

'votess: A multi-target, GPU-capable, parallel Voronoi tessellator'

Samridh Dev Singh¹, Chris Byrohl², and Dylan Nelson²

¹ Grinnell College, 1115 8th Avenue, 50112 Grinnell, United States of America ² Heidelberg University, Institute for Theoretical Astronomy, Albert-Ueberle-Str. 2, 69120 Heidelberg, Germany

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#)
- [Repository](#)
- [Archive](#)

Editor: [Open Journals](#)

Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#))

Statement of need

The Voronoi tessellation is a spatial decomposition that partitions space into a set of convex hulls based on proximity to a discrete set of seed points. It is an interesting problem due to its applications in biology, data science, geography, and physics. A few notable examples exist in computational cosmology—initially pioneered by van de Weygaert in the analysis of matter distribution (?), further developed through Optimal Transport theory for early-universe reconstruction ([Levy et al., 2021](#)), and also in observational data analysis and numerical simulations of cosmic structure formation ([Springel, 2010](#)).

The increasing size of datasets produced today have underscored the need for more efficient algorithms to both generate and analyse these datasets, and the rise of heterogenous computing facilities would enable such new algorithms to be run. There do exists several sequential and parallel implementations of the Voronoi Diagram problem [[Marot et al. \(2019\)](#)][[Wu et al., 2023](#)][[The CGAL Project \(2018\)](#)][[Inria, 2018](#)], however, they are mostly restricted to CPU or specific GPU architectures, thus limiting their potential as a portable multi-architecture algorithm.

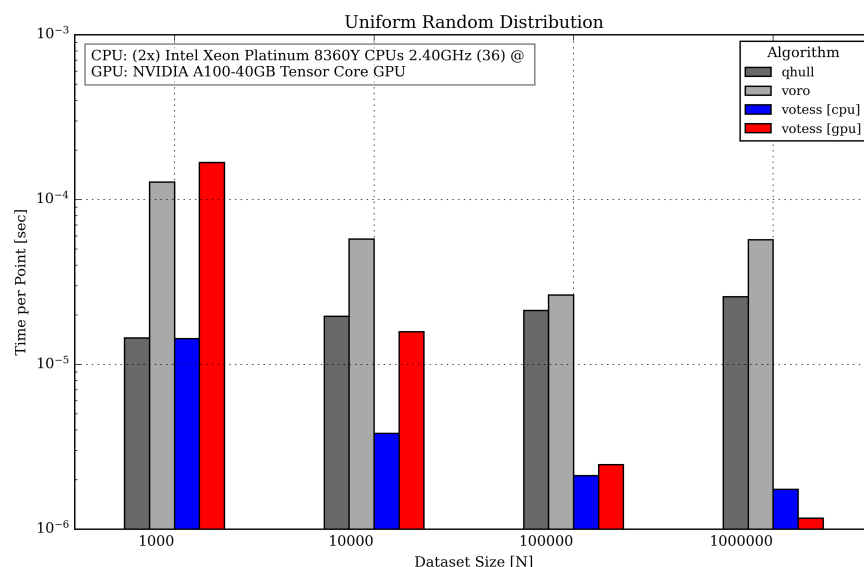
Summary

votess is a library for computing parallel 3D Voronoi tessellations on heterogeneous platforms, from CPUs to GPUs to future accelerator architectures. To do so, it uses the SYCL single-source framework abstraction in the C++ language. votess was designed to be portable yet performant, accessible to both developers and users with several easy-to-use interfaces.

The core method of votess consists of two main steps. First, the input set of points is sorted into a grid, and a k-nearest neighbors search is performed. Once the k nearest neighbors are identified for each point, the Voronoi cell is computed by iteratively clipping a bounding box using the perpendicular bisectors between the point and the identified neighbors. A *security radius* condition ([Lévy & Bonneel, 2013](#)) ensures that the resulting Voronoi cell is valid, and if the cell cannot be validated, an CPU fallback mechanism is used.

One advantage of this algorithm is the ability for each cell to be computed independently ([Ray et al., 2018](#)), making it suitable for parallel execution. It also produces the geometry of the Voronoi cells via their neighbor connectivity information, rather than a full combinatorial mesh data structure, thus making it more ammenable to data parallel architectures than alternatives such as sequential insertion or the Bowyer-Watson algorithm [[Bowyer \(1981\)](#)][[Watson, 1981](#)]. Additionally, a grid based datastructure for the all-k-nearest-neighbors sub problem, succeeded with the iterative clipping enables better caching behaviour on the GPU.

39 Performance



40

41 From the graph above, votess outperforms single-threaded alternatives.

42 In Figure 1, we show its performance compared to two other single-threaded Voronoi tessellation
43 libraries: Qhull and Voro++. Both are well-tested and widely used. Qhull is a computational
44 geometry library that constructs convex hulls and Voronoi diagrams using an indirect projection
45 method (Barber et al., 1996), while Voro++ is a C++ library specifically designed for three-
46 dimensional Voronoi tessellations, utilizing a cell-based computation approach that is well-suited
47 for physical applications (Rycroft, 2009).

48 We find that votess performs best on GPUs with large datasets. The CPU implementation
49 can outperform other implementations by a factor of 10 to 100.

50 Multithreaded Voronoi tessellation codes do exist, and these include ParVoro++ (Wu et al.,
51 2023), CGAL (The CGAL Project, 2018), and GEOGRAM (Inria, 2018). However, they do not
52 natively support GPU architectures.

53 Features

54 votess is designed to be versatile. It supports various outputs, including the natural neighbor
55 information for each Voronoi cell. This is a 2D jagged array of neighbor indices of the sorted
56 input dataset.

57 Users can invoke votess in three ways: through the C++ library, a command-line interface
58 clvotess, and a Python wrapper interface pyvotess. The C++ library offers a simple
59 interface with a tessellate function that computes the mesh. The Python wrapper, mirrors
60 the functionality of the C++ version, with native numpy array support, providing ease of use
61 for Python-based workflows.

62 The behavior of votess can be fine-tuned with run time parameters in order to (optionally)
63 optimize runtime performance.

Acknowledgements

CB and DN acknowledge funding from the Deutsche Forschungsgemeinschaft (DFG) through an Emmy Noether Research Group (grant number NE 2441/1-1).

References

- Barber, C. B., Dobkin, D. P., & Huhdanpaa, H. (1996). The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.*, 22(4), 469–483. <https://doi.org/10.1145/235815.235821>
- Bowyer, A. (1981). Computing dirichlet tessellations*. *The Computer Journal*, 24(2), 162–166. <https://doi.org/10.1093/comjnl/24.2.162>
- Inria, P. A.-G. (2018). *Geogram: A programming library of geometric algorithms*. <http://alice.loria.fr/software/geogram/doc/html/index.html>
- Levy, B., Mohayaee, R., & Hausegger, S. von. (2021). A fast semidiscrete optimal transport algorithm for a unique reconstruction of the early universe. *Monthly Notices of the Royal Astronomical Society*, 506(1), 1165–1185. <https://doi.org/10.1093/mnras/stab1676>
- Lévy, B., & Bonneel, N. (2013). Variational anisotropic surface meshing with voronoi parallel linear enumeration. In X. Jiao & J.-C. Weill (Eds.), *Proceedings of the 21st international meshing roundtable* (pp. 349–366). Springer Berlin Heidelberg. ISBN: 978-3-642-33573-0
- Marot, C., Pellerin, J., & Remacle, J.-F. (2019). One machine, one minute, three billion tetrahedra. *International Journal for Numerical Methods in Engineering*, 117(9), 967–990. <https://doi.org/https://doi.org/10.1002/nme.5987>
- Ray, N., Sokolov, D., Lefebvre, S., & Lévy, B. (2018). Meshless voronoi on the GPU. *ACM Transactions on Graphics*, 37(6), 1–12. <https://doi.org/10.1145/3272127.3275092>
- Rycroft, C. (2009). *VORO++: A three-dimensional voronoi cell library in c++*. <https://doi.org/10.2172/946741>
- Springel, V. (2010). Moving-mesh hydrodynamics with the AREPO code. *Proceedings of the International Astronomical Union*, 6(S270), 203–206. <https://doi.org/10.1017/S1743921311000378>
- The CGAL Project. (2018). *CGAL user and reference manual* (4.12.1 ed.). CGAL Editorial Board. <https://doc.cgal.org/4.12.1/Manual/packages.html#PkgSpatialSortingSummary>
- Watson, D. F. (1981). Computing the n-dimensional delaunay tessellation with application to voronoi polytopes*. *The Computer Journal*, 24(2), 167–172. <https://doi.org/10.1093/comjnl/24.2.167>
- Wu, G., Tian, H., Lu, G., & Wang, W. (2023). ParVoro++: A scalable parallel algorithm for constructing 3D voronoi tessellations based on kd-tree decomposition. *Parallel Computing*, 115, 102995. <https://doi.org/10.1016/j.parco.2023.102995>