The Journal of Open Source Software

startinpy: A Python library for modelling and processing 2.5D triangulated terrains

Hugo Ledoux [1]

1 Delft University of Technology, the Netherlands

DOI: 10.21105/joss.07123

Software

- Review
- Repository
- Archive

Editor: Michael Mahoney

Reviewers:

- @weiji14
- @kylemann16
- @gadomski

Submitted: 17 July 2024
Published: 12 November 2024

Summary

startinpy is a Python library for modelling and processing terrains using a two-dimensional (2D) Delaunay triangulation (DT). The triangulation is computed in 2D, but the z-elevation of the vertices is kept (which is referred to as 2.5D modelling).

Given a dataset formed of elevation samples (eg collected with lidar or photogrammetry), startinpy allows us to reconstruct a terrain, to remove points, to search efficiently in the triangulation, to attach attributes to the vertices, and to convert to a gridded terrain. Additional functionality is provided, including a few spatial interpolation methods based on the DT and/or its dual structure, the Voronoi diagram, have been implemented: linear interpolation, natural neighbours (Sibson, 1981), and Laplace interpolation (Belikov et al., 1997).

The core of the library is written in Rust (so it can process large datasets quickly), robust arithmetic is used for the updating of the DT (the robust predicates of Shewchuk (1996) are used), and it uses NumPy for input/output of data, which allows it to integrate with other Python libraries used by researchers.

The core of the library has been used to build a global coastal terrain using laser altimetric measurements from the space station (Pronk et al., 2024), and it has been used for several projects dealing with aerial and space lidar datasets, for instance Meschin (2023) and Kan (2024).

While startinpy was developed primarily to build and manipulate terrains, it can be used as an easy-to-use and fast 2D Delaunay triangulator (and Voronoi diagram generator), which, as elaborated in Aurenhammer et al. (2012), are two structures that play an essential role in several disciplines: astronomy, geology, ecology, engineering, etc.

The Journal of Open Source Software

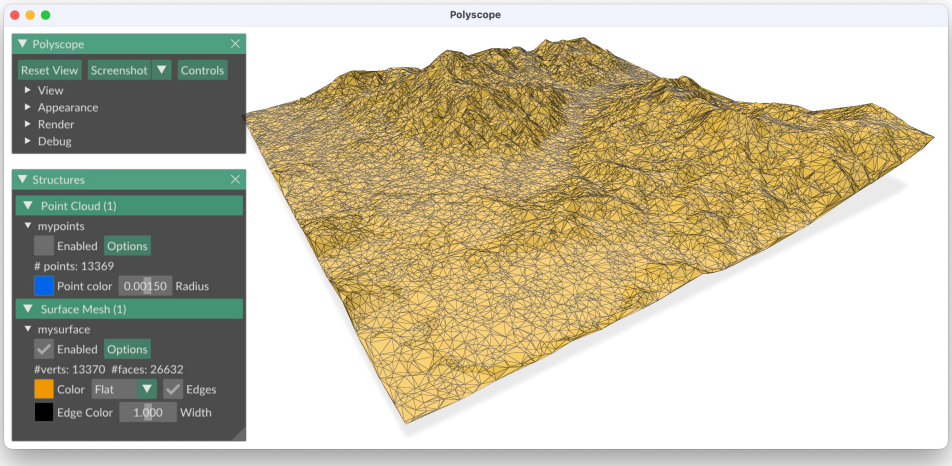

Figure 1: A lidar dataset terrain reconstructed with startinpy and visualised with another Python library (Polyscope).

Statement of need

While there exist many Python libraries for computing the DT in 2D (a search for "Delaunay triangulation" in the *Python Package Index* (PyPI) returns over 200 packages), most of them are not fully suitable for the modelling of 2.5D triangulated terrain.

startinpy has the following properties, which greatly improve the modelling and processing of 2.5D terrains.

It is fast for large datasets. With a modern lidar scanner, we can easily collect 50 samples/m^2, which means that a 1 km^2 area will contain 50+ million samples. Since constructing a DT requires several steps, if those steps are implemented in pure Python then the library becomes very slow. As can be seen in the DT construction comparison, startinpy is faster than most other libraries for large datasets. This is partly because it is 100% developed in Rust; the core library is called "startin" and its the source code is available at https://github.com/hugoledoux/startin.

Its data structure is exposed. Most libraries only return a list of vertices and triangles (triplets of vertex identifiers), which means that the user has to build an auxiliary graph to be able to find the adjacent triangles of a given one, or to find all the triangles that are incident to a given vertex (eg to calculate the normal). The name "startinpy" comes from the fact that the data structure implemented is based on the concept of *stars* in a graph (Blandford et al., 2005), which allows us to store adjacency and incidence, and have a very compact data structure. startinpy exposes methods to search triangles, find the adjacent triangles of a triangle, and find the incident triangles to a vertex.

The DT is incrementally constructed and deletion of vertices is possible. Unlike the majority of 2D DT implementations, startinpy implements an *incremental* insertion algorithm (Lawson, 1972), which allows for constructing a simplified TIN that best approximates the original terrain with only 10% of the points; see Garland & Heckbert (1995) for different strategies. startinpy also implements a modification of the deletion algorithm in Mostafavi et al. (2003), extended to allow the deletion of vertices on the boundary of the convex hull. The deletion of vertices in a DT is useful to remove outliers (which are detected by analysing neighbouring triangles in the DT) and for the implementation of the natural neighbours interpolation method (Sibson, 1981).

The z-values are stored and xy-duplicates handled. Some libraries allow us to attach extra

information to a vertex, but most often one has to build an auxiliary data structure in Python to manage those. Doing so is error-prone, tedious, and makes operations in 3D more complex (eg calculating the slope of an area, calculating the normal of a vertex, estimating the elevation with spatial interpolation, calculating volumes). By storing the z-values, startinpy allows us to merge vertices that are close to each other (in the xy-plane; the tolerance can be defined by the user) and if there are xy-duplicates, then a user-defined z-value can be kept (eg lowest or highest, depending on the application).

Extra attributes can be stored in the DT. It is possible to attach extra attributes with each vertex of the terrain. This can be used to preserve the lidar properties of the input points (eg intensity, RGB, number of returns, etc.).

The documentation of startinpy contains several examples of the library and how it can be used to prepare datasets for input to machine learning algorithms, to convert to different formats used in practice, to interpolate, etc.

Acknowledgements

I acknowledge the help of the students following the course *Digital terrain modelling (GEO1015)* at TUDelft over the last few years. Their feedback, questions, and frustrations on preliminary versions of startinpy helped me greatly.

References

Aurenhammer, F., Klein, R., & Lee, D.-T. (2012). *Voronoi diagrams and Delaunay triangulations*. World Scientific. https://doi.org/10.1142/8685

Belikov, V. V., Ivanov, V. D., Kontorovich, V. K., Korytnik, S. A., & Semenov, A. Y. (1997). The non-Sibsonian interpolation: A new method of interpolation of the values of a function on an arbitrary set of points. *Computational Mathematics and Mathematical Physics*, *37*, 9–15.

Blandford, D. K., Blelloch, G. E., Cardoze, D. E., & Kadow, C. (2005). Compact representations of simplicial meshes in two and three dimensions. *International Journal of Computational Geometry and Applications*, *15*(1), 3–24. https://doi.org/10.1142/S0218195905001580

Garland, M., & Heckbert, P. S. (1995). *Fast polygonal approximation of terrain and height fields* (CMU-CS-95-181). School of Computer Science, Carnegie Mellon University; School of Computer Science, Carnegie Mellon University.

Kan, L. W. L. (2024). *Spatial height prediction of ICESat-2 data using random forest regression* [Master's thesis]. MSc thesis in Geomatics, Delft University of Technology.

Lawson, C. L. (1972). Transforming triangulations. *Discrete Applied Mathematics*, *3*, 365–372. https://doi.org/10.1016/0012-365X(72)90093-3

Meschin, K. (2023). *Canopy gap fraction estimation from ICESat-2 ATL08 product* [Master's thesis]. MSc thesis in Geomatics, Delft University of Technology.

Mostafavi, M. A., Gold, C. M., & Dakowicz, M. (2003). Delete and insert operations in Voronoi/Delaunay methods and applications. *Computers & Geosciences*, *29*(4), 523–530. https://doi.org/10.1016/S0098-3004(03)00017-7

Pronk, M., Hooijer, A., Eilander, D., Haag, A., Jong, T. de, Vousdoukas, M., Vernimmen, R., Ledoux, H., & Eleveld, M. (2024). DeltaDTM: A global coastal digital terrain model. *Nature Scientific Data*, *11*(273). https://doi.org/10.1038/s41597-024-03091-9

Shewchuk, J. R. (1996). Robust adaptive floating-point geometric predicates. *Proceedings 12th Annual Symposium on Computational Geometry*, 141–150. https://doi.org/10.1145/

237218.237337

Sibson, R. (1981). A brief description of natural neighbour interpolation. In V. Barnett (Ed.), *Interpreting multivariate data* (pp. 21–36). Wiley.