result is a scene where each rank's local data segments overlap all other ranks in-depth, posing a severe challenge to traditional sort-last compositors. Without an order-independent transparency method, a traditional sort-last compositor will need to exchange and sort $4n$ segments (i.e., RGBA color fragments) per pixel to produce a correct image. Note that although the memory requirement can be arbitrarily large for a real-world dataset to store all rendered fragments, we deliberately construct this synthetic where the number of per-pixel fragments grows linearly with the number of nodes for the purpose of scaling pattern comparisons.

We then examine the efficiency of our method on the driving dataset, the FUN3D Mars Lander. This dataset comprises spatially dispersed subvolumes from the simulation, resulting in an uneven distribution across nodes and unpredictable boundaries. Our performance analysis involves loading one subvolume per node up to all 72 subdomains. The performance evaluation is conducted by varying the number of computational ranks. Figure 8 displays the sets of FUN3D subvolumes loaded at different rank counts for the benchmark.

Furthermore, to demonstrate the generality of our method, we include another real-world unstructured CDF dataset, NASA's Exajet, which describes a half-span model of a large civilian transport aircraft consisting of 656 million cell-centered cubic hexahedra. Different from the FUN3D dataset distribution, the Exajet mesh is generated as a single file outputted by the simulation solver. The original Exajet dataset comprises two components: the geometry file detailing the positions of all hexahedra and the value file encoding the velocity magnitude per cell. To create the distributed version of this dataset, we segment

the global files into 128 subfiles, each containing an equal number of elements in the original sequence. As displayed in Figure 9, the resulting boundaries reflect a blend of the native simulation layouts, such as the clustering of turbulence structures from airplane engines, and artificial boundaries introduced by enforcing a consistent element count per rank. This nonuniform distribution intensifies the imbalance in segment overlap across compute nodes.

5.2 Image Quality

To evaluate the rendering quality of APC, we perform an image quality comparison with the single-node MBOIT and single-node sort-last results. As the entire dataset will not fit into a regular Frontera compute node, for the single node rendering each dataset is aggregated to run on one large memory Intel Xeon Platinum 8280M node with 2.1TB of Optane memory. We compare the images with the two real-world datasets in Figure 10.

Due to its more uniform element distribution, the FUN3D dataset is rendered in a smoother and more transparent manner to showcase the internal composition. In contrast, the Exajet has more significant cell size variation, becoming extremely dense near the fuselage and wing, but sparse further from the object. Thus, this dataset is rendered in a more opaque setting for higher visibility of the smaller features. Note that we are able to capture the thin structures close to the landing gear (Figure 10a) and the air turbulence at the bottom of the wings (Figure 10i) with correct object occlusions.

The APC rendered images showcase the capabilities of our method in ensuring smooth volume rendering while capturing finer grain details within AMR meshes, producing high-quality transmittance approximation similar to sort-last's results. We further note that both APC and single-node MBOIT produce identical images for both datasets, demonstrating that APC does not introduce artifacts to the MBOIT computation and that it is able to provide high-fidelity details in various rendering settings.

The inherent image difference between the MBOIT method and sort-last is visually more prominent at higher density, transparent regions like the Exajet wings areas, where a slight overestimation in the transmittance curve accumulates faster by including more samples before termination. As a result, a near-opaque pixel rendered from a tight cluster of transparent elements tends to appear brighter in the final image. Nevertheless, the approximation results still resemble the sort-last images in that the overall color difference is small, and they succeed in preserving important occlusion clues to avoid inaccurate depth perception. Furthermore, since some accuracy loss is expected from using an approximation method, an overall minor overestimation configuration is preferable in practice as an underestimated transmittance curve could lead to structural changes such as surface loss.

5.3 Compositing Performance

As we are primarily interested in supporting in situ rendering of large-scale datasets, we focus our performance evaluation on weak-scaling benchmarks for the synthetic sandwich stress test case (Figures 6 and 7), and the incremental subdomain loading benchmarks for the FUN3D dataset (Figure 8) and the Exajet dataset (Figure 9).

For the synthetic data, we record performance over a four-camera position orbit (illustrated on the synthetic data in Figure 7); on the FUN3D data, we measure performance over a five-camera position orbit performed at three distances from the dataset; and we use a three-camera position orbit on the Exajet dataset. We further note that the performance of traditional sort-last compositors is affected by the camera position as it dictates the projected area of each rank's local data and the sorting. The compositing steps in our pipeline are viewpoint oblivious since only two reductions over the image are needed.

Figure 11 displays overall rendering performance results on the synthetic dataset benchmark on up to 64 ranks. We find that APC closely follows the ideal weak-scaling trend of a flat trend line with both the rendering and the compositing, displaying a clear performance advantage over the sort-last method. When breaking down performance to inspect the stages of our proposed Approximate Puzzlepiece Compositing algorithm (Figure 11), we notice that the data rendering stages are similar for both methods since data-parallel rendering is trivially parallel (Figure 11c), allowing for good local scaling. With the same fixed step size sampling, the slight variation of the performance curves is caused by respective data structure overheads, which are more prominent on the chart in this experiment due to a smaller local rendering load. We also break out compositing alone in Figure 11d, which consists of data communication and final image blending. For APC this part includes the moments transfer and the final image reduction, and for sort-last, it includes segment layers transfer and alpha blend with sorting. APC shows a near-constant scaling and outperforms the sort-last algorithm, which is quickly dominated by communication cost.

To match the in situ analysis process of the distributed dataset derived from the simulation, where each rank handles its own local data, the FUN3D experiments are conducted by loading one subvolume per rank and run up to 72 ranks. The benchmarks on FUN3D are run at three orbit radii, with five camera positions in each orbit. This nonuniform distribution with jagged boundaries results in local ranks' data bounds overlapping, posing a challenge to standard sort-last compositing methods.

Figure 12b shows rendering performance on the five camera positions, and Figure 12a shows performance over the different camera distances. Again, we find that our method does not introduce significant performance overhead at higher core counts overall camera positions and distances. In Figure 13 we break down rendering costs into the local rendering and compositing stages. As in Figure 12, both algorithms' compositing communication costs (Figure 13c) exhibit similar patterns as in the synthetic case, with APC achieving relatively constant cost and outperforming the sort-last method as the full domain is loaded. The crossover of the two performance curves happens at around half of the domain, as seen in Figure 13a. Thus, we find that our sorting and redistribution-free approach is well suited to low overhead rendering of large-scale distributed unstructured meshes. We also note jumps in rendering times as more mesh partitions are added due to the uneven rendering workload each partition incurs.

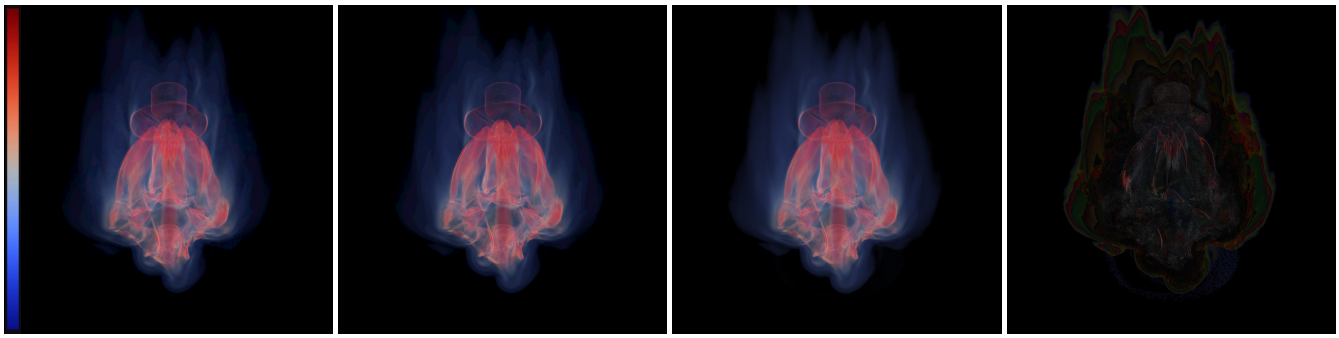
The Exajet experiment is run with three camera positions around an orbit, and the results are shown in Figure 14. Again, for the real-world data benchmark, we vary the number of loaded subdomains with each rank handling its own data. Despite being affected by the more unbalanced rendering loads, the rendering curves for both methods show similar behaviors as in the FUN3D case. As shown in Figure 14's break down, APC suffers from a second rendering stage but has a near-constant communication curve, whereas sort-last's communication costs grow rapidly with the number of ranks. Furthermore, due to the relatively smaller rendering load and more complex boundaries on each rank, communication dominates the overall performance starting from very low core counts.

With three datasets of various data distribution scenarios, we have shown that APC is resistant to communication overheads at high core counts for unstructured meshes with unassuming boundary shapes. In particular, when using 4 power moments, the memory requirement is reduced to 16 bytes per pixel to store transmittance information in the first pass and colors for additively blending to a final image. This is in contrast to sort-last segment compositing, which would require 16 bytes$\times$#segments per-pixel. MBOIT was originally implemented for GPU rendering [23], and thus our method can be easily ported to a GPU use case requiring only 4-8 single-precision values per pixel depending on the moments variant used. By trading off a second local rendering stage, our method provides a more scalable solution to large-scale AMR meshes of unpredictable data distributions.

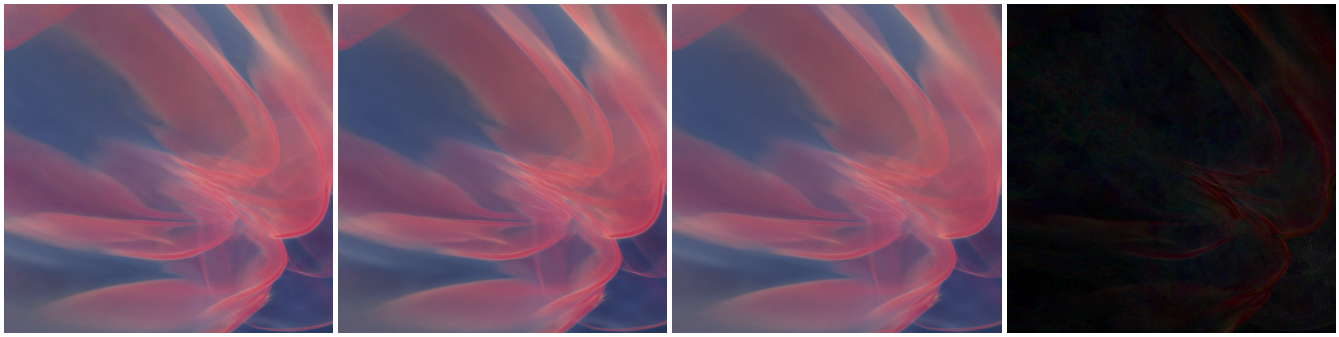
5.4 Algorithmic Analysis

Finally, we perform an algorithmic comparison against standard compositing algorithms to evaluate how traditional sort-last compositing techniques may scale when adapted to support per-segment sorting and compositing. Traditional sort-last methods, e.g., Binary Swap [14], rely on constructing a global sort over the ranks' individual partial images. When ranks' local data overlaps, it is no longer possible to construct this order. To support such data distributions, one could consider extending sort-last compositing to support multiple color segments per

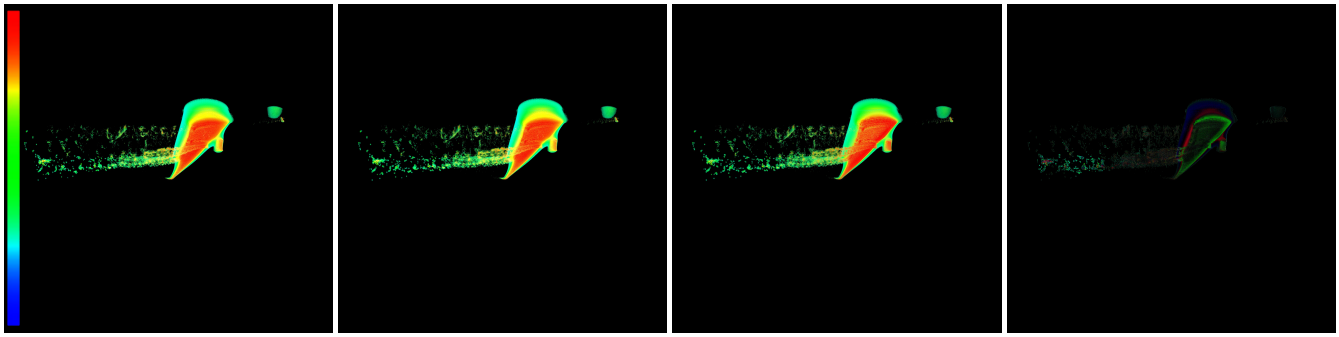
This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2025.3567124



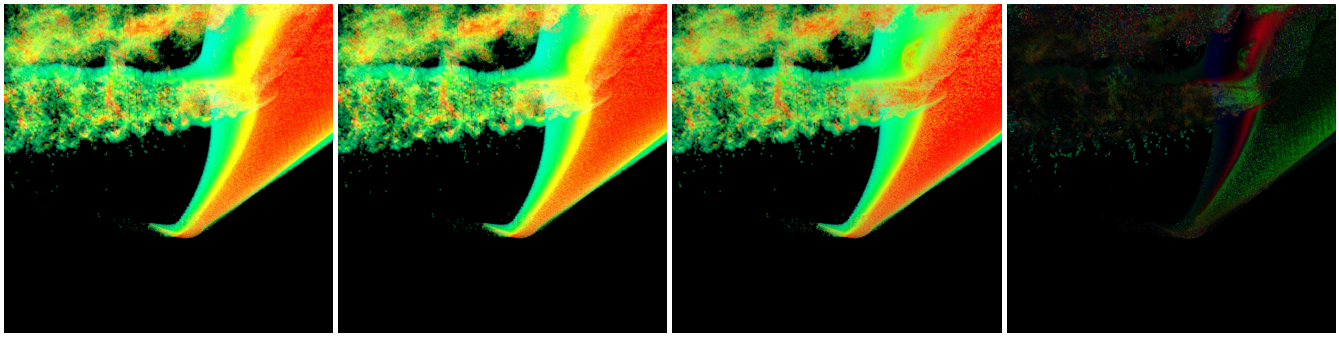
(a) 72 Nodes APC (b) Single-node MBOIT (c) Single-node sort-last (d) Diff (a) and (c), brightness $\times 3$



(e) 72 Nodes APC (f) Single-node MBOIT (g) Single-node sort-last (h) Diff (e) and (g), brightness $\times 3$



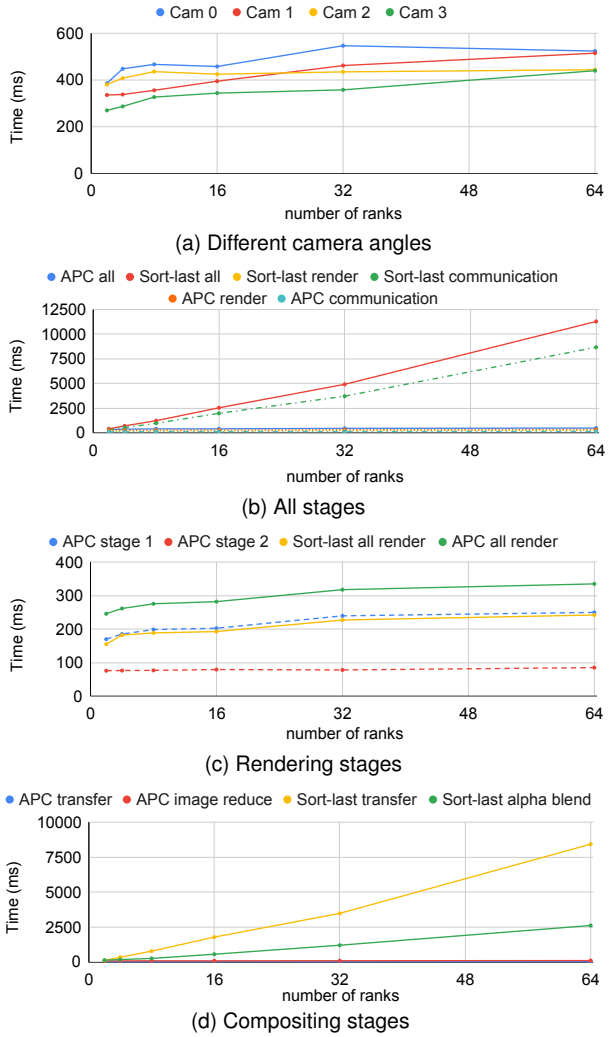
(i) 128 Nodes APC (j) Single-node MBOIT (k) Single-node sort-last (l) Diff (i) and (k)



(m) 128 Nodes APC (n) Single-node MBOIT (o) Single-node sort-last (p) Diff (m) and (o)

Fig. 10: Image comparison with real-world datasets. The first two rows show an overview and a closeup of the FUN3D dataset with distributed APC, single node MBOIT rendering, single node sort-last rendering and the difference image with brightness scaled by 3 for visibility. The following two rows show corresponding images for the Exajet. The FUN3D dataset is rendered with a cold-warm transfer function for its smooth overall structure. The Exajet dataset is rendered with a more opaque rainbow transfer function for a distinct view of the scattered turbulence elements. The respective image similarity measurements for the renderings (a, e, i, m) vs. ground truth images (c, g, k, o) are (a) SSIM=0.88, MSE=19.86, PSNR=35.18 (e) SSIM=0.96, MSE=28.5, PSNR=33.6045 (i) SSIM=0.97, MSE=63.4532, PSNR=30.1402 (m) SSIM=0.82, MSE=315.405, PSNR=23.1761. We can see that by faithfully representing the MBOIT rendering results, APC enables smooth volume rendering while preserving precise high-frequency details, and the rendered images exhibit a close resemblance to sort-last results with the color differences more noticeable at higher-density regions.

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2025.3567124



(a) Different camera angles

(b) All stages

(c) Rendering stages

(d) Compositing stages

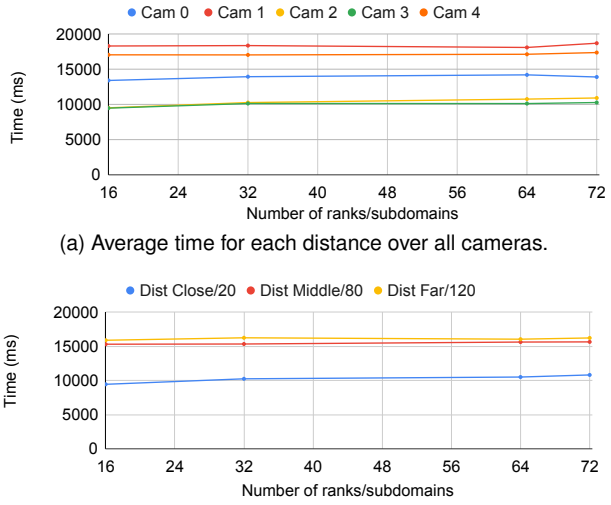
Fig. 11: Synthetic example. (a) APC results remain consistent over different views. (b) The overall APC outperforms and scales better than sort-last. (c) Both the sort-last and APC scale well in local rendering, with APC's second pass doubling the local rendering runtime, as expected. (d) The compositing step (moments transfer and image sum for APC, and segment transfer and alpha-blend for sort-last) is the dominating cost in sort-last, whereas APC's compositing cost remains small and near constant.

rank instead of a single partial image per rank. Each rank would then produce a color segment for each continuous ray-volume interval, and pass this set of segments to the compositing pipeline. Essentially, each rank is treated as multiple virtual ranks per pixel, with one virtual rank per segment.

Given n ranks and m total segments, APC's communication cost scales with $O(n)$; however, Direct send compositing and binary swap would scale with $O(m)$. A traditional sort-last rendering case would have $m = n$, i.e., each rank produces a single segment per pixel for its local brick of data. However, for large-scale unstructured datasets there would be many segments produced on each rank due to the jagged boundaries, and we would expect $m >> n$. Thus, the linear scaling with n of APC would lead to better overall performance in practice.

To evaluate the compositing in a real-world scenario, we compute the number of segments per pixel for the middle distance view of the FUN3D on 64 ranks (Table 2). This configuration has a maximum of 32 segments per pixel. We find that, overall, a large number of pixels produce a single segment for this configuration, resulting in good data transfer costs for direct send and binary swap. APC's communication costs, on the other hand, are fixed, using just an all-reduce and a reduce.

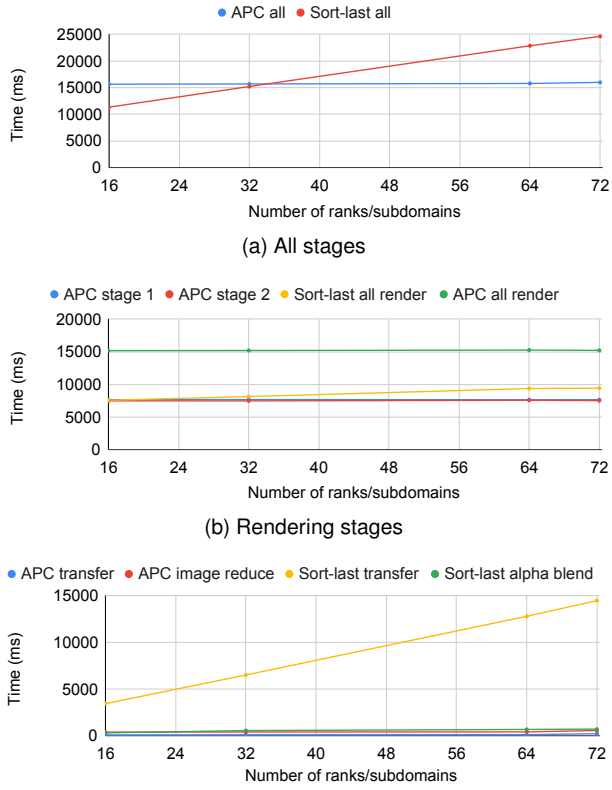
Besides sorting the entire list, direct send does not consider load-balancing in image space whereas our method ensures an even workload



(a) Average time for each distance over all cameras.

(b) Average time for each camera over all distances.

Fig. 12: FUN3D rendering performance on various camera settings. (a) The results from different camera angles. (b) Results from different camera distances. We see that our method performs consistently to different viewing configurations.



(a) All stages

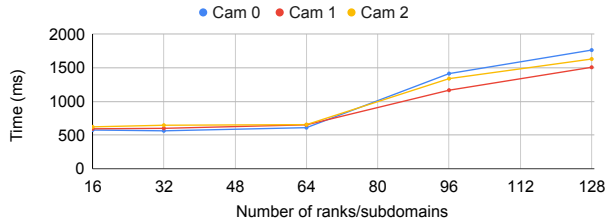
(b) Rendering stages

(c) Compositing stages

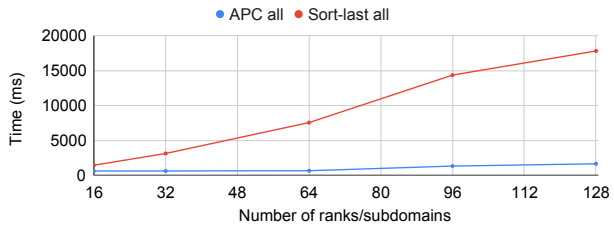
Fig. 13: FUN3D rendering stage performances. Again, for APC both the overall rendering performance and the compositing scale almost constantly. Even though the double rendering pass penalizes overall cost at lower core counts, sort-last becomes disadvantaged with its poor communication scalability. The crossover shown in (a) occurs at about half of the full domain.

for all pixels with a single image-add operation, and the constant-size reduce/all-reduce communication is well optimized by MPI libraries. Binary swap provides better scaling than direct-send but requires a large number of pair-wise image swaps and leads to underutilization at the higher levels of the swap tree. Whereas sort-last methods are unbounded in potential data transfer costs to support such overlapping distributions, our method is bounded by a constant message size, i.e.,

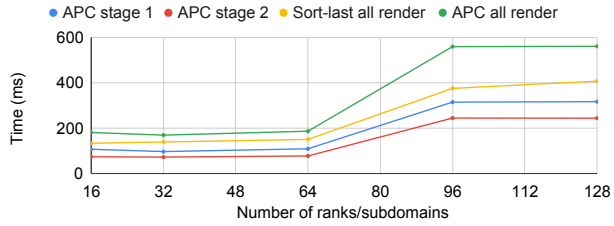
This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2025.3567124



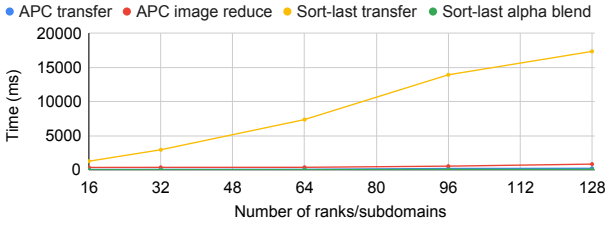
(a) All cameras



(b) All stages



(c) Rendering stages



(d) Compositing stages

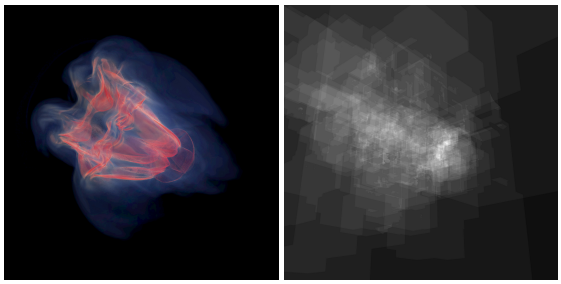
Fig. 14: Exajet rendering stage performances up to 128 core counts. (a) Varying cameras. (b) The overall weak-scaling of APC remains flat compared to sort-last. (c, d) show pipeline timing breakdowns. We see similar patterns in local rendering on both methods with divergence in communication scaling, leading to faster end-to-end performance for APC, especially at high core counts.

the number of ranks, regardless of scene complexity. The main scaling limitation of our approach is the performance of MPI Allreduce and Reduce, which are highly optimized operations in MPI libraries.

6 Conclusion and Limitations

We have presented a technique for compositing large-scale unstructured mesh data for in situ rendering. By adopting an order-independent blending technique, our APC pipeline eliminates the need for sorting or ordering partial images across ranks. Our performance evaluations across synthetic and real-world datasets demonstrate APC's scalability and effectiveness under diverse data boundary conditions.

The compositing performance is discussed in further detail with comparison to the traditional sort-last compositing techniques, and APC shows superior scalability, leading to potentially better overall performance, especially at higher core counts. Even though APC introduced a second local rendering overhead, the message communication steps can both be achieved in small constant sizes through single MPI calls that are optimized by the library. We also examine the output images by comparing to both single-node MBOIT and single-node sort-last rendering, validating that APC delivers high-precision results. Our technique enables efficient parallel distributed visualization with a high-quality



(a) Overview (b) Heatmap

(c) Closeup (d) Heatmap

Fig. 15: Segment heatmap with max number of segments = 28 and 32, grayscale image scaled for visibility. Segment density, within or across ranks, is heavy in certain regions, requiring large amounts of data to be transferred in sort-last compositing.

Method	Average	Upper Bound
Ours	11.7531	128
Direct Send	6.9848	∞
Binary Swap	18.4772	∞

Table 2: Average number of segments transferred per nonempty pixel on FUN3D overview with n=64, Figure 15 (a). The number of ranks (n=64) is chosen as a power of two for fair comparison with binary swap. Our method guarantees an upper bound of twice the number of ranks, which is 128 in this case.

transparency approximation for rendering complex data distributions that are not suited to traditional sort-last compositing techniques.

The main limitations of our proposed method come from using order-independent transparency techniques to eliminate the need for sorting, at the cost of per-sample transmittance accuracy. As MBOIT is an estimation in the end, APC inherently produces images that are slightly different from those of sort-last alpha blending. We note that this would be the case for any order-independent transparency method, as all form an approximation of the transparency term in some form. Eventually, the sort-last technique itself is also one way to approximate real-world light behavior with the advantage of strict fragment ordering. Thus, we believe that an exact color match to the traditional rendering results is not the ultimate goal. The sort-last images are used more as a reference to ground truth in terms of depth perception. The approximated result effectively preserves object ordering, with the cost of being not entirely energy-conserving and thus may need additional bias adjustments. The reconstruction method also requires that there is no volume overlapping, as the opacity is not well defined in this case. Although the requirement that the data be rendered twice for MBOIT incurs an additional cost, we note that this workload is entirely local to each rank and thus achieves good scaling by itself.

Acknowledgments

This work was funded in part by NSF OAC award 2138811, NSF CI CoE Award 2127548, NSF OISE award 2330582, the Advanced Research Projects Agency for Health (ARPA-H) grant no. D24AC00338-00, the Intel oneAPI Centers of Excellence at University of Utah, the NASA AMES cooperative agreements 80NSSC23M0013 and NASA JPL Subcontract No. 1685389. Results presented in this paper were obtained in part using the Chameleon, Cloudlab, CloudBank, Fabric, and

ACCESS testbeds supported by the National Science Foundation. This work was performed in part under the auspices of the DoE by LLNL under contract DE-AC52-07NA27344, (LDRD project SI-20-001).

REFERENCES

[1] W. K. Anderson, R. T. Biedron, J.-R. Carlson, J. M. Derlaga, C. T. Druyor Jr, P. A. Gnoffo, D. P. Hammond, K. E. Jacobson, W. T. Jones, B. Kleb, et al. FUN3D Manual: 14.0. 1. Technical report, NASA, 2023. 1, 2, 3

[2] D. Arndt, W. Bangerth, D. Davydov, T. Heister, L. Heltai, M. Kronbichler, M. Maier, J.-P. Pelteret, B. Turcksin, and D. Wells. The deal.II finite element library: Design, features, and insights. *Computers & Mathematics with Applications*, 2021. 1, 3

[3] L. Bavoil, S. P. Callahan, A. Lefohn, J. L. Comba, and C. T. Silva. Multi-fragment effects on the gpu using the k-buffer. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pp. 97–104, 2007. 2

[4] C. Brownlee, J. Patchett, L.-T. Lo, D. DeMarle, C. Mitchell, J. Ahrens, and C. Hansen. A study of ray tracing large-scale scientific data in parallel visualization applications. In *Proceedings of the Eurographics Workshop on Parallel Graphics and Visualization, EGPGV*, vol. 12, pp. 51–60, 2012. 2

[5] C. Burstedde, L. C. Wilcox, and O. Ghattas. P4est : Scalable Algorithms for Parallel Adaptive Mesh Refinement on Forests of Octrees. *SIAM Journal on Scientific Computing*, 2011. 1, 3

[6] L. Carpenter. The a-buffer, an antialiased hidden surface method. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pp. 103–108, 1984. 2

[7] D. Casalino and A. Hazir. Lattice Boltzmann Based Aeoracoustic Simulation of Turbofan Noise Installation Effects. *23rd International Congress on Sound & Vibration*, 2016. 3

[8] T. D. Economon, F. Palacios, S. R. Copeland, T. W. Lukaczyk, and J. J. Alonso. SU2: An Open-Source Suite for Multiphysics Simulation and Design. *AIAA Journal*, 2016. 1, 3

[9] E. Enderton, E. Sintorn, P. Shirley, and D. Luebke. Stochastic transparency. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, pp. 157–164, 2010. 2

[10] C. Everitt. Interactive order-independent transparency. *White paper, nVIDIA*, 2(6):7, 2001. 2

[11] A. Grosset, A. Knoll, and C. Hansen. Dynamically scheduled region-based image compositing. In *Eurographics Symposium on Parallel Graphics and Visualization*, vol. 2016. Univ. of Utah, Salt Lake City, UT (United States), 2016. 2

[12] W. M. Hsu. Segmented ray casting for data parallel volume rendering. In *Proceedings of the 1993 symposium on Parallel rendering*, pp. 7–14, 1993. 2

[13] G. Karypis, K. Schloegel, and V. Kumar. Parmetis: Parallel graph partitioning and sparse matrix ordering library. Technical report, University of Minnesota, 1997. 1, 3

[14] 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, 1994. 2, 6

[15] M. Maule, J. Comba, R. Torchelsen, and R. Bastos. Hybrid transparency. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pp. 103–118, 2013. 2

[16] M. McGuire and L. Bavoil. Weighted blended order-independent transparency. *Journal of Computer Graphics Techniques (JCGT)*, 2(2):122–141, December 2013. 2

[17] M. McGuire and M. Mara. Phenomenological transparency. *IEEE transactions on visualization and computer graphics*, 23(5):1465–1478, 2017. 2

[18] H. Meshkin. Sort-independent alpha blending. *GDC Talk*, 2(4), 2007. 2

[19] H. Meuer, E. Strohmaier, J. Dongarra, and H. Simon. Top500 supercomputer sites, 2001. 2

[20] S. Molnar, M. Cox, D. Ellsworth, and H. Fuchs. A sorting classification of parallel rendering. *IEEE computer graphics and applications*, 14(4):23–32, 1994. 2

[21] P. Moran. Exajet data portal, 2023. 2

[22] K. D. Moreland. Icet users' guide and reference. Technical report, Sandia National Laboratories (SNL), Albuquerque, NM, and Livermore, CA, 2011. 2

[23] C. Münstermann, S. Krumpen, R. Klein, and C. Peters. Moment-based order-independent transparency. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(1):1–20, 2018. 2, 3, 5, 6

[24] A. Neic, F. O. Campos, A. J. Prassl, S. A. Niederer, M. J. Bishop, E. J. Vigmond, and G. Plank. Efficient computation of electrograms and ECGs in human whole heart simulations using a reaction-eikonal model. *Journal of Computational Physics*, 2017. 3

[25] N. Offermans, A. Peplinski, O. Marin, and P. Schlatter. Adaptive mesh refinement for steady flows in Nek5000. *Computers & Fluids*, 2020. doi: 10.1016/j.compfluid.2019.104352 1, 3

[26] F. Palacios, J. Alonso, K. Duraisamy, M. Colonno, J. Hicken, A. Aranake, A. Campos, S. Copeland, T. Economon, A. Lonkar, T. Lukaczyk, and T. Taylor. Stanford University Unstructured (SU^2): An open-source integrated computational environment for multi-physics simulation and design. In *51st AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*. American Institute of Aeronautics and Astronautics, 2013. 1, 3

[27] 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, 2009. 2

[28] C. Peters and R. Klein. Moment shadow mapping. In *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games*, pp. 7–14, 2015. 2

[29] C. Peters, C. Münstermann, N. Wetzstein, and R. Klein. Improved moment shadow maps for translucent occluders, soft shadows and single scattering. *Journal of Computer Graphics Techniques (JCGT)*, 6(1), 2017. 2, 3

[30] M. Pharr and W. R. Mark. ispc: A spmd compiler for high-performance cpu programming. In *2012 Innovative Parallel Computing (InPar)*, pp. 1–13. IEEE, 2012. 4

[31] T. Porter and T. Duff. Compositing digital images. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pp. 253–259, 1984. 2

[32] M. Salvi, K. Vidimče, A. Lauritzen, and A. Lefohn. Adaptive volumetric shadow maps. In *Computer Graphics Forum*, vol. 29, pp. 1289–1296. Wiley Online Library, 2010. 2

[33] W. Usher, I. Wald, J. Amstutz, J. Günther, C. Brownlee, and V. Pascucci. Scalable ray tracing using the distributed FrameBuffer. *Computer Graphics Forum*, 38(3):455–466, jun 2019. doi: 10.1111/cgf.13702 2, 4

[34] I. Wald, G. P. Johnson, J. Amstutz, C. Brownlee, A. Knoll, J. Jeffers, J. Günther, and P. Navrátil. OSPRay-a cpu ray tracing framework for scientific visualization. *IEEE transactions on visualization and computer graphics*, 23(1):931–940, 2016. 2, 4

[35] Q. Wu, W. Usher, S. Petruzza, S. Kumar, F. Wang, I. Wald, V. Pascucci, and C. D. Hansen. Visit-ospray: Toward an exascale volume visualization system. In *EGPGV@ EuroVis*, pp. 13–23, 2018. 2

[36] D.-L. Yang, J.-C. Yu, and Y.-C. Chung. Efficient compositing methods for the sort-last-sparse parallel volume rendering system on distributed memory multicomputers. *The Journal of Supercomputing*, 18:201–220, 2001. 2

[37] W. Zheng. Research trend of large-scale supercomputers and applications from the top500 and gordon bell prize. *Science China Information Sciences*, 63:1–14, 2020. 2