Meshing the Universe: Identifying Voids in Cosmological Simulations Through In Situ Parallel Voronoi Tessellation

Tom Peterka* Juliana Kwan Adrian Pope Hal Finkel Katrin Heitmann
Salman Habib
Argonne National Laboratory

ABSTRACT

Mesh tessellations are effective constructs for the visualization and analysis of point data, because they transform sparse discrete samples into dense and continuous functions. We present a prototype method for computing a Voronoi tessellation in parallel from large particle datasets; the same method, in principle, is applicable to the Delaunay. Computing large tessellations is computationally intensive and must be constructed in parallel on a distributed-memory supercomputer in order to satisfy time and memory constraints. We perform the mesh computation and analysis in situ with the simulation in order to minimize storage pressure and generate early results, specifically identifying voids in cosmological data. We demonstrate performance and scalability in a single time step, and we also compute time-varying tessellations to better understand the temporal dynamics of voids.

Index Terms: D.1.3 [Programming Techniques]: Concurrent Programming—Parallel Programming; H.3.4 [Information Storage and Retrieval]: Systems and Software—Distributed Systems; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, Surface, Solid, and Object Representations

1 INTRODUCTION

Dark matter is thought to account for over 80% of the matter content of the universe: its existence is inferred from a number of observations involving Cosmic Microwave Background [12], galactic dynamics [25] and gravitational lensing [14]. Indeed, indirect detections inform us that dark matter is the backbone of large scale structure in the universe, which drives the distribution of the gas and galaxies that we can observe.

We can simulate the nonlinear time evolution of the universe to a high precision using dark matter tracer particles that experience Newtonian gravitational forces in a Friedmann-Lemaître-Robertson-Walker background. The structures formed by particles can be classified into four main categories: halos, voids, filaments, and walls.

Methods to identify halos have been successfully implemented since Davis et al. [6] first developed the Friends-of-Friends algorithm, but void identification remains an elusive task because of the unpredictable and irregular shapes that voids can occupy. Adaptive methods such as Voronoi tessellation are suited to this task, because cells are determined by point distributions rather than by assuming a priori shapes. Furthermore, while Voronoi cells are convex, their tessellation into larger structures can produce arbitrary and concave regions.

We present the computation of a Voronoi tessellation in parallel on HPC architecture in situ with a cosmological simulation of dark matter in the universe. N-body methods are used to simulate many phenomena, and the analysis of particle simulations and some experimental observations is enabled by imposing a geometric structure on the field data. Sampling onto a regular grid is one common method, but when the density distribution of particles must be preserved, polyhedral tessellations such as the Delaunay and Voronoi are advantageous.

Most existing algorithms for computing convex hulls, Delaunay triangulations, and Voronoi diagrams are serial or shared-memory, apart from [2]. Our contribution is a general-purpose tessellation library for parallelizing the Quickhull algorithm [1], and we demonstrate its scalability in situ with a cosmological simulation. We show that a simple minimum volume threshold can be used to partition Voronoi cells into connected components that correspond to voids of irregular, concave geometries. We benchmark the performance of our method at scale, with simulations of one billion particles running on thousands of processes.

Our construction of geometric structures is not limited to a particular domain. Other areas that would benefit from our approach include molecular dynamics, computational chemistry, groundwater transport, and materials science.

2 BACKGROUND

Formally, each Voronoi region is defined as:

$$V_i = \left\{\mathbf{x} | d(\mathbf{x}, \mathbf{x_i}) < d(\mathbf{x}, \mathbf{x_j})\right\} \forall j \neq i, \tag{1}$$

where $d(\mathbf{x}, \mathbf{x_i})$ is the distance between points $\mathbf{x}$ and $\mathbf{x_i}$. In other words, given a set of n 3D input points called *sites*, the volume of

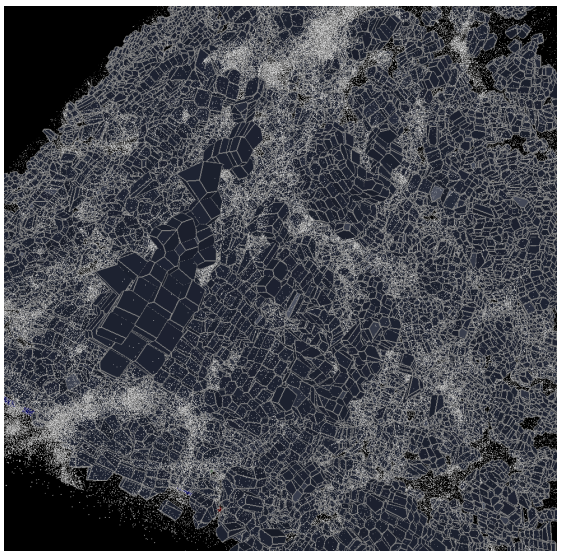

Figure 1: Voronoi tessellation of cosmological simulations filtered at a volume threshold reveals regions of irregular low-density voids amid clusters of high-density halos.

*e-mail: tpeterka@mcs.anl.gov

each Voronoi cell is formed by partitioning the 3D space into disjoint regions that are closest to a particular site than to all other sites. This produces a set of n convex, volume filling, polyhedra called Voronoi cells containing one of the input points somewhere in the interior of each cell. The dual of the Voronoi tessellation is the Delaunay tessellation, where the space is partitioned into tetrahedra whose vertices are the input points.

2.1 Feature Identification in Cosmology

Being able to identify features of the cosmic web is an important part of understanding the large scale structure in the universe, because this additional shape information allows us to probe beyond the traditional 2-point statistics, such as the power spectrum and the correlation function. Furthermore, tracking the shapes of these objects can give a more detailed understanding of the formation history of small scale structures beyond the halo model, in which overdensities are assumed to be spherical. The anisotropic distribution of tracer particles implies that a reconstruction of the density and velocity fields should ideally be adaptive. Delaunay and Voronoi methods adapt by adjusting the resolution of the reconstruction in response to the local number of particles, since each particle is a Voronoi site or Delaunay vertex.

Weygaert et al. [33] overviews the advantages of using tessellation-based methods in cosmology as opposed to a fixed grid. The ZOBOV void finder [17] begins with a Delaunay Tessellation Field Estimator (DTFE) [27]. The Watershed Void Finder [20] attempts to locate voids by using the DTFE algorithm to first reconstruct the density field, and then local minima are connected together at some density threshold. The procedure is analogous to filling a landscape with water with the valleys acting as voids and the ridges between valleys as filaments and walls. Shandarin et al. [28] combines tessellations with multistream techniques to identify Zel'dovich pancakes for the first time in N-body simulations.

Sheth and Weygaert [29] present a model of void evolution over time and conclude that in low-density regions, larger voids are formed by the merger of smaller voids, similar to the evolution of halo amalgamation. Furthermore, they show that in high-density regions, voids disappear over time, being overtaken by the high density collapse that forms halos. Weygaert [31] also evaluates statistics and distributions of the resulting Voronoi tessellations and correlates clustering to Voronoi vertices.

Cosmological structures have been identified using other means than tessellations. Slices and spherical regions are easier to compute but do not represent the actual cosmological structure, as documented by the variety of answers presented by Colberg et al. [5]. The ORIGAMI algorithm [9] identifies morphological structures by counting the number of shell crossings produced whenever a particle exchanges its position with another particle in an earlier time step, in 6D phase space. Tracer particles are collisionless and travel in multistream flows in high density halo regions, so the identification and tracking of multistream features in the particle velocity is another way to locate halos [23].

In terms of observations, these techniques have been successfully applied on the PSCz survey [24] and the Sloan Digital Sky Survey [21] to reconstruct the density and velocity fields from a distribution of galaxies.

Besides being used for analysis, tessellations can be incorporated into the N-body calculations themselves: in hydrodynamics simulations, Springel [30] used a Voronoi tessellation to convert Lagrangian particle behavior to a moving mesh. The performance of this code in replicating complicated gas phenomena such as shocks and the Kelvin-Helmholtz instability is far superior to fixed resolution methods such as smoothed particle hydrodynamics [10, 13].

2.2 Computational Geometry Algorithms

Quickhull is a serial algorithm for computing convex hulls, from which Delaunay triangulations and Voronoi diagrams are derived. [1]. It is robust in the presence of imprecise floating-point inputs and improves over the performance of earlier algorithms such as Clarkson [4].

CGAL (Computational Geometry Algorithms Library) [8] is an alternate implementation that can compute a Voronoi diagram given a set of input sites. Unlike Quickhull, a direct implementation of a 3D Voronoi tessellation does not exist; the Delaunay graph must be computed first, from which the dual is calculated to produce Voronoi vertices.

Parallel computational geometry algorithms of limited dimension and concurrency have been researched as well. Rycroft [26] published and implemented a parallel Voronoi algorithm for shared-memory threads in the Voro++ library. Miller and Stout examined parallel algorithms for a convex hull of 2D points, tuned for various network topology [15]. Dehne et al. [7] presented a parallel 3D convex hull algorithm for distributed memory architecture in $O(nlogn)$ local computation and one communication phase.

2.3 HACC Framework

HACC, or Hardware Accelerated Computational Cosmology, is a simulation framework for computing cosmological models on various supercomputing architectures such as GPUs, Cell Broadband Engines, and multicore CPUs. It solves a 6-D N-body nonlinear Vlasov-Poisson equation with periodic boundary conditions in three regimes. Further details of the algorithm are contained in [22].

Very large simulation sizes are required in order to compute even a fraction of the sky, requiring hundreds of billions to trillions of particles for accurate power spectrum measurements [11] and generating data sizes of over 100 TB per run. The spatial dynamic range required to compute accurate structure formation is $10^6 : 1$. HACC also includes in situ analysis capability for finding halos and subhalos [35], which can be stored and later examined in post-processing tools such as ParaView. These analysis output files are much smaller than the original particle data, and can also be used to track the evolution of halos over time.

3 Method

3.1 Program Organization

Rather than developing a new parallel algorithm from the ground up, our approach is to take an existing serial algorithm and parallelize it by combining local computation with communication. Thus, we build on the many years of computational geometry research embodied in existing and mature tools. We selected the Quickhull algorithm because of its widespread use, documented performance, and numerical stability. The Quickhull algorithm is implemented in an open-source package called Qhull.[1] Qhull is a set of standalone command-line programs for computing convex hulls, Delaunay triangulations, and Voronoi tessellations that call the underlying `libqhull` library. The standalone programs are well-documented, but the library API is not. We examined Qhull's printing routines so that we could parse its data structure; and once we understood Qhull's data model and copied its output into our own data structures, we redirected its output to `/dev/null`. No changes were made to Qhull itself.

While Qhull contains libraries for C and C++, the C version is robust and recommended. Figure 2 shows the combination of C and C++ codes in our final system, as well as the overall program structure. We call our library *tess*, and its main features are:

- Standalone as well as in situ domain decomposition
- Neighborhood particle ghost zone exchange

[1] http://www.qhull.org

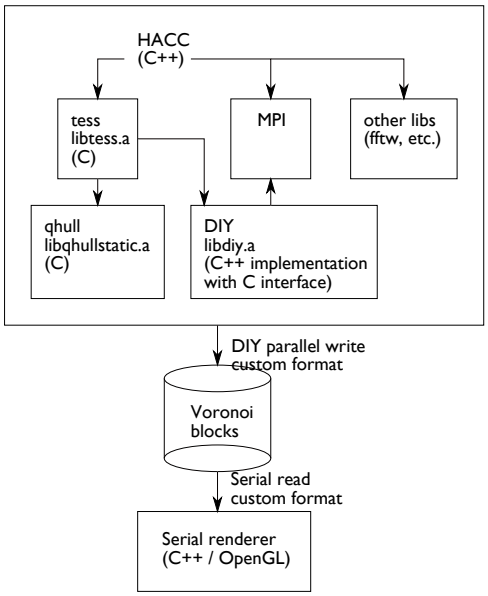

Figure 2: The software structure is a combination of C and C++ code.

- Local Voronoi cell computation
- Identification of complete cells
- Early volume threshold culling
- Local convex hull computation for faces, areas, and volumes of cells
- Parallel write of Voronoi cells to storage
- Offline serial reader and renderer

The HACC framework contains some analysis capability directly in the code, but tess is linked to HACC as a separate library with its own parallel infrastructure. This modularity affords different analyses to be executed depending on the goals of the simulation campaign. Parallelizing a serial algorithm like Quickhull is done with the aid of a library called DIY [19]. This library provides utilities for configurable data partitioning, scalable data exchange, and efficient parallel I/O. DIY is initialized with information from HACC about its block decomposition and neighborhood connectivity, and then DIY performs data movement and communication on behalf of tess. Developing tess required two new features to be added to DIY: neighborhood periodic boundary conditions and reduced neighborhood communication (see Section 3.4).

3.2 Analysis Data Model

In tess, each process maintains a data structure for the block of Voronoi cells local to it; these blocks are written in parallel to a single file by the DIY library. Each block contains a conventional unstructured mesh data model. Vertices are listed once, and integer indices connect vertices into faces and cells. Original particle locations (Voronoi sites) are also saved, as well as cell volumes, areas, and block extents.

On average, Voronoi cells in HACC contain 15 faces per cell and 5 vertices per face. Each cell consists of approximately 35 total vertices, since vertices are shared between at least two faces in the same cell. Vertices are also shared among five cells on average; hence, approximately seven new Voronoi vertices are added for each new cell in a full tessellation.

The total data size of a full tessellation is approximately 450 bytes per particle. As we will see in Section 4, a large fraction of cells have insignificant volume with respect to voids. When we choose to eliminate these, as we often do, the data size is reduced to approximately 100 bytes per particle. By comparison, a HACC checkpoint that saves only particle data uses 40 bytes per particle.

Of tess' total output size, approximately 7% is used to store floating-point geometry of vertices, particles, volumes, and areas. The remaining 93% is used for connectivity of the mesh. A more efficient data structure for general polyhedral grids has been published by Muigg et al. [16], and we are investigating its use.

3.3 Parallel Voronoi Algorithm

Our approach exchanges a ghost zone of particles prior to computing Voronoi cells locally. Hence, we rely on an approximation of the width of this ghost zone, and a block size that is several times larger. Both of these conditions are satisfied in HACC; average cell size is on the order of initial particle spacing, and block size is approximately ten times that distance.

Algorithm 1 Parallel Voronoi Algorithm

1: Get decomposition and input particles from simulation
2: Exchange ghost zone of particles in one direction only
3: Generate Voronoi tessellation locally
4: **for all** cells **do**
5: Delete incomplete cells and cells whose vertices are outside overall domain boundaries
6: Delete cells smaller than circumscribing sphere of threshold volume
7: Compute convex hull, faces, areas, and volumes
8: Deleted cells smaller than threshold volume
9: **end for**
10: Parallel file write of original sites and local Voronoi cells

The steps in our algorithm are illustrated in Figure 3 and listed in Algorithm 1. The first step is initialization, explained in Section 3.5. In step 2, particles are exchanged as explained in Section 3.4. The local Voronoi tessellation is computed in step 3 using the Quickhull algorithm.

The Voronoi cells are each examined in steps 4-9. Incomplete cells are eliminated; those cells are not closed and lack particles surrounding them on all sides, but the preceding particle exchange ensured that the same cell was completed in another block. Occasionally invalid cells with vertices out of the overall domain bounds arise; a simple bounds check eliminates these as well.

As Section 4 shows, a large fraction of cells have small volume and may be of no interest, especially for void finding. These will be eliminated once their volume is accurately computed, but in step 6 we perform a conservative quick estimate of the cell volume and eliminate as many cells as early as possible. We only keep the cells whose distance between any two vertices exceeds the diameter of a circumscribing sphere of the threshold volume.

In step 7, we call the Quickhull algortihm again to compute the convex hull of the vertices in the Voronoi cell. Voronoi vertices are convex by definition, but the convex hull alorithm orders the vertices into faces and computes the volume and surface area of the cell. Afterwards, we recheck the volume and further cull cells below the volume threshold, writing the remaing data to storage in parallel according to the data model in Section 3.2.

3.4 Neighborhood Particle Exchange

Two new communication patterns were added to DIY as a result of this research: periodic boundary neighbors and asymmetric particle exchange. HACC's nearest-neighbor connectivity includes periodic boundaries, meaning that blocks at one edge of the entire domain have neighbors at the opposite edge of the domain. Also, of those neighbors within the ghost zone distance from a particle, it is sufficient to send the particle in only one direction, say the minus direction, in all axes. No more than half of the neighbors need to receive

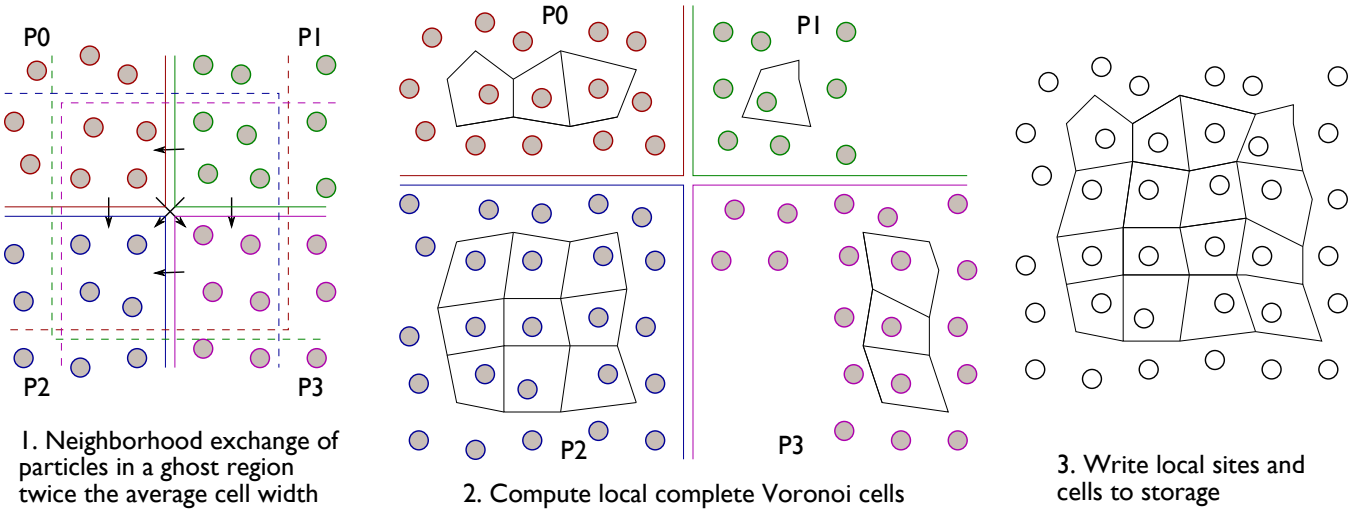

Figure 3: Steps to resolve Voronoi cells at block boundaries. 1. Particles are exchanged in ghost regions among four processes marked with solid lines, dashed lines indicating expanded ghost regions. Arrows indicate direction of exchange. 2. Local voronoi cells are computed at each process. 3. Completed cells and original particles are saved to storage in parallel. Data remain distributed throughout the process.

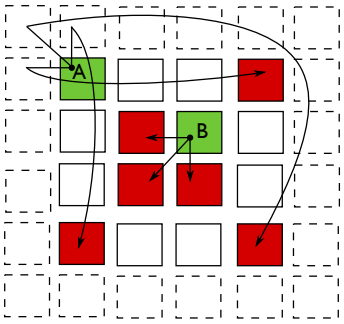

Figure 4: Neighborhood communication with periodic boundary conditions and neighbors that are near enough to a target point. Green blocks denote sources, and red blocks denote destinations. Particle A is at the domain boundary and is sent to the virtual neighbors marked with dashed lines, but actually on the other side of the domain. The coordinates of the particle are transformed accordingly. Particle B is sent to actual neighbors near enough to receive it.

a particle, such that the same Voronoi cell is not computed twice by two different blocks.

These ideas are diagrammed in Figure 4. Particles originate in the green blocks and follow the indicated paths to the red blocks. Particles within the ghost zone distance are exchanged with all neighbors in the minus direction of each axis. If the neighbor is a periodic boundary neighbor, the particle is translated in each of the periodic dimensions to the other side of the domain.

3.5 Calling Tess In Situ

Tess is built as a standalone library; it and DIY are statically linked to HACC via small modifications to the HACC build system. In the future, tess and other in situ analysis tools will be included as HACC build options and called through a common API for analysis tools. For the time being, calls to tess are inserted directly into the main time loop of the HACC simulation driver. Initialization is done as follows:

```
tess_init(num_blocks, tot_blocks, gids,
bounds, neighbors, num_neighbors,
cell_size, ghost_factor, data_mins,
data_maxs, wrap, min_vol,
max_vol, comm, times)
```

The number of blocks per process (`num_blocks`) is always one in HACC and total number of blocks in the domain (`tot_blocks`) is the number of processes in HACC. The `gids` are global block identifiers for the local blocks, in HACC the MPI rank of the process. Bounding boxes for the local blocks are in `bounds`. The list of neighboring blocks is provided in `neighbors` and `num_neighbors`. There are 26 neighbors for each block in HACC, including periodic boundary neighbors. The `cell_size` and `ghost_factor` control the size of the neighborhood ghost zone. `Data_mins` and `data_maxs` are the overal domain bounds. The `wrap` argument enables periodic boundary neighbors. `Min_vol` and `max_vol` are optional volume limits on the Voronoi cells that are returned. `Comm` is the MPI communicator, and performance timing is returned in `times`.

Once initialized, tess is called as frequently as desired by providing the particles, number of particles, and output file name:

```
tess(particles, num_particles, out_file)
```

4 Results

Our small scale tests were run on a Linux desktop workstation with a quad-core Intel i7 processor capable of running eight hardware threads with 12 GB of RAM. Beacuse HACC, tess, and DIY scale across different architecture sizes and types, we were able to test for correctness, albeit not performance, at small scale.

Our larger tests were run on *Intrepid*, a 557-teraflop IBM Blue Gene/P (BG/P) supercomputer operated by the Argonne Leadership Computing Facility (ALCF) at Argonne National Laboratory. Tests were run in symmetric multiprocessor mode (one MPI process per node) to maximize available memory per process; HACC and tess are memory-size bound. The libraries and simulation were all compiled with the IBM `xlcxx_r` compiler using `-O3 -qarch=450d -qtune=450` optimizations.

In our HACC configuration, we varied the number of particles from 32^3 to 1024^3, up to one billion particles. The particles are initialized on a grid of the same number of grid points (*ng*) per dimension as number of particles (*np*). The physical size of the simulation

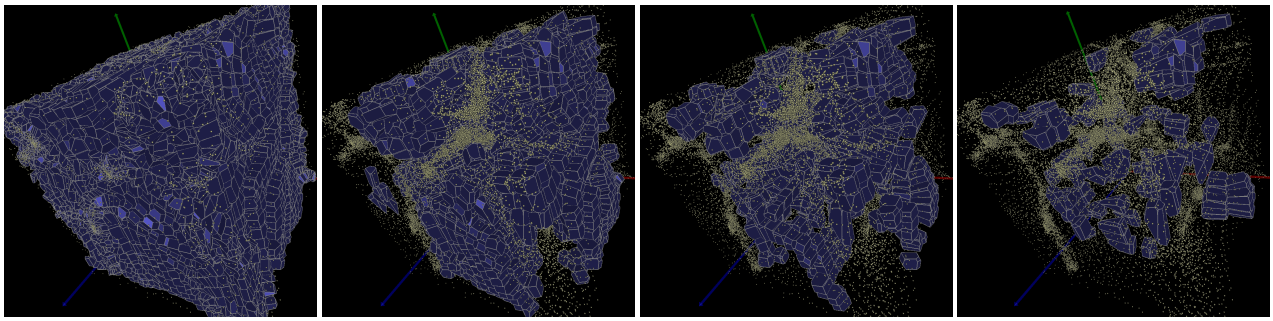

Figure 6: Culling cells below a minimum volume threshold reveals connected components of cells, which constitute voids. Left to right: original cells ranging from 0.0001 to 2.005 $(Mpc/h)^3$, and progressing through minimum volume threshold of 0.0, 0.5, 0.75, and 1.0 $(Mpc/h)^3$, respectively.

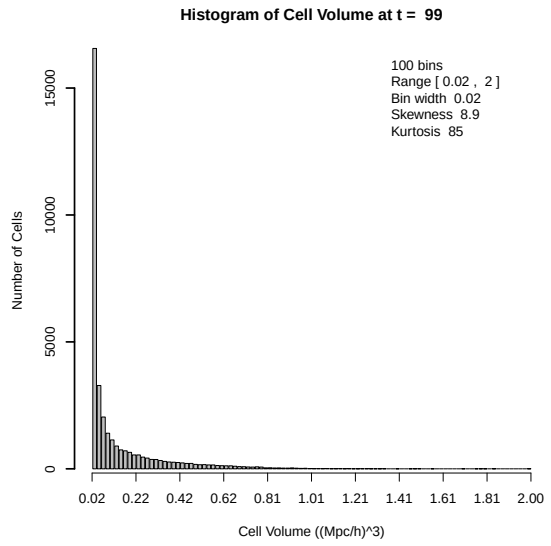

Figure 5: Histogram of cell volume distribution in small scale test.

box is also the same as ng and np; hence, particles begin spaced 1 Mpc/h apart in each dimension, where h is a scaling factor.

4.1 Small Scale Exploration

Our intial testing consisted of the 32^3 particle simulation run on a Linux workstation up to 32 processes. Particles evolved for 100 time steps, and the Voronoi tessellation was computed at the end of the last time step. Several characteristics of the Voronoi tessellation applicable to larger scale were uncovered at this small scale.

The distribution of Voronoi cell volume after 100 time steps appears in Figure 5. The distribution exhibits extreme skew in the positive direction, with the majority of cells at the left side and a long thin tail at the right. In fact, 75% of the cells are in smallest 10% of the volume range.

This characteristic distribution holds in all of our larger scale runs and indicates that a simple threshold operator can dramatically reduce the number of cells. Voids by definition are regions of low particle density; small, dense cells appear in the peak of the distribution, and larger low-density cells are found in the long tail at the right. Emprically, we found that a 10% volume threshold is a reasonable value that eliminates many small, uninteresting cells while safely retaining all of the cells that contribute to voids.

Thresholding serves a second purpose besides reducing output size. We can further filter cells during either computation or rendering to reveal voids more clearly. Figure 6 shows a sequence of progressive thresholding during rendering. From left to right, culling cells below an increasing volume threshold reveals connected components of cells that correspond to voids.

In the left image, the difference in cell size is visible, but the connection of cells into larger structures is not. The other images reveal a small number, approximately 7-10, distinct connected components, or voids. In our current implementation, the cosmologist selects the minimum volume threshold empirically based on the cell volume distribution and previous experience, and the grouping of the remaining cells into voids appears visually. In the future, we plan to identify connected components automatically in situ.

4.2 Medium Scale Performance

Figure 1 shows a tessellation of 128^3 = 2 M particles generated in situ on BG/P. Full timing results for a range of problem and system sizes appear in Table 1, which shows the performance of running one Voronoi tessellation in situ after running a number of simulation time steps on BG/P. This test was run with culling the smallest 10% of volume range of the Voronoi cells, resulting in approximately four times reduction in file size. The file sizes shown in the last column of Table 1 indicate the culled file size.

The tessellation time is 1 - 10% of the total run time, depending on the number of time steps executed before calling tess. The cost of tessellation compared with simulation is reasonable, especially considering that HACC takes longer to compute later time steps. Strong scaling efficiency for tess is between 30-40%, including I/O, consistent with the simulation strong scaling. Aggregate I/O bandwidth up to 5 GB/s was measured.

The bottlenecks are the Voronoi tessellation time and to a lesser extent the convex hull from each Voronoi cell. These are computationally intensive and subject to load imbalance, as the range of times indicates. We are investigating whether it makes sense to redistribute particles to better balance load, but this discussion led us to think about whether HACC itself should employ dynamic rebalancing, which it currently does not. It is also possible that our asymmetric neighbor exchange affects load balance. We will investigate this further.

4.3 Time-varying Void Evolution

Tess can generate tessellations at any number of time steps in the simulation, so that we can study the evolution of voids over time. We tested the time-varying capability by producing output at every ten time steps of a simulation consisting of 100 time steps. This test was at small scale, on 32^3 particles on our workstation, with the results in Figure 7. The left column shows histograms of cell volume at time steps 11, 21, 31, and 41. The center column shows the cell density contrast (δ) distribution, because much theory has

Table 1: High Resolution Performance Data

Particles	Time Steps	Processes	Total Time (s)	Simulation Time (s)	Tessellation Time (s)	Particle Exchange Time (s)	Voronoi Time [min, max] (s)	Convex Hull Time [min, max] (s)	Output Time (s)	Output Size (GB)
128^3	100	128	1862	1809	53	1	[5, 30]	[9, 19]	2	0.3
		256	1354	1322	32	1	[2, 18]	[2, 10]	2	
		512	1116	1096	20	1	[1, 12]	[1, 5]	2	
		1024	745	729	16	1	[1, 9]	[1, 3]	3	
256^3	100	512	3090	3016	74	2	[9, 56]	[9, 12]	3	1.7
		1024	2391	2346	45	2	[4, 33]	[3, 6]	4	
		2048	1861	1830	32	2	[2, 23]	[1, 3]	4	
		4096	1334	1305	29	2	[1, 14]	[1, 2]	12	
512^3	50	2048	3852	3684	167	4	[24, 116]	[25, 40]	6	14
		4096	2008	1918	89	3	[9, 57]	[12, 20]	9	
		8192	1784	1722	62	3	[4, 38]	[5, 10]	11	
		16384	1406	1344	61	2	[1, 27]	[1, 5]	27	
1024^3	25	8192	2331	2119	212	6	[49, 176]	[2, 10]	20	101
		16384								
		32768								

been developed in terms of density contrast. It is the difference between the cell density (d) and mean density (μ_d), normalized by the mean.

$$\delta = (d - \mu_d)/\mu_d \quad (2)$$

The right column shows images of the tessellations at these same time steps.

The early time steps begin with a normally distributed cell size and shape, because particles begin their evolution on regular grid points. As time progresses, the range of volume and density expands. The kurtosis increases as the distributions become more pointed, and skewness increases as well. These statistics correspond to the breakdown of perturbation theory governing the physics. In the future, we will investigate whether such summary statistics can be used as a simple indicator of the change in the physical model behavior. The images in the right column confirm the behavior of the volume and density contrast distributions.

The findings of Sheth and Weygaert [29] are further confirmed in Figure 7. (Time step and red shift are synonomous in this context.) As the left column shows, cells grow in size with red shift, as the increasing volume range shows. The quantity of smaller cells (height of the spike) decreases while the tail grows to the right of the distribution, consistent with small voids coalescing into larger ones. The center column is consistent with the theory that voids in high-density regions disappear with increasing red shift. The density contrast range increases with red shift, corrresponding to the formation of high and low density structures over time.

5 Summary

We presented a solution for computing scalable parallel tessellations and demonstrated its in situ application to cosmological compuations. This produced statistical summaries of volume and density distributions and the visual identification of voids. Performance was benchmarked on medium-scale problems and system sizes, with good scalability. Our time-varying results were consistent with earlier findings.

We have only presented a cursory example of where an in situ Voronoi tessellation would be advantageous over a grid based fixed-resolution approach. Our approach can be extended to investigate void and genus statistics over a range of cosmologies as a means of distinguishing between competing cosmological models. This has been proposed in [18, 32, 34, 36] as a precision probe cosmology, but has never been verified with the full nonlinear N-body solution.

It would also be interesting to perform these reconstructions with halos as Voronoi sites instead of directly using the particle distribution, since halos can be matched to direct observables such as galaxies. This would involve smaller, prefiltered data and a combination of in situ analysis techniques.

Of course, there are also improvements that could be made to the algorithm itself, such as organizing vertices into faces and cells using a more compact data structure and computing the volume faster than Qhull's convex hull algorithm. We also plan to coalesce cells that have been prefiltered into larger structures (no longer convex hulls), connect them together logically through connected component labeling, remove their interior faces to save space, and compute volumes and other statistics on them. These connected components will also need to adhere to periodic boundary conditions. We will also look to tracking temporal evolution of connected components using the feature tree method of Chen et al. [3].

Our rendering application is a small serial prototype. Improvements can be made by using parallel rendering with a more efficient and standard mesh model. Our next version of the viewer will probably be written with a combination of DIY and VTK, and include parallel rendering performance, additional visualization methods such as volume rendering, and a richer user interface.

Acknowledgements

We gratefully acknowledge the use of the resources of the Argonne Leadership Computing Facility at Argonne National Laboratory. This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357. Work is also supported by the DOE Office of Science, Advanced Scientific Computing Research award No. DE-FC02-06ER25777, program manager Lucy Nowell.

References

[1] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The Quickhull Algorithm for Convex Hulls. *ACM Trans. Math. Softw.*, 22:469–483, Dec. 1996.

[2] M. C. Cautun and R. van de Weygaert. The DTFE Public Software - The Delaunay Tessellation Field Estimator Code. *ArXiv e-prints*, May 2011.

[3] J. Chen, D. Silver, and L. Jiang. The Feature Tree: Visualizing Feature Tracking in Distributed AMR Datasets. In *Proceedings of the*

2003 IEEE Symposium on Parallel and Large-Data Visualization and Graphics, PVG '03, pages 14–, Washington, DC, USA, 2003. IEEE Computer Society.

[4] K. L. Clarkson. Applications of Random Sampling in Computational Geometry, II. In *Proceedings of the fourth annual symposium on Computational geometry*, SCG '88, pages 1–11, New York, NY, USA, 1988. ACM.

[5] J. M. Colberg, F. Pearce, C. Foster, E. Platen, R. Brunino, M. Neyrinck, S. Basilakos, A. Fairall, H. Feldman, S. Gottloeber, O. Hahn, F. Hoyle, V. Mueller, L. Nelson, M. Plionis, C. Porciani, S. Shandarin, M. S. Vogeley, and R. van de Weygaert. The Aspen–Amsterdam Void Finder Comparison Project. Technical Report arXiv:0803.0918, Mar. 2008.

[6] M. Davis, G. Efstathiou, C. S. Frenk, and S. D. M. White. The Evolution of Large-Scale Structure in a Universe Dominated by Cold Dark Matter. *The Astrophysical Journal*, 292:371–394, May 1985.

[7] F. Dehne, X. Deng, P. Dymond, A. Fabri, and A. A. Khokhar. A Randomized Parallel 3D Convex Hull Algorithm for Coarse Grained Multicomputers. In *Proceedings of the seventh annual ACM symposium on Parallel algorithms and architectures*, SPAA '95, pages 27–33, New York, NY, USA, 1995. ACM.

[8] A. Fabri and S. Pion. CGAL: the Computational Geometry Algorithms Library. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '09, pages 538–539, New York, NY, USA, 2009. ACM.

[9] B. L. Falck, M. C. Neyrinck, and A. S. Szalay. ORIGAMI: Delineating Halos using Phase-Space Folds. *ArXiv e-prints*, Jan. 2012.

[10] R. A. Gingold and J. J. Monaghan. Smoothed Particle Hydrodynamics - Theory and Application to Nonspherical Stars. *Monthy Notices of the Royal Astronomical Society*, 181:375–389, Nov. 1977.

[11] K. Heitmann, M. White, C. Wagner, S. Habib, and D. Higdon. The Coyote Universe. I. Precision Determination of the Nonlinear Matter Power Spectrum. *Astrophysical Journal*, 715:104–121, May 2010.

[12] N. Jarosik, C. L. Bennett, J. Dunkley, B. Gold, M. R. Greason, M. Halpern, R. S. Hill, G. Hinshaw, A. Kogut, E. Komatsu, D. Larson, M. Limon, S. S. Meyer, M. R. Nolta, N. Odegard, L. Page, K. M. Smith, D. N. Spergel, G. S. Tucker, J. L. Weiland, E. Wollack, and E. L. Wright. Seven-year Wilkinson Microwave Anisotropy Probe (WMAP) Observations: Sky Maps, Systematic Errors, and Basic Results. *Astrophysical Journal Supplement*, 192:14, Feb. 2011.

[13] L. B. Lucy. A Numerical Approach to the Testing of the Fission Hypothesis. *Astronomical Journal*, 82:1013–1024, Dec. 1977.

[14] M. Markevitch, A. H. Gonzalez, L. David, A. Vikhlinin, S. Murray, W. Forman, C. Jones, and W. Tucker. A Textbook Example of a Bow Shock in the Merging Galaxy Cluster 1E 0657-56. *Astrophysics Journal Letters*, 567:L27–L31, Mar. 2002.

[15] R. Miller and Q. F. Stout. Efficient Parallel Convex Hull Algorithms. *IEEE Trans. Comput.*, 37(12):1605–1618, Dec. 1988.

[16] P. Muigg, M. Hadwiger, H. Doleisch, and E. Groller. Interactive Volume Visualization of General Polyhedral Grids. *IEEE Transactions on Visualization and Computer Graphics*, 17:2115–2124, 2011.

[17] M. C. Neyrinck. ZOBOV: a Parameter-Free Void-Finding Algorithm. *Monthly Notices of the Royal Astronomical Society*, 386:2101–2109, Jun. 2008.

[18] C. Park and Y.-R. Kim. Large-Scale Structure of the Universe as a Cosmic Standard Ruler. *Astrophysics Journal Letters*, 715:L185–L188, Jun. 2010.

[19] T. Peterka, R. Ross, W. Kendall, A. Gyulassy, V. Pascucci, H.-W. Shen, T.-Y. Lee, and A. Chaudhuri. Scalable Parallel Building Blocks for Custom Data Analysis. In *Proceedings of the 2011 IEEE Large Data Analysis and Visualization Symposium LDAV'11*, Providence, RI, 2011.

[20] E. Platen, R. van de Weygaert, and B. J. T. Jones. A Cosmic Watershed: The WVF Void Detection Technique. *Monthy Notices of the Royal Astronomical Society*, 380:551–570, Sep. 2007.

[21] E. Platen, R. van de Weygaert, B. J. T. Jones, G. Vegter, and M. A. A. Calvo. Structural Analysis of the SDSS Cosmic Web - I. Nonlinear Density Field Reconstructions. *Monthy Notices of the Royal Astronomical Society*, 416:2494–2526, Oct. 2011.

[22] A. Pope, S. Habib, A. Lukic, D. Daniel, P. Fasel, N. Desai, and K. Heitmann. The Accelerated Universe: A Hybrid Cosmology Code for Roadrunner. *Computing in Science and Engineering*, 12:17–25, Jul. 2010.

[23] U. Popov, K. Heitmann, J. Ahrens, S. Habib, and A. Pang. The Evolution of Multistreaming Events in the Formation of Large Scale Structures. In *Proceedings of the 2011 IEEE Pacific Visualization Symposium*, pages 207–14, 2011.

[24] E. Romano-Díaz and R. van de Weygaert. Delaunay Tessellation Field Estimator Analysis of the PSCz Local Universe: Density Field and Cosmic Flow. *Monthly Notices of the Royal Astronomical Society*, 382:2–28, Nov. 2007.

[25] V. C. Rubin, D. Burstein, W. K. Ford, Jr., and N. Thonnard. Rotation Velocities of 16 SA Galaxies and a Comparison of Sa, Sb, and Sc Rotation Properties. *Astrophysical Journal*, 289:81–98, Feb. 1985.

[26] C. Rycroft. Voro++: A Three-dimensional Voronoi Cell Library in C++. Technical report, 2009. http://www.osti.gov/energycitations/servlets/purl/946741-A8FxbI/946741.pdf.

[27] W. E. Schaap. *DTFE: The Delaunay Tesselation Field Estimator*. University of Groningen, The Netherlands, 2007. Ph.D. Dissertation.

[28] S. Shandarin, S. Habib, and K. Heitmann. The Cosmic Web, Multi-Stream Flows, and Tessellations. (arXiv:1111.2366), Nov. 2011.

[29] R. K. Sheth and R. van de Weygaert. A Hierarchy of Voids: Much Ado About Nothing. *Monthly Notices of the Royal Astronomical Society*, 350:517–538, May 2004.

[30] V. Springel. Hydrodynamic Simulations on a Moving Voronoi Mesh. *ArXiv e-prints*, Sep. 2011.

[31] R. van de Weygaert. Voronoi Tessellations and the Cosmic Web: Spatial Patterns and Clustering across the Universe. *ArXiv e-prints*, Jul. 2007.

[32] R. van de Weygaert, P. Pranav, B. J. T. Jones, E. G. P. Bos, G. Vegter, H. Edelsbrunner, M. Teillaud, W. A. Hellwing, C. Park, J. Hidding, and M. Wintraecken. Probing Dark Energy with Alpha Shapes and Betti Numbers. *ArXiv e-prints*, Oct. 2011.

[33] R. van de Weygaert and W. Schaap. The Cosmic Web: Geometric Analysis. *ArXiv e-prints*, Aug. 2007.

[34] X. Wang, X. Chen, and C. Park. Topology of a Large-Scale Structure as a Test of Modified Gravity. *Astrophysics Journal Letters*, 747:48, Mar. 2012.

[35] J. Woodring, K. Heitmann, J. Ahrens, P. Fasel, C.-H. Hsu, S. Habib, and A. Pope. Analyzing and Visualizing Cosmological Simulations with ParaView. (arXiv:1010.6128. LA-UR-10-06301), Nov. 2010.

[36] C. Zunckel, J. R. Gott, and R. Lunnan. Using the Topology of Large-Scale Structure to Constrain Dark Energy. *Monthly Notices of the Royal Astronomical Society*, 412:1401–1408, Apr. 2011.

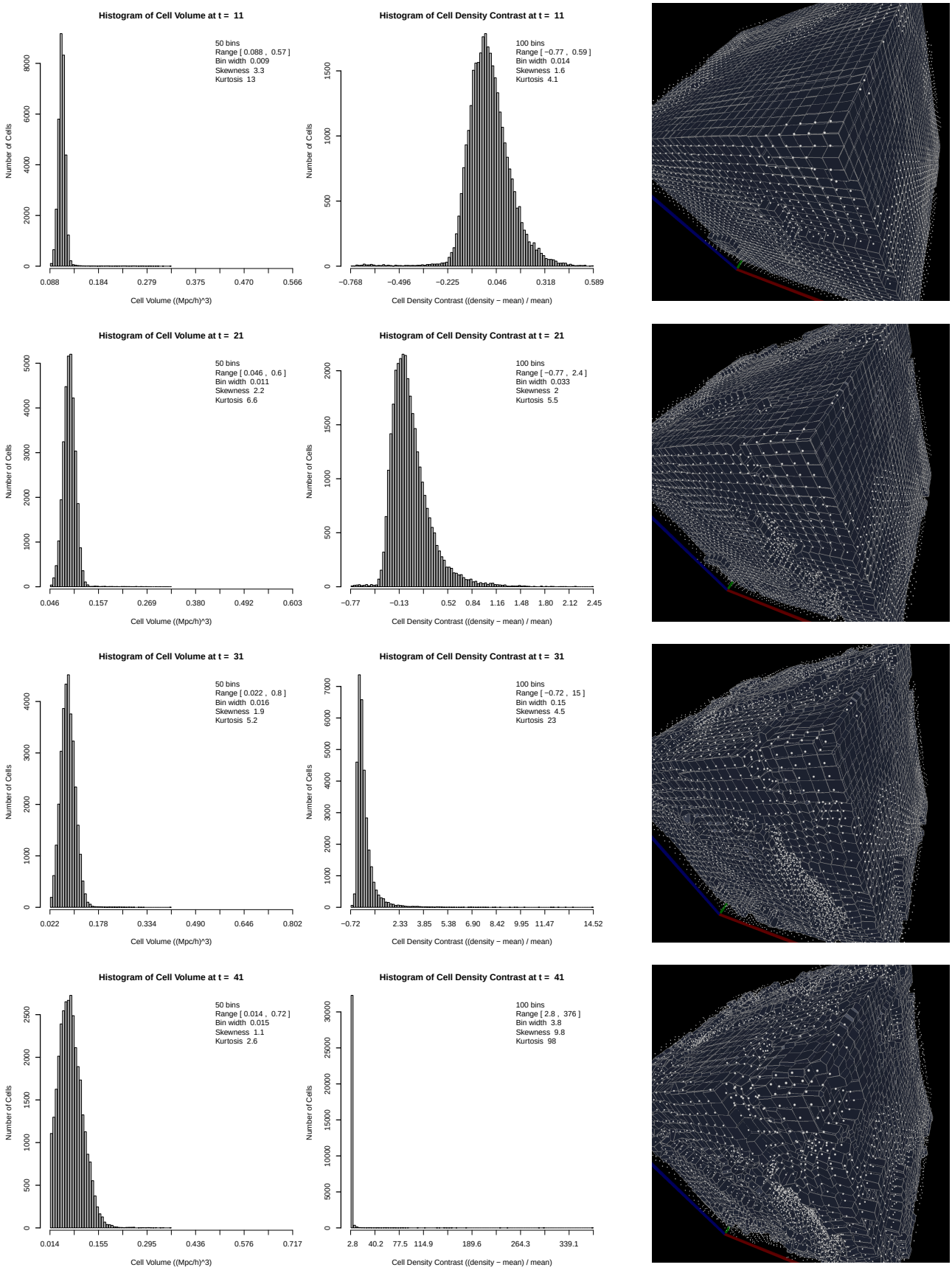

Figure 7: Evolving Voronoi cell volume distribution (left), Voronoi cell density contrast distribution (center), and images of same time steps (right).