Eurographics Symposium on Parallel Graphics and Visualization (2019)
H. Childs, S. Frey (Editors)

Scalable Parallel Flow Visualization Using 3D Line Integral Convolution for Large Scale Unstructured Simulation Data

Y. Liao[†][1], H. Matsui[2,3], O. Kreylos[3] and L.H. Kellogg[2,3]

[1]Dept. of Computer Science, University of California, Davis, Davis, California, United States
[2]Computational Infrastructure for Geodynamics, University of California, Davis, Davis, California, United States
[3]Dept. of Earth and Planetary Sciences, University of California, Davis, Davis, California, United States

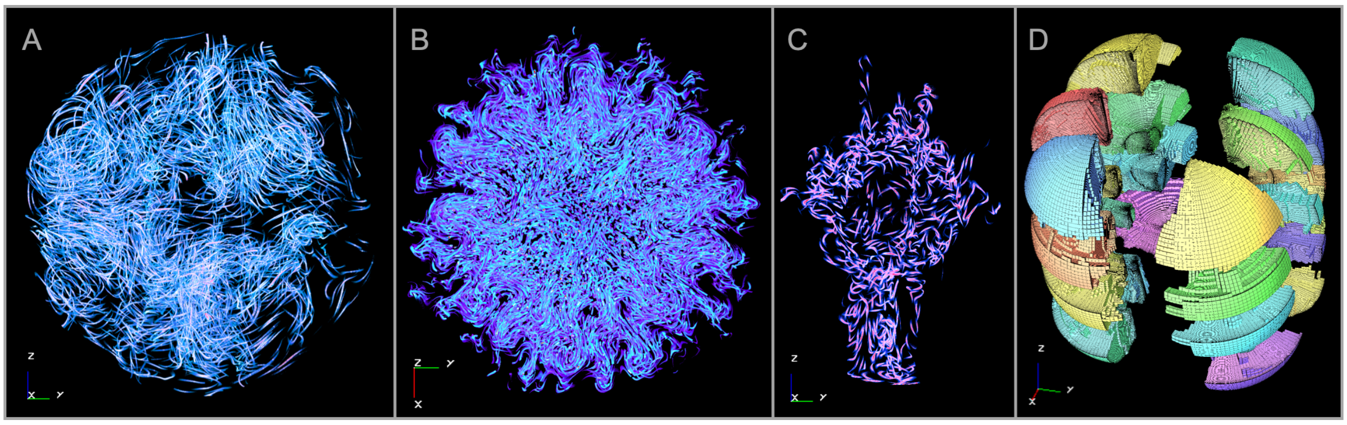

Figure 1: *Visualization of a magnetohydrodynamics (MHD) simulation. A: Visualization of one time step of the geomagnetic field ($\hat{B}$) with a filter ($3.0 < \|\hat{B}\| < 14.0$). B: Geomagnetic field in normalized temperature ($0.2 < T < 1.0$), shown from the North Pole. C: Side view of geomagnetic field with high temperature. D: Load-balancing dataset decomposition. One time step of sample data are 6.7 terabyte.*

Abstract

To address the need of highly efficient and scalable parallel flow visualization methods, we developed a flow visualization system for large unstructured simulation data using parallel 3D line integral convolution (LIC). The main consideration for a parallel LIC implementation is a trade-off between the additional memory cost of replicating cells at sub-domain boundaries, or the communication cost of exchanging those data among computation nodes. To improve scalability, we introduce a load-balancing scheme that partitions datasets based on estimated LIC computation time. We also introduce a data-driven sub-domain extension scheme that determines which external cells at sub-domain boundary need to be added based on current boundary cells, which reduces memory overhead because the same visual quality can be achieved with a significantly smaller number of replicated external cells. We evaluate our visualization method by first comparing its parallel scalability to traditional integral field lines methods. Next, we compare our cost-driven domain decomposition method to existing methods to verify that ours leads to more balanced computation and improved scalability. Finally, we compare our data-driven sub-domain expansion method to traditional layer-based expansion methods in terms of memory overhead and visual quality. We conclude that our parallel 3D LIC method is an efficient and scalable approach to visualization of large and complex 3D vector fields.

CCS Concepts

• ***Human-centered computing*** $\rightarrow$ ***Scientific visualization;*** • ***Software and its engineering*** $\rightarrow$ ***Parallel programming languages;*** • ***Computing methodologies*** $\rightarrow$ *Concurrent algorithms;*

1. Introduction

Flow visualization has been an active field of scientific visualization for decades. Practical methods need to address the large

[†] ygliao@ucdavis.edu

DOI: 10.2312/pgv.20191106

size of flow datasets, and the complex computational processes that generated them. Thus, computation performance is a key factor driving new directions in flow visualization. Many well-established methods are based on the extraction of representative information from large scale data in order to meet the performance and memory requirements of personal computers, such as feature extraction [PVH*03], partition-based clustering [SJWS08], and topology-based approaches [WTS*05, LHZP07]. However, to view flow datasets in their entirety and at full resolution, visualization methods typically need to be executed on massively parallel supercomputers, often on the same computers that generated the datasets. Depending on the architecture of such supercomputers, different approaches can be applied to improve the parallel efficiency of flow visualization in shared memory, distributed memory and hybrid parallel systems [YWM07, PCG*09, PRN*11].

Many flow visualization methods are based on integral curves, i. e., the paths of mass-less particles seeded in a dataset's domain and advected by one of its defined vector fields. Examples of such methods are direct particle tracing [KL96], integral curve-based geometry primitives [BMP*90], and texture-based line integral convolution [CL93]. One fundamental approach to parallel visualization is data parallelism, where each computation node only holds a sub-domain of the full dataset, depending on some decomposition strategy. However, inappropriate decomposition can result in high load imbalance and poor scalability. While several algorithms to minimize load imbalance have been presented, there is no single approach that can handle all the different flow visualization methods. Another fundamental approach is task parallelism. The tasks involved in particle tracing integration are independent, so multiple tasks can be parallelized across the entire domain. Hence, the demands of changing data can result in either a burden of communication or I/O operations. Considering various aspects of parallel system setting, neither parallelization model can be universally optimal for parallel flow visualization. In other words, for most flow visualization parallelization models, load balance and communication are crucial factors for optimizing overall performance.

Line integral convolution (LIC) was originally introduced by Cabral and Leedom [CL93]. It is a texture-based visualization method conveying both local and global vector field features in a dense representation. In particular, LIC is controllable, stable, and generates visualizations over a dataset's entire domain with rich local features. Due to its reliance on densely-packed short integral curves, LIC is computationally expensive, and several parallel methods were proposed to reduce its total computation time for large-scale time-varying 2D data. However, we are not aware of existing parallel visualization methods for 3D vector fields using an extension of LIC to three dimensions (3D LIC). We propose such a method, in order to provide high-performance 3D visualization with full spatial coverage of local flow features for large-scale 3D flow data.

We demonstrate and evaluate our method on a state-of-the-art numerical model of a planetary dynamo [MKB14, MHA*16]. This model performs a magnetohydrodynamics (MHD) [MHLE17] simulation in a rotating spherical shell modeled on the Earth's outer core. The shell (the computational and data domain) extends from the Earth's inner core boundary (ICB: $r_i = 1215\,km$) to the core-mantle boundary (CMB: $r_o = 3485\,km$), in a rotating frame with constant angular velocity. The inner core is solid and considered to be an electrical insulator. The numerical model uses governing equations to investigate how convection of the electrically conductive fluid can continuously regenerate the magnetic field. The dynamo simulation is performed by a pseudo-spectral method and outputs visualization data as a set of multi-dimensional fields defined on a finite element method (FEM) mesh. Although we use a specific MHD simulation as a test case, our proposed 3D LIC method also applies to general flow field data defined on arbitrary domains.

We propose an innovative parallel visualization system using 3D LIC to show flow field features, motivated by the need to visualize vector field data with rich local features and full spatial coverage, which is not addressed by geometry-based flow visualization. Specifically, the main contributions of this paper are: 1. A parallel texture-based 3D LIC visualization method with strong scalability. 2. A new data partition method guided by estimates of total LIC computation time per sub-domain to improve load balance. 3. A data-driven sub-domain expansion algorithm to significantly reduce memory overhead to represent external cells while retaining high visual quality.

We review previous work on LIC and parallel LIC methods, and summarize solutions for parallel efficiency in the next section. Our proposed parallel 3D LIC visualization method is introduced in Section 3. In Sections 4 and 5, we present our workload-based domain decomposition and our data-driven sub-domain expansion, respectively. We analyze our visualization results and evaluate our method's performance, including load imbalance and scalability analysis, in Section 6. We discuss the motivation behind and benefits of using parallel 3D LIC, and how to improve data decomposition for load balancing, in Section 7. Finally, in Section 8, we draw conclusions from the presented analyses and list possible future extensions to our method.

2. Background and related work

A comprehensive survey summarizing significant research on texture-based flow visualization is provided by [LHD*04]. Line integral convolution (LIC) was first introduced by B. Cabral and L. C. Leedom [CL93]. The LIC method was originally introduced for 2D vector fields and used a white-noise texture. Texels are convolved by gathering noise values along the path of field lines by using a filter kernel function to generate a dense visualization of the vector field. Specifically, in a steady vector field, given a field line σ, the pixel located at $x_0 = \sigma(s_0)$ is calculated as [SH95]:

$$I(x_0) = \int_{s_0-L/2}^{s_0+L/2} k(s_0 - s)T(\sigma(s))ds, \tag{1}$$

where I is the intensity of the pixel, T represents the noise texture, which is usually a white noise or random texture, k denotes a kernel function to smoothly taper off contributing field lines, L is the length of the filter kernel and also the length of the field line for convolution, and s stands for the arc length parameterization of the field line curve. Falk and Weiskopf extend this method into 3D domains [FW08], using a volume-rendering process that computes LIC on-the-fly only when needed for a respective sample point. We apply a similar on-the-fly computation idea to a hybrid parallel volume renderer based on MPI and OpenMP [HBC10].

Original LIC proved well-suited for 2D flow visualization, and sparked many advancements and optimizations. For example, fast LIC minimizes redundant computation [SH95]; surface LIC is calculated on a 2D surface embedded in a 3D vector field [MKFI97], and volume LIC [IG97] visualizes 3D vector fields over arbitrary domains. As data scales and simulation scales grow, researchers have focused on parallelizing LIC to support complex and time-varying datasets. Parallel fast LIC [ZSH97] was the first parallel LIC implementation for distributed systems. Later Shen and Kao [SK97] introduced UFLIC to visualize time-varying 2D flow data. A value-scattering approach was developed to increase temporal coherence between animation frames when visualizing time-varying data. To accelerate the computation process, Liu and Moorhead introduced accelerated UFLIC (AUFLIC) to save and reuse results from the value-scattering process along the same pathline [LM05]. Recently, Ding et al. [DLYC15] use an idea similar to UFLIC to implement an accurate parallel unsteady flow LIC for large time-varying flows. Another similar texture-based method, FTLE (the Finite-Time Lyapunov Exponent) is also well parallelized in [NLL*12]. Although there are numerous studies of parallel LIC on both steady and unsteady 2D flows, we are not aware of prior work on parallel 3D LIC on a distributed parallel systems.

Data partitioning, the problem of distributing data logically or physically between processing elements for parallel computation, can be optimized along two dimensions: parallel performance and load balancing. Several intuitive data partitioning methods are geometry-based, such as recursive coordinate subdivision [BB87], recursive inertial subdivision [Sim91], and space-filling curves [PB94]. Partitioning strategies can be carried out either statically, once before the process [PRN*11], or dynamically, repeating at process intervals [PCG*09]. Moloney et al. [MWMS07] apply dynamic load balancing to a sort-first parallel volume method based on predicted load imbalance between computation nodes. Advanced data structures like the kd-tree [ZGH*18] are used to accelerate re-partitioning for load balance. Nouanesengsy et al. [NLS11] propose a workload estimation algorithm to help their partitioning method to achieve better load balancing. Chen et al. [CF08] exploit flow features by applying a dynamic spectral mesh partition to reduce communication and synchronization overhead to improve performance.

Overlapping external cells around sub-domain boundaries, also known as ghost cells, are supported in widely-used parallel visualization platforms like ParaView [AGL*05] and VisIt [CBW*12] to avoid visualization artifacts arising from missing neighborhood information from adjacent sub-domains. External cells are used to generate stitch cells for resolving grid resolution differences in Weber et al. [WCM12]. Isenburg et al. [ILC10] present a scheme for decomposed structured grids that works in a parallel distributed system to stream the needed external cells that are preserved. Patchett et al. [PNP*17] introduce an algorithm to generate external cells in parallel distribution system with no global cell or point IDs. However, as the amount of required external cells depends on visualization algorithms and their parameters, it is desirable to develop methods to minimize the number of external cells, and therefore memory overhead, based on the needs of the algorithm, in this case parallel 3D LIC.

3. Parallel 3D LIC flow visualization

Our parallel 3D LIC visualization method treats every time step of a time-varying dataset as a static flow field. The rendering model is based on a hybrid parallelization of volume rendering on a distributed system, where the ray-casting volume rendering algorithm's scalar field sampling step is replaced with on-the-fly LIC computation. Final image result is generated by using direct send image compositing method [EP07]. As 3D LIC only requires local data to compute each voxel value, it can be parallelized trivially by decomposing a dataset into sub-domains and computing the image contribution of each sub-domain on separate computation nodes. In detail, our algorithm divides per-pixel rays into segments lying inside individual sub-domains and calculates LIC voxel values at evenly-spaced sample position along those ray segments. Ray segments for the same sub-domain are processed in parallel using multiple threads, and samples along each segment are processed serially, using early ray termination and empty-space skipping to avoid evaluating the LIC integral where possible. As usual, LIC values are calculated as the convolution of a noise texture along a field line segment, modulated by a kernel function, and LIC samples along a ray segment are converted into per-segment image contributions via application of a transfer function and back-to-front compositing. After all parallel processes have finished calculating their ray segments' image contributions, those are assembled into the final image in a second parallel compositing step. Our hybrid parallelization approach, employing the MPI framework for parallelization over sub-domains and the OpenMP framework for parallel computation inside each sub-domain, is shown in Figure 2

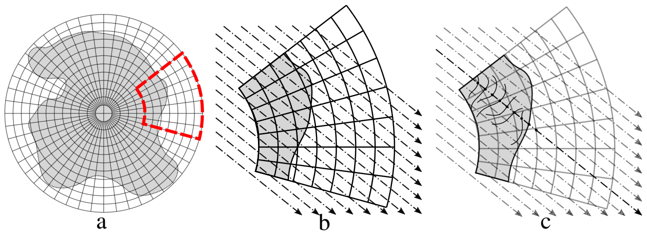

Figure 2: *a) one partition of an entire data domain. b) parallelization over rays to render one sub-domain (marked as red in panel a). Along each ray, like the one shown in panel c, LIC computation is executed serially at each sampling point.*

Our LIC computation is based on a cell-based field line integral method to construct an integral curve segment as a succession of line segments limited to the data set's unstructured grid cells. To compute each such line segment within a hexahedral cell, a local vector field is interpolated from that cell's nodes, and the segment is determined by the entry position and exit position on the surfaces of each cell. We use a second-order iterative estimation method, the midpoint method, to estimate the line segment in one data cell. Higher-order methods can also be used if a more precise result is required. In our case, since the vector field in each cell is linearly interpolated, the second-order method is sufficient. The method is explained in Figure 3, where an unstructured grid cell is represented by square cell in 2D.

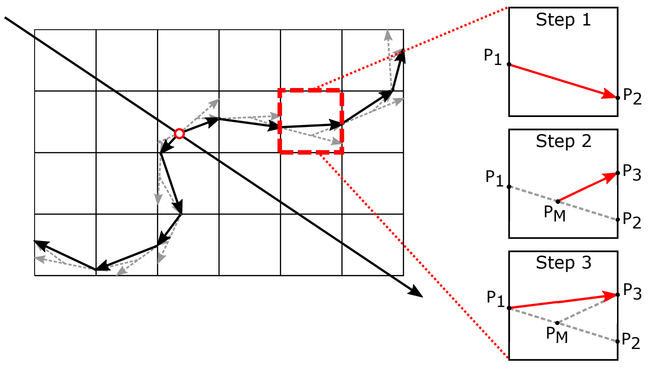

Figure 3: *The midpoint method to calculate line segments of a vector field's integral curves. Right images show the three steps of midpoint method: Step 1 calculates the first ray from the entry point P_1 of the cell by using the interpolated vector at this point. The first ray $\vec{P_1P_2}$ exits current cell at P_2. Step 2 uses the midpoint P_M of $\vec{P_1P_2}$ and its corresponding interpolated vector to calculate the second ray $\vec{P_MP_3}$. This ray exits current cell at P_3. At step 3, the final line segment is decided by the entry point of the cell and the end point of the second ray, which is $\vec{P_1P_3}$.*

The main benefit of parallel 3D LIC over other parallel flow visualization methods such as geometry-based methods is that 3D LIC is more easily optimized for load balancing due to its embarrassingly parallel nature, and the workload of each parallel process being approximately proportional to the total volume of each sub-domain. This means that per-process workloads can be equalized by applying different domain decomposition methods.

4. Adjustment of partitioning for load balancing

One of the most important considerations in parallel computation is keeping workloads balanced between processes, as, typically, total computation time is determined by the longest-running process. There are two basic data partitioning approaches: partition-by-volume and partition-by-nodes (vertices of a dataset's unstructured grid). In 3D LIC's core volume rendering algorithm, we use a fixed step size for sampling steps along each cast ray. The number of samples on a ray is thus proportional to the length of the ray, and the total length of the ray is defined by its intersection with the dataset's domain. Therefore, the total number of samples is proportional to the volume of that domain. In the case of unstructured grids representing FEM mesh datasets, as used in the present study, volumes of individual cells vary across the domain. Due to varying cell volumes and sub-domain shapes, partition-by-node methods cannot determine the total number of sampling steps just based on a sub-domain's number of nodes. Instead, since the number of samples is proportional to a sub-domain's volume, we use a partition-by-volume method to decompose the dataset. We first compute the volume of each cell in the unstructured grid, and subsequently are able to partition the dataset by sub-domain volume. The effects of different domain decomposition schemes on load balance are shown in Figure 5.

As can be seen in Figure 5, the total workload of 3D LIC is not fully balanced even when the number of samples computed in each sub-domain is balanced. This is due to the volume rendering algorithm's sampling step being replaced by on-the-fly LIC computation. Unlike the original sampling step's constant cost, LIC computation requires tracing and convolving along a fixed-length field line segment to avoid artifacts, as pointed out in [CL93, FW08]. Since we trace integral curves by assembling line segments limited to unstructured grid cells, the number of LIC tracing steps per sample depends on the size of the grid cells in a sample's neighborhood. Consequently, if cell sizes vary greatly across the unstructured grid, an equal-volume partitioning scheme still leads to an unbalanced computation process.

To solve this problem, we propose a method to estimate the total required computation time for 3D LIC in each sub-domain. First, we create a set of equal-volume sub-domains by grouping nodes from the unstructured grid. Before reconstructing the grid's connectivity and assembling corresponding field data, we seed a set of particles to simulate the line integration process. By seeding a sufficient number of particles, we approximate the average calculation time of one single LIC computation in each sub-domain, and estimate the total computation time for each sub-domain as the product of that time and the sub-domain's volume. Afterwards, we re-partition the entire dataset so that every sub-domain's volume is inversely proportional to its estimated per-sample LIC computation time. While this adjusted decomposition scheme is still only an estimate of total computation time and can therefore not guarantee perfectly balanced workloads, our experiments indicate that the adjusted method is a significant improvement over the original equal-volume scheme in the context of 3D LIC visualization.

5. Vector-driven external cell expansion

In the traditional layer-based external cells expansion method, external cells are generated from boundary cells by their adjacency relation and expanded by layers: Layer 0 is initialized to the sub-domain's boundary cells, and layer $n+1$ is created by collecting all neighbors of cells in layer n that are not yet part of any layer. As the external cell layers expand, some collected cells may not be used in the LIC computation. In the field line integration of LIC, we compute one line segment in each cell from face to face by using a second order iterative method. Most field lines entering a cell will have similar paths and are very likely to exit through the same face and enter the same next cell. Even in a two-way iteration in both the forward and backward direction, one step of the line integration will enter at most two of neighboring cells.

In our vector-driven external cell expansion method, we selectively expand external cells that have a high probability of being used in line integration to reduce memory cost. Initializing external cell layer 1 as above, we then, for all cells in layer n, cast a ray from the center position of each cell with a vector that is interpolated from that cell's nodes. The ray will hit two faces, in forward and backward direction, respectively, or more faces if the ray hits a cell edge. We choose the adjacent cells sharing those faces as candidates for layer $n+1$. This operation provides an estimation of which cells the next line integration segment will pass through. After casting rays for all external cells of layer n, we will collect all marked candidate cells as external cell layer $n+1$. This vector-driven external cell expansion is a subset of traditional layer-based expansion and therefore not uniform. The difference between our vector-driven expansion method and the traditional layer-based expansion method is shown in Figure 4.

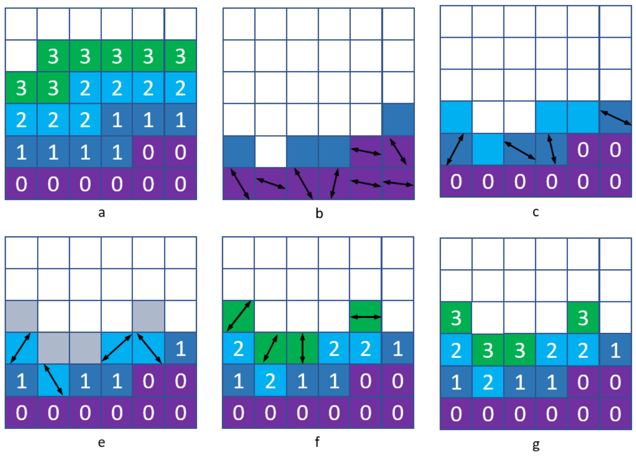

Figure 4: *A 2D abstract diagram illustrating the difference between vector-driven external expansion method and traditional layer-based expansion method. Each square cell represents an unstructured grid cell in our case. (a) is an example of 3 layers expansion by using the traditional layer-based method and pictures from (b) to (g) show the procedure of our method to expand the external cells by the vector features of each cell. The 0 cells are internal cells on the boundary, and the numbers 1 to 3 indicate external cell layers expanded by order.*

6. Results

We developed our parallel 3D LIC visualization system to visualize data from a magnetohydrodynamics (MHD) simulation of the Earth's outer core [MHLE17]. This visualization system is built on a hybrid parallel foundation, using the Message Passing Interface (MPI) for distributed memory parallelism and OpenMP for multi-threading parallelism in each computation node. The test dataset used in this analysis has 31,236,480 ($N_r \times N_\theta \times N_\varphi = 255 \times 384 \times 319$) nodes, where N_r, N_θ, N_φ are the radial, elevation, and azimuthal directions, respectively. Due to the dataset's high number of variables, each timestep weighs in at about 36 GB. We set the pixel count of each output image to be 1600×1200. We ran our experiments on the Intel Xeon Platinum 8160 ("Skylake") skx nodes in Texas Advanced Computing Center's (TACC) Stampede2 cluster. Our experiments used up to 768 skx nodes with 6144 cores total.

The visualization system comprises two separate pieces of software: a data partitioning pre-processor for static data partitioning, and the main visualization program for parallel 3D LIC visualization. The data partitioner is in charge of generating mesh data, expanding external cells and constructing field data. The outputs of the data partitioner are used as the inputs of visualizer to generate the final result. The final visualization is returned to the user as a bitmap or PNG format file.

6.1. Load balancing

To show that our LIC-adapted volume-based partitioning method achieves a more balanced workload, we compare it to two other domain decomposition approaches: equal-nodes and equal-volume, on the same test data. The dataset is decomposed into 24 sub-domains and the result is shown in Figure 5. As the final image can only be assembled once all parallel tasks have completed, the slowest task will determine overall calculation time. As seen in the bottom right of Figure 5, our method is more balanced based on comparing the maximum and minimum computation time among all tasks. The small standard deviation of the elapsed time for our method shows an even distribution of work among tasks and better parallel performance. However, as mentioned before, our method is not optimal in terms of load balance because there are several estimation errors that misguide the data partitioning scheme: first, workload estimation is based on an initial equal-volume partition; second, we divide sub-domains according to an estimate of cells' volumes instead of their actual volumes; third, the workload estimation result has unavoidable errors caused by the sample particle seeding strategy.

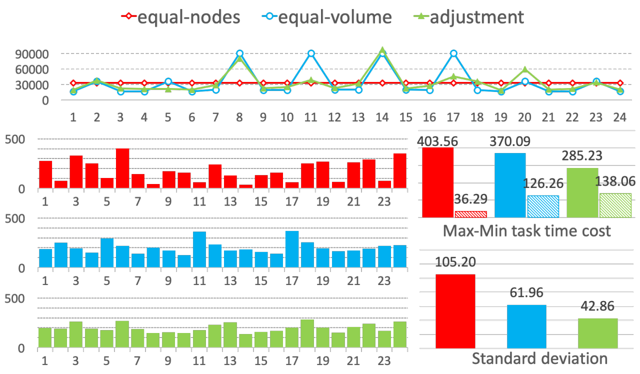

Figure 5: *A set of graphs for load balancing and data partitioning. The equal-nodes method is colored red; the equal-volume method is colored blue; our LIC-adapted volume-based method is colored green. The top graph is the number of nodes in each sub-domain as selected by the three methods. The bottom left histograms are the performance of each task by using the three partition methods. On the right, the top shows the maximum and minimum task time of each method and the bottom shows the standard deviation of task time of each method.*

6.2. Scalability analysis

To evaluate our visualization system's capability of handling growing problem scales, we conducted a scalability analysis on both the MPI parallelization and the OpenMP Multi-threading parallelization. In the following two sub-sections, we will discuss how the solution time varies when the MPI process number is increased with a fixed computation resources on each task, and how the solution time varies when the number of OpenMP threads is increased with a fixed computation resources for entire tasks. This analysis can predict the applicability of our parallel 3D LIC method for larger-scale problems.

6.2.1. Scalability of MPI parallelization

The two major factors that could affect the parallel scalability are load imbalance and communication among processing elements. The 3D LIC rendering in each sub-domain is independent from that in other sub-domains. Communication is only required during final image compositing where the system needs to calculate final pixel values by combining all ray segments computed by all tasks. Under a reasonable dataset partition, we can expect our parallel 3D LIC to scale well.

In this scalability analysis of MPI, we compare our parallel 3D LIC with a parallel streamline visualization (Streamline) by applying both methods on the same geodynamo MHD simulation data. The geodynamo simulation is designed to scale up on massively parallel computers, to enable the domain scientists to sufficiently resolve features in the flow [MHA*16]. Consequently, a visualization method with good scalability is necessary to keep up with the growth of the simulation scale. There are difficulties to fairly comparing the efficiency of parallel 3D LIC and parallel streamline without a consistent computation scale. Considering the nature of their representation with different parameters, the visualization quality is also not the objective of this comparison. However, we can compare these two methods' parallel scalability to show their potential to handle a growing data scale. Due to its lack of required communication between computation nodes, parallel 3D LIC is expected to scale better than parallel Streamlines.

We conducted an experiment by using an increasing number of processor cores on an increasing number of sub-domains to visualize one test dataset. To eliminate the influence from multi-threading, we always keep the same number (8 in our case) of OpenMP threads for each MPI process. The speed-up factors of parallel 3D LIC and parallel Streamline are shown in Figure 6. The scalability of 3D LIC is close to the ideal across all tested configurations, while the scalability of Streamline is weaker (approximately $\frac{1}{16}$ times the ideal scaling). Parallel 3D LIC mainly comprises a ray casting sub-process and an image composition sub-process. As we expected, ray casting achieves good scaling while image composition fares worse due to the need for ray segment data exchange among processing elements. We noticed a sudden improvement in speed-up factor around 96 MPI processes. We believe this is caused by the total per-process memory footprint dropping below L3 cache size as the dataset is decomposed into smaller sub-domains. On the whole, we can conclude that the scaling efficiency of parallel 3D LIC is close to optimal and that 3D LIC scales much better than Streamline visualization.

6.2.2. Scalability of OpenMP Multi-threading

Our hybrid parallel model for 3D LIC applies not only MPI-based parallelization on multiple data sub-domains, but also OpenMP-based multi-threading parallelization inside each sub-domain. Multi-threading can improve the performance and concurrency of each MPI process. We expect that increasing the number

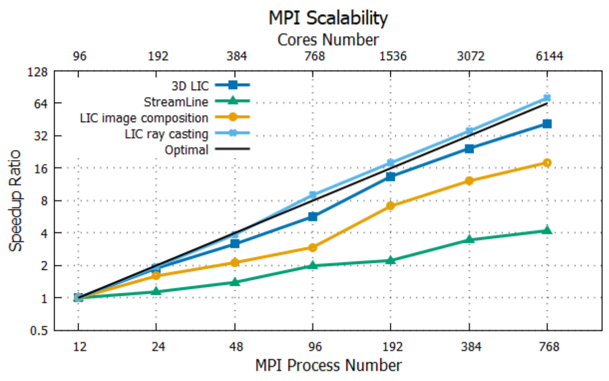

Figure 6: *Performance speed-up plot for different numbers of MPI processes. Each process element is assigned a fixed number of 8 cores for multi-threading. The bottom horizontal axis is the increasing number of MPI processes and the top horizontal axis is the corresponding increasing number of cores.*

of OpenMP threads while keeping the number of MPI processes fixed will also achieve good scaling as the total number of processor cores is increasing with the number of threads used.

In our experiment, the dataset was divided into 192 sub-domains. We measured the computation time of the entire solution while increasing the number of threads in each MPI process from 1 to 24 (each chip socket has 24 cores); consequently, between 192 and 4608 cores were used in this experiment. The speed-up factor of computation time as a function of number of threads is shown in Figure 7. As seen in Figure 7, 3D LIC scales well with respect to OpenMP parallelization. As in our analysis of MPI scalability, the ray casting sub-process scales better than the image composition sub-process because it does not require inter-process communication. But, as the number of sub-domains increases, image composition is taking up a larger portion of total computation time because OpenMP can not parallelize data communication. It will become a bottleneck for multi-threading performance.

6.3. Visualization with external cells

Although our parallel 3D LIC visualization method requires fewer exterior cells than other integration-based flow visualization methods, the number of external cell layers around each sub-domain still affects image quality at sub-domain boundaries. Consequently, the memory cost to represent external cells can still be substantial if a large number of external cell layers are used. Considering the low utilization of external cells, it is necessary to apply an intelligent expansion method that can selectively expand external cells to minimize that memory cost.

Our external cell expansion method will not include external cells that have lower likelihood of being used in the LIC computation. Ideally, the visualization result will be of similar quality with our method while it is using many fewer external cells than the traditional way. As it is hard to compare visualization quality precisely by looking at the entire visualization, we highlight subtle changes in the rendered images near the sub-domain boundary in

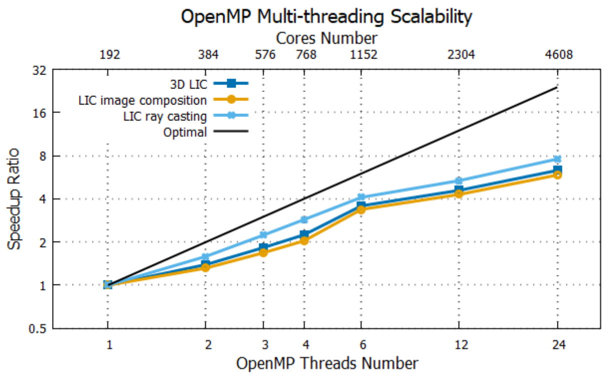

Figure 7: *Speed-up plots for OpenMP multi-threading parallelization on TACC Stampede2. The data set is divided into 192 subdomains. As the number of assigned computation nodes increases, the OpenMP multi-threading scale for each task is increased which is shown in the bottom horizontal axis.*

Figure 8. The figure shows the magnetic field at normalized temperature range $0.3 < T < 1.0$ to display how the magnetic field is twisted by the upwelling flow from the inner core boundary. As shown in the top table of Figure 8, our vector-driven method requires much fewer external cells than the traditional layer-based method. As seen in the result images in the bottom row, the horizontal discontinuity is fading away as the number of external cell layers increases. Our method's significant reduction in memory cost creates a trade-off: comparing a vector-driven expansion to 9 layers and a layer-based expansion to 5 layers shows similar visual quality, while the memory cost of the vector-driven expansion is significantly lower. Overall, our method needs to expand more levels than the layer-based method, but still requires much fewer cells.

7. Discussion

7.1. The benefit of using 3D LIC on simulation flow data visualization

A main benefit of LIC-based visuaization methods is the freedom to trade off feature density and visual complexity. The noise function that is one of the inputs to LIC (the visualized vector field being the other) will not directly affect the shape of vector features, but the appearance of the final image. For example, LIC can be used to visualize data at different levels of detail by changing the frequency distribution of the noise function, as shown in Figure 9. In addition, by applying different kernel functions for the LIC computation, LIC can show additional properties of the visualized vector field. For example, by using an asymmetric kernel, the final LIC image can indicate the direction of the vector field.

Because the distribution of global vector features is difficult to predict in geometry-based flow visualization, it is difficult to control the visualization to cover all features within a specific area. On the other hand, as a texture-based method focusing on local features, 3D LIC is able to provide results that fully cover the specified region of interest. For example, we show how magnetic field changes in areas with different ranges of temperature in Figure 10.

For unsteady vector fields, animation is an useful method to show dynamic features. LIC can generate more natural-looking animations for unsteady or time-varying datasets. Due to its texture-based nature, our method can limit the animation to any selected area all times, while geometry-based methods can potentially change their objects' geometries as the vector field is varying. The local LIC volume streaks have simple and small structures whose changes are easier to observe and understand. An example of a 3D LIC animation for a time-varying dataset is provided in the supplemental video.

7.2. Optimization of data partitioning

The output of MHD simulation consists of mesh data and field data. In the present case, both components are not only used for visualization but also to partition the dataset domain, and, after repartitioning the mesh data, field data needs to be re-associated with the new partitions. Currently, we use a serial process to partition mesh data when there is a need for re-partitioning, and a parallel process to re-associate field data with the decomposed mesh. As a result, mesh data partitioning is currently a bottleneck for time-varying data which may require multiple re-partitioning operations as visualization progresses through time. We plan to parallelize the re-partitioning process to support dynamic data partitioning and larger mesh data sizes in the future.

In our workload estimation algorithm, we seed particles to simulate the field line integration during LIC computation. The accuracy of the estimation increases with the number of particle samples. However, as the number of seed samples increases, so does the cost for the estimation process. The trade-off between estimation accuracy and estimation cost requires additional consideration. In addition, we currently apply an evenly distributed seeding strategy to sample particles. Other advanced seeding strategies could potentially improve the accuracy of the result.

8. Conclusions and Future Work

We presented a novel parallel visualization system using 3D line integral convolution to visualize flow data. We demonstrated our parallel 3D LIC method's scalability both in terms of number of computation nodes as well as number of threads and cores per computation node. Our system used a novel LIC-adjusted volume-based partitioning method to achieve more balanced parallel workloads and better parallel efficiency and scalability. In addition, we presented an optimized vector-driven external cell expansion method specifically designed for 3D LIC, which significantly reduces the amount of redundant external cells generated by traditional layer-based expansion methods.

Future work includes to enable our system to handle sequence of unsteady 3D flow data. an in-situ visualization with existing simulation process is one solution for improving the performance, for example the MHD geodynamo simulation software. Since the simulation system is now well-parallelized, we anticipate an improvement if the parallelization of the simulation and visualization can be

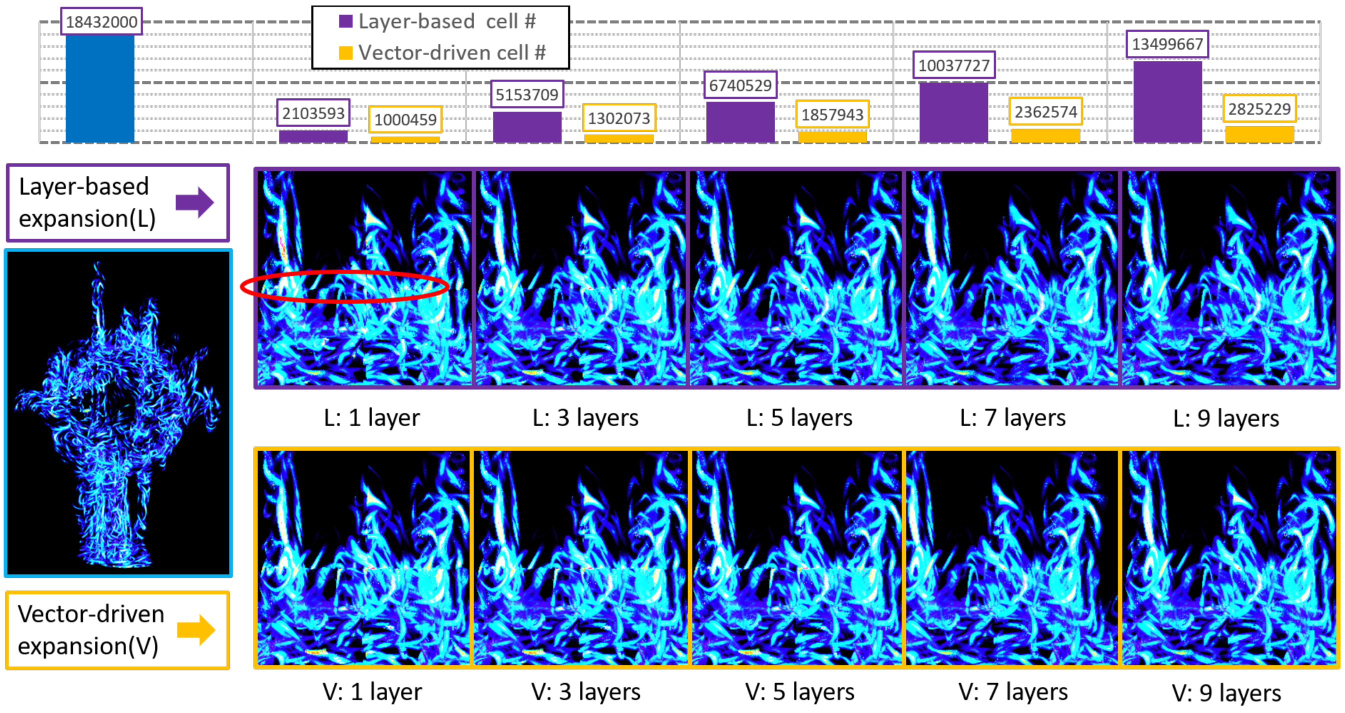

Figure 8: *Comparison of visualization quality for the vector-driven and the layer-based external cell expansion method. The top graph shows the number of external cells generated by different methods using different numbers of layers. The blue bar is the total number of cells in the dataset. The purple bar represents the number of external cells generated by the layer-based method, and the orange bar represents the number of external cells generated by our vector-driven method. The images in the bottom row are a close-ups showing the same area covering the boundary between two sub-domains, from images generated with different numbers of external cell layers. The left blue frame image is the entire visualization result. Images with purple frame are layer-based results in different layers. Images with orange frame are vector-driven results. There is a horizontal discontinuity in the middle of vertical direction.*

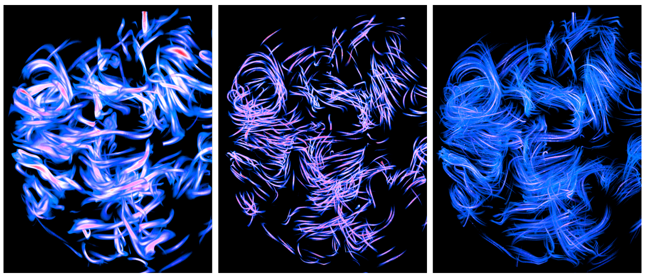

Figure 9: *An example of using different frequency of noise texture. From left to right, pictures are magnetic field data whose amplitude is between 5.0 and 14.0 by using from low to high frequency noise textures.*

combined without an intermediate I/O operation. We also expect a parallel dynamic partition algorithm can be applied to improve the parallelism efficiency. In addition to the geodynamo simulation, the parallel 3D LIC method can be generally applied to other flow field data.

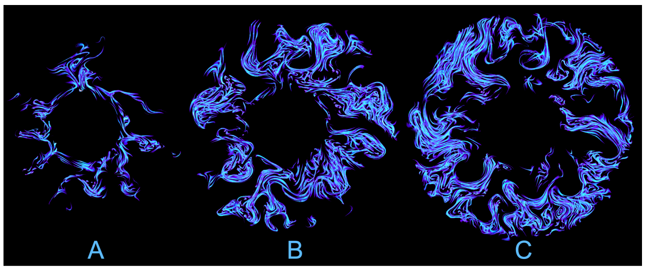

Figure 10: *An example of showing magnetic field in different temperature range from the north pole. Vertical range is $-0.195r_o < z < 0.0$. From left to right, pictures are within normalized temperature range A:$0.2 < T < 1.0$, B:$0.08 < T < 0.2$, C:$0.0 < T < 0.08$.*

References

[AGL*05] AHRENS J., GEVECI B., LAW C., HANSEN C., JOHNSON C.: 36-paraview: An end-user tool for large-data visualization. *The visualization handbook 717* (2005). 3

[BB87] BERGER M. J., BOKHARI S. H.: A partitioning strategy for nonuniform problems on multiprocessors. *IEEE Transactions on Computers*, 5 (1987), 570–580. 3

[BMP*90] BANCROFT G. V., MERRITT F. J., PLESSEL T. C., KELAITA P. G., MCCABE R. K., GLOBUS A.: Fast: a multi-processed environment for visualization of computational fluid dynamics. In *Visualization, 1990. Visualization'90., Proceedings of the First IEEE Conference on* (1990), IEEE, pp. 14–27. 2

[CBW*12] CHILDS H., BRUGGER E., WHITLOCK B., MEREDITH J., AHERN S., PUGMIRE D., BIAGAS K., MILLER M., WEBER G. H., KRISHNAN H., ET AL.: *VisIt: An end-user tool for visualizing and analyzing very large data*. Tech. rep., Ernest Orlando Lawrence Berkeley National Laboratory, Berkeley, CA (US), 2012. 3

[CF08] CHEN L., FUJISHIRO I.: Optimizing parallel performance of streamline visualization for large distributed flow datasets. In *2008 IEEE Pacific Visualization Symposium* (2008), IEEE, pp. 87–94. 3

[CL93] CABRAL B., LEEDOM L. C.: Imaging vector fields using line integral convolution. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (1993), ACM, pp. 263–270. 2, 4

[DLYC15] DING Z., LIU Z., YU Y., CHEN W.: Parallel unsteady flow line integral convolution for high-performance dense visualization. In *Visualization Symposium (PacificVis), 2015 IEEE Pacific* (2015), IEEE, pp. 25–30. 3

[EP07] EILEMANN S., PAJAROLA R.: Direct send compositing for parallel sort-last rendering. In *Proceedings of the 7th Eurographics conference on Parallel Graphics and Visualization* (2007), Eurographics Association, pp. 29–36. 3

[FW08] FALK M., WEISKOPF D.: Output-sensitive 3d line integral convolution. *IEEE Transactions on visualization and computer graphics 14*, 4 (2008), 820–834. 2, 4

[HBC10] HOWISON M., BETHEL E., CHILDS H.: *MPI-hybrid parallelism for volume rendering on large, multi-core systems*. Tech. rep., Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States), 2010. 3

[IG97] INTERRANTE V., GROSCH C.: Strategies for effectively visualizing 3d flow with volume lic. In *Proceedings of the 8th Conference on Visualization'97* (1997), IEEE Computer Society Press, pp. 421–ff. 3

[ILC10] ISENBURG M., LINDSTROM P., CHILDS H.: Parallel and streaming generation of ghost data for structured grids. *IEEE Computer Graphics and Applications 30*, 3 (2010), 32–44. 3

[KL96] KENWRIGHT D. N., LANE D. A.: Interactive time-dependent particle tracing using tetrahedral decomposition. *IEEE Transactions on Visualization and Computer Graphics 2*, 2 (1996), 120–129. 2

[LHD*04] LARAMEE R. S., HAUSER H., DOLEISCH H., VROLIJK B., POST F. H., WEISKOPF D.: The state of the art in flow visualization: Dense and texture-based techniques. In *Computer Graphics Forum* (2004), vol. 23, Wiley Online Library, pp. 203–221. 2

[LHZP07] LARAMEE R. S., HAUSER H., ZHAO L., POST F. H.: Topology-based flow visualization, the state of the art. In *Topology-based methods in visualization*. Springer, 2007, pp. 1–19. 2

[LM05] LIU Z., MOORHEAD R. J.: Accelerated unsteady flow line integral convolution. *IEEE Transactions on Visualization and Computer Graphics 11*, 2 (2005), 113–125. 3

[MHA*16] MATSUI H., HEIEN E., AUBERT J., AURNOU J. M., AVERY M., BROWN B., BUFFETT B. A., BUSSE F., CHRISTENSEN U. R., DAVIES C. J., ET AL.: Performance benchmarks for a next generation numerical dynamo model. *Geochemistry, Geophysics, Geosystems 17*, 5 (2016), 1586–1607. 2, 6

[MHLE17] MATSUI H., HEIEN E., LOKAVARAPU H., ESSER T.: Calypso v1.2.0 [software]. doi:10.5281/zenodo.890016, 2017. 2, 5

[MKB14] MATSUI H., KING E., BUFFETT B.: Multiscale convection in a geodynamo simulation with uniform heat flux along the outer boundary. *Geochemistry, Geophysics, Geosystems 15*, 8 (2014), 3212–3225. 2

[MKFI97] MAO X., KIKUKAWA M., FUJITA N., IMAMIYA A.: Line integral convolution for 3d surfaces. In *Visualization in Scientific ComputingâĂŹ97*. Springer, 1997, pp. 57–69. 3

[MWMS07] MOLONEY B., WEISKOPF D., MÖLLER T., STRENGERT M.: Scalable sort-first parallel direct volume rendering with dynamic load balancing. 3

[NLL*12] NOUANESENGSY B., LEE T.-Y., LU K., SHEN H.-W., PETERKA T.: Parallel particle advection and ftle computation for time-varying flow fields. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis* (2012), IEEE Computer Society Press, p. 61. 3

[NLS11] NOUANESENGSY B., LEE T.-Y., SHEN H.-W.: Load-balanced parallel streamline generation on large scale vector fields. *IEEE Transactions on Visualization & Computer Graphics*, 12 (2011), 1785–1794. 3

[PB94] PILKINGTON J. R., BADEN S. B.: Partitioning with spacefilling curves. 3

[PCG*09] PUGMIRE D., CHILDS H., GARTH C., AHERN S., WEBER G. H.: Scalable computation of streamlines on very large datasets. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis* (2009), ACM, p. 16. 2, 3

[PNP*17] PATCHETT J. M., NOUANESENGESY B., POUDEROUX J., AHRENS J., HAGEN H.: Parallel multi-layer ghost cell generation for distributed unstructured grids. In *2017 IEEE 7th Symposium on Large Data Analysis and Visualization (LDAV)* (2017), IEEE, pp. 84–91. 3

[PRN*11] PETERKA T., ROSS R., NOUANESENGSY B., LEE T.-Y., SHEN H.-W., KENDALL W., HUANG J.: A study of parallel particle tracing for steady-state and time-varying flow fields. In *Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International* (2011), IEEE, pp. 580–591. 2, 3

[PVH*03] POST F. H., VROLIJK B., HAUSER H., LARAMEE R. S., DOLEISCH H.: The state of the art in flow visualisation: Feature extraction and tracking. In *Computer Graphics Forum* (2003), vol. 22, Wiley Online Library, pp. 775–792. 2

[SH95] STALLING D., HEGE H.-C.: Fast and resolution independent line integral convolution. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), ACM, pp. 249–256. 2, 3

[Sim91] SIMON H. D.: Partitioning of unstructured problems for parallel processing. *Computing systems in engineering 2*, 2-3 (1991), 135–148. 3

[SJWS08] SALZBRUNN T., JANICKE H., WISCHGOLL T., SCHEUERMANN G.: The state of the art in flow visualization: Partition-based techniques. 2

[SK97] SHEN H.-W., KAO D. L.: Uflic: a line integral convolution algorithm for visualizing unsteady flows. In *Proceedings of the 8th conference on Visualization'97* (1997), IEEE Computer Society Press, pp. 317–ff. 3

[WCM12] WEBER G. H., CHILDS H., MEREDITH J. S.: Efficient parallel extraction of crack-free isosurfaces from adaptive mesh refinement (amr) data. In *Large Data Analysis and Visualization (LDAV), 2012 IEEE Symposium on* (2012), IEEE, pp. 31–38. 3

[WTS*05] WEINKAUF T., THEISEL H., SHI K., HEGE H.-C., SEIDEL H.-P.: Extracting higher order critical points and topological simplification of 3d vector fields. In *Visualization, 2005. ViS 05. IEEE* (2005), IEEE, pp. 559–566. 2

[YWM07] YU H., WANG C., MA K.-L.: Parallel hierarchical visualization of large time-varying 3d vector fields. In *Proceedings of the 2007 ACM/IEEE conference on Supercomputing* (2007), ACM, p. 24. 2

[ZGH*18] ZHANG J., GUO H., HONG F., YUAN X., PETERKA T.: Dynamic load balancing based on constrained kd tree decomposition for parallel particle tracing. *IEEE transactions on visualization and computer graphics 24*, 1 (2018), 954–963. 3

[ZSH97] ZÖCKLER M., STALLING D., HEGE H.-C.: Parallel line integral convolution. *Parallel Computing 23*, 7 (1997), 975–989. 3