




# ‘votess: A fast, multi-target voronoi tessellator using the SYCL framework’

Samridh Dev Singh<sup>1</sup>, Chris Byrohl<sup>2</sup>, and Dylan Nelson<sup>2</sup>

<sup>1</sup> Grinnell College, 1115 8th Avenue, 50112 Grinnell, United States of America <sup>2</sup> 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Published: unpublished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

With the rise of parallel architectures, it has now become possible to solve problems that would have been computationally too expensive in the past. One example would be a Voronoi tessellation on large datasets. Many projects utilize such an algorithm, be it from Cosmology, Earth Science, Material Science, Biochemistry, etc.

There isn’t however, a tessellation program that can run on the variety of platforms without severe modification to the source code. Thus forcing each problem requiring a voronoi tessellation to have a bespoke solution.

## Summary

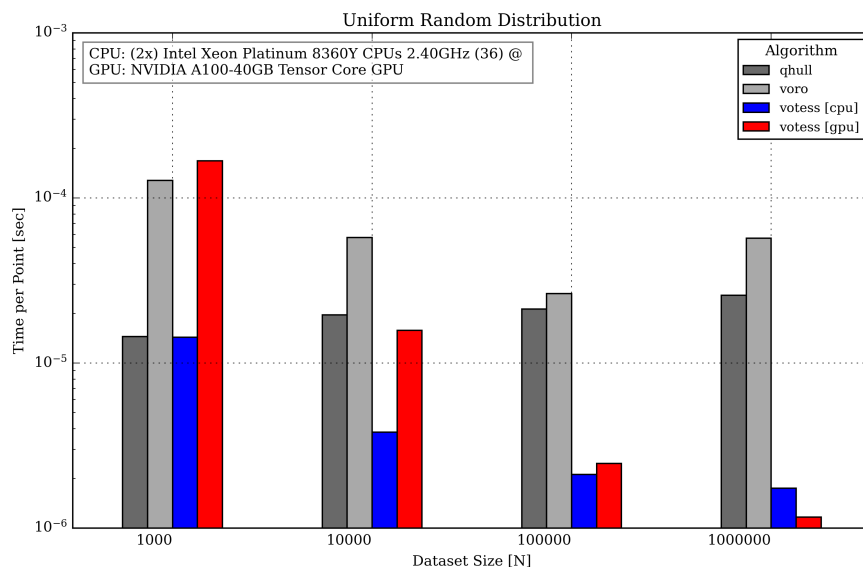
votess is a library for performing 3D Voronoi tessellations on heterogeneous platforms via SYCL framework. votess was designed to be portable, but performant, with an easy to use interface.

The underlying algorithm is based on a paper ([Ray et al., 2018](#)), which describes how to compute a Voronoi diagram without the need for a combinatorial mesh data structure, as required by classical approaches like the Bowyer-Watson algorithm. The core algorithm employed by votess consists of two main steps. First, given an input set of points, a k-nearest neighbors search is performed after sorting the points into a grid. With the nearest neighbors identified for each point, the Voronoi cell is computed by iteratively clipping a bounding box using the perpendicular bisectors of the point and its neighbors. To avoid iterating through all neighbors, a security radius condition is applied. If a Voronoi cell cannot be validated, a CPU fallback mechanism ensures robustness.

This simple, efficient algorithm allows for independent thread execution, making it well-suited for GPU parallelism.

## Performance

With a working implementation of votess, it can be seen that it outperforms several single-threaded applications:



The GPU implementation of votess is designed to be high throughput, and it is apparent as it is highest performing at datasets beyond a million points. The CPU implementation outperforms other applications of atleast tenfold, almost reaching a hundred fold at larger datasets. It must also be noted, that the benchmark was taken before votess had recieved optimizations.

## Features

votess provides a simple C++ interface to compute tesellations, that being a single function tesellate. There also exists an interface to select ipnuts One can select the target device to run the tesellation on, and currently CPU and GPU devices are supported.

A classvtargs is provided, with usage following closely to the `std::unordered_map` STL class. It enables users to tune parameters, which could help with the runtime performance of the application, if that is ever necessary.

The tesellate function returns a templated class dnn, as a 2 dimensional jagged array of neighbors that contribute to the voronoi cell of each particle in the sorted dataset. How the dataset is sorted can be tuned by class vtargs.

There also exists a python wrapper to tesellate, named pyvotess, with the same usage as the C++ implementation.

## Acknowledgements

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

## References

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>