

---

# Polytope: A New Parallel Framework for Computing Voronoi Meshes on Complex Boundaries

D. P. Starinshak<sup>1</sup>, J. M. Owen<sup>2</sup>, and J. N. Johnson<sup>3</sup>

<sup>1</sup> Lawrence Livermore National Laboratory, [starinshak1@llnl.gov](mailto:starinshak1@llnl.gov)

<sup>2</sup> Lawrence Livermore National Laboratory, [owen@llnl.gov](mailto:owen@llnl.gov)

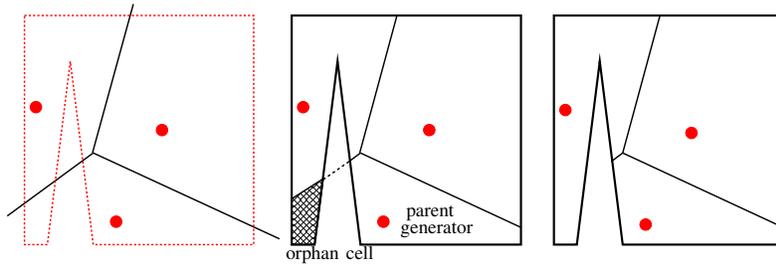
<sup>3</sup> Lawrence Berkeley National Laboratory, [jnjohnson@lbl.gov](mailto:jnjohnson@lbl.gov)

## Introduction

Applications for Voronoi mesh generation are numerous: automatic construction of finite volumes for mesh-free numerical methods [8, 3]; robust mesh structures for models spanning multiple spatial scales in astrophysical hydrodynamics, geophysical flows, and fluid dynamics [2, 8, 3]; convex mesh elements for particle tracking and transport methods; and visualization of discrete, spatially-distributed data.

This article outlines a new platform for computing Voronoi and Voronoi-like meshes in a robust and efficient manner from both shared and distributed input data. Polytope [12] is open-source software designed to provide a uniform interface to several underlying packages for computing Voronoi and Delaunay tessellations. A code using Polytope can experiment with many different tessellation libraries through a single interface, without having to adapt to each package's unique interface and data structures. The code performs internal operations using integers for computational robustness and respects the floating point precision specified by the user. Most importantly, the interface is parallel: Polytope provides a communication layer to serial tessellation packages to transform distributed input points into consistent unstructured mesh data.

Polytope supports unbounded meshes in 2D and 3D and bounded meshes in 2D. A clipping algorithm is used to generate boundary-conformal meshes that can handle complicated geometries, including non-convex boundaries containing holes. The capability to generate three-dimensional bounded meshes is forthcoming.



**Fig. 1.** Unbounded Voronoi cells before clipping (left); orphaned cell created by clipping (center); neighboring cell “adopts” orphan based on VSP principle (right).

## 1 The Serial Algorithm

Polytope currently interfaces direct Voronoi libraries: Voropp [4] and the Voronoi capabilities in Boost.Polygon [10]; and Delaunay solvers: Triangle [5] and Tetgen [7]. Polytope builds unstructured mesh data from the topology of computed Voronoi and Delaunay diagrams. For Delaunay diagrams, Polytope constructs the Voronoi dual internally to exact floating point precision using Shewchuk’s predicates [6].

Mesh cells are initially unbounded and may contain infinite edges. If boundary data is supplied, the unbounded mesh is clipped, in the spirit of [11] to form a boundary-conformal elements. Pertinent aspects include:

### *Cell Clipping*

Polytope’s clipping algorithm computes polygon-polygon intersections using the C++ computational geometry library Boost.Geometry [1]. Each cell from the computed Voronoi diagram is stored as a Boost polygon and intersected with the faceted input boundary. Intersections are performed using integers to handle degenerate cases in a robust and reliable manner.

### *Orphaned Cells*

Intersecting a Voronoi cell with a non-convex boundary is not guaranteed to return a single polygon, especially if generator density is low local to a non-convex feature or hole. (See Figure 1.) One polygon will contain the original generator and become the new boundary-conformal mesh cell. The remaining polygons are orphaned by the parent Voronoi cell and must be accounted for. Polytope redistributes the area occupied by orphaned cells to neighboring cells by constructing sub-tessellations from neighboring generator data.

### *Degenerate Mesh Nodes*

Voronoi mesh generation tools are often plagued by the existence of degenerate states. Failure to recognize degenerate Voronoi vertices can result in undesirably small edges and inconsistent mesh topologies. Polytope maps floating

point vertex data to an integer grid having a user-defined minimum length scale. Vertices within one grid spacing of one another are merged, and the topology of the underlying Voronoi diagram is updated accordingly.

## 2 The Parallel Algorithm

Polytope provides an MPI communication layer around serial tessellation libraries to handle distributed input data. The parallel algorithm established communication between spatially-distributed (and not necessarily disjoint) domains. Generator data is communicated with neighbor domains such that the local mesh on each compute core is consistent with the global set.

The details of the parallel algorithm are beyond the scope of this article (see [9]). It has been rigorously tested in 2D with robust scaling up to tens of millions of input points distributed across 256 processors. Initial tests on parallel efficiency are favorable. Optimization of the algorithm is forthcoming.

## 3 Results

### *Some Example Boundaries*

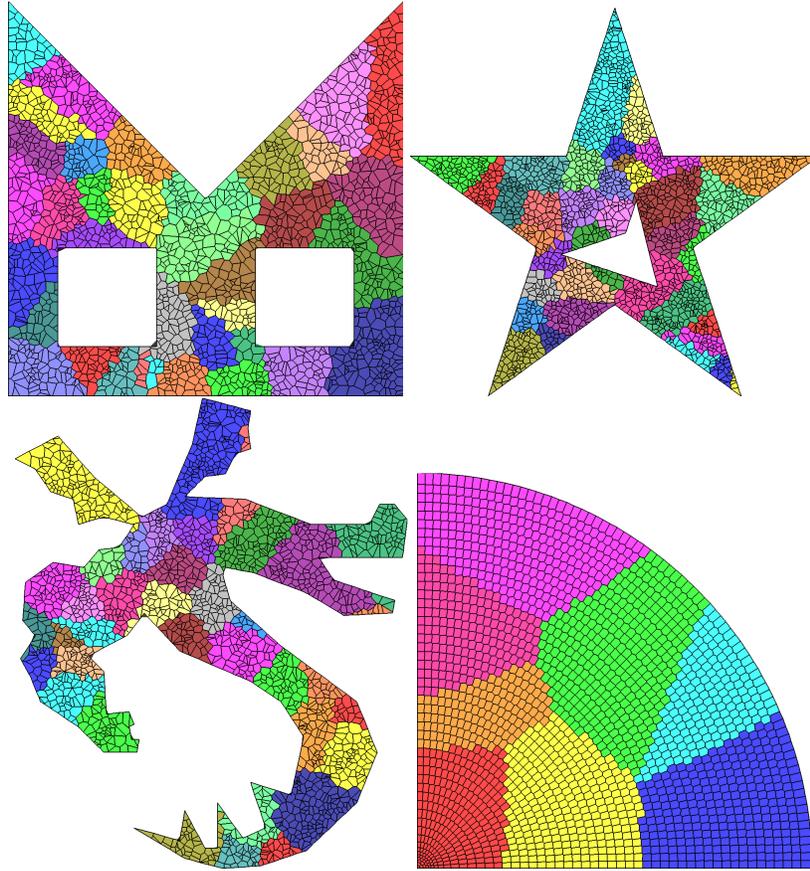
Figure 2 gives a collection of complicated 2D example boundaries. In most cases, 2000 random generator positions are distributed across 36 processors. Color changes indicate domain boundaries. The parallel algorithm respects non-convex regions. Mesh topology is consistent across domain boundaries.

### *Centroidal Relaxation of a Mesh*

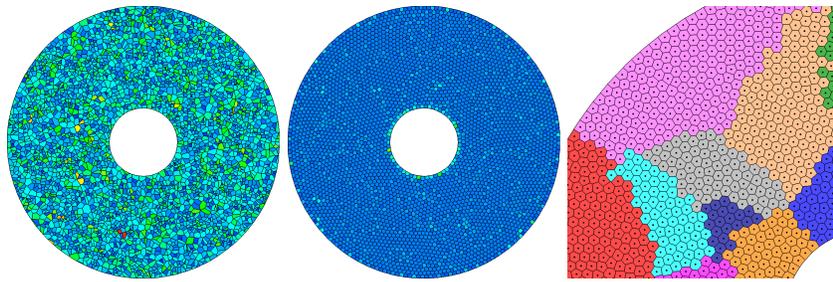
Polytope's parallel capabilities improve the efficiency of iterative mesh optimization schemes such as Lloyd's algorithm. Figure 3 demonstrates 1000 iterations of Lloyd's algorithm on 4000 generators, randomly distributed onto a toroidal boundary, and decomposed onto 20 processors.

### *Cosmological Dataset*

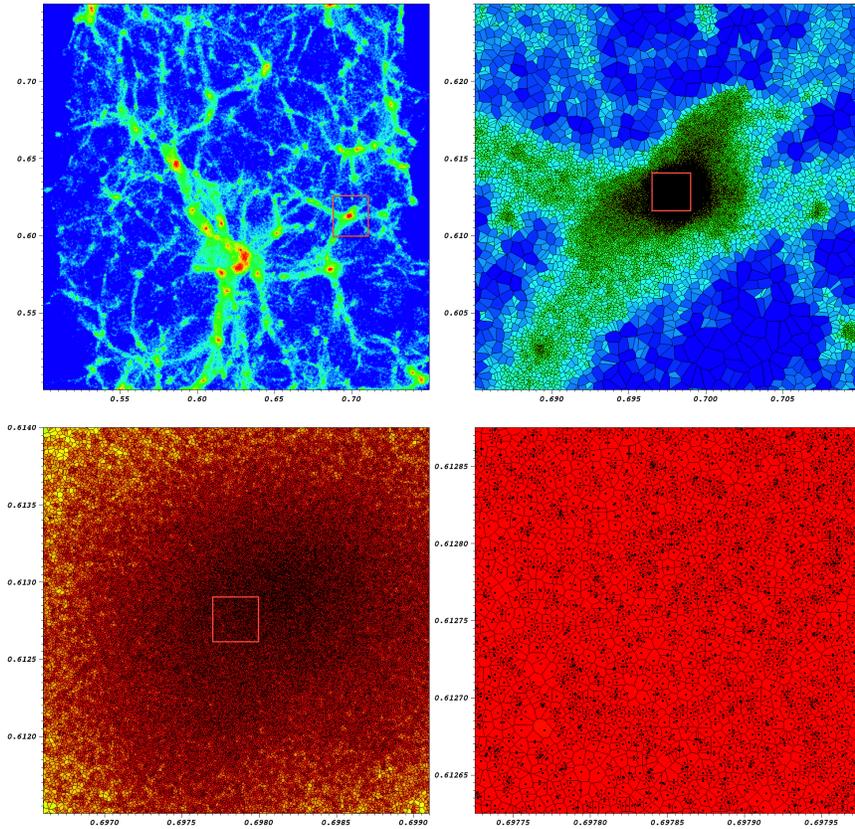
Automatic mesh generation is complicated by the existence of multiple spatial scales. Figure 4 demonstrates a mesh of more than 2 million generators. Point density varies over 5 orders of magnitude. Generators correspond to point masses taken from an astrophysical simulation of cosmological-scale structure growth. Polytope took 89 seconds to compute the mesh on 64 processors.



**Fig. 2.** Distributed meshes computed on a collection of complicated boundaries.



**Fig. 3.** Log condition number before and after 1000 iterations of Lloyd's algorithm. Zoom in of the relaxed mesh showing domain decomposition (right).



**Fig. 4.** Mesh cells colored by log of area. Zooming in by factors of 10 on a high-density region of the global mesh.

## References

1. Gehrels B, Lalande B, Loskot M, Wulkiewicz A Boost.Geometry Library, <http://www.boost.org>
2. Loubere R, Maire P-H, Shashkov M, Breil J, Galera S (2010) JCP 229
3. Owen J (2011) Multimat
4. Rycroft C (2009) Chaos 19:041111
5. Shewchuk J (1996) App Comput Geom 1148:203–222
6. Shewchuk J (1997) Discrete and Comput Geom 18:305–363
7. Si H (2011) Tetgen <http://tetgen.org>
8. Springel V (2009) Mon Not of the R Astron Soc
9. Starinshak D, Owen J, Johnson J (2013) Preprint
10. Sydorчук A (2012) Boost.Polygon Voronoi Library <http://www.boost.org/>
11. Yan D-M, Wang W, Levy B, Liu Y (2010) GMP Conference Proceedings
12. Johnson J, Owen J, Starinshak D (2013) Polytope <https://bitbucket.org/jjphatt/polytope>