The Journal of Open Source Software

stl2sdf: Parallel scalable signed-distance-field generator for arbitrary cartesian grids

Akshay Patil [1], **Stelios Vitalis** [1], **Ivan Padjen** [1], **Pedro Costa** [2], **and Clara Garcia-Sanchez** [1]

1 3D-Geoinformation Research Group, Faculty of Architecture and the Built Environment, Delft University of Technology, Delft, The Netherlands **2** Process & Energy Department, Faculty of Mechanical, Maritime and Materials Engineering, Delft University of Technology, Delft, The Netherlands

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

In this paper, we present a message-passing interface (MPI) based signed distance field (SDF) utility for fluid flow simulations over arbitrary Cartesian grids. Computing non-de-generate SDF for large grids (order 10 Million or greater grid count) can take anywhere from 10 minutes to hours in serial. To solve this issue, we developed a scalable Python-based code that can accurately and efficiently compute the SDF over cartesian grids as large as 2 Billion grid points.

Statement of need

Urban fluid flow simulations have become increasingly appealing and viable due to the accelerated growth and accessibility in Graphics Processing Units (GPU)-based computing platforms as demonstrated by the development of GPU-based Computational Fluid Dynamics (CFD) codes such as Muñoz-Esparza et al. (2020) and Costa (2018). These CFD codes use an Immersed Boundary Method (IBM) to introduce a solid interface within the Cartesian computational domain. One flavour of the IBM method that was originally introduced by Yang & Balaras (2006) can be easily implemented using SDF. For urban flow simulations and other high-Reynolds number simulations, the grid requirements can easily reach large numbers e.g., order Billion grid points, as a result, these simulations require SDF computations over such a Cartesian grid that prove to be unfeasible when done in serial.

`stl2sdf` was designed with the CFD users in mind that generally have access to high-performance computing (HPC) computing clusters and can leverage the use of MPI-based speed up. We have thoroughly tested `stl2sdf` on various different geometries, such as urban landscapes and coral beds, to demonstrate the versatile application of this utility. The code is based on `mpi4py`, `mesh_to_sdf`, and `trimesh` libraries that are freely available for Python users from standard repositories. Our code has been extensively tested on various geometries such as urban landscape as shown in Figure 1 and coral beds as shown in Figure 2.

Methodology

The central idea of the SDF computation in this code is to utilise the existing `mesh_to_sdf` Python library and port it to an MPI-based framework. This code uses the sampling method to compute the SDF where the numerical grid over which the SDF is computed is decomposed using a 1D slab decomposition (i.e., the x coordinate is split into chunks). As for the geometry (`stl/obj` file), each processor holds a copy, thus leading to a relatively larger memory requirement. Consequently, this ends up being the most memory-intensive part of the program. It is important to realise that decomposing the geometry file can lead to other bottlenecks

in the algorithm. Since most CFD users will have access to larger computing facilities, the memory requirements are not demanding. The input for the code can be provided through the supported file formats within the `trimesh` library. However, we recommend the `stl` and `obj` file formats. For applications where the SDF computation is required to be computed only close to the geometry, such as urban fluid dynamics simulations and heterogenous roughness of fixed height, the `clipSDF` user flag can expedite the SDF computation by not computing the SDF far away from the bottom wall thus further accelerating the usability of `stl2sdf`. Our MPI-version of the SDF generator would be the first of its kind utilising a well-known SDF computation library `mesh_to_sdf`, and porting it to a parallel version, thereby making it operational for more CFD users and codebases that rely on computing accurate SDF's.

Figures

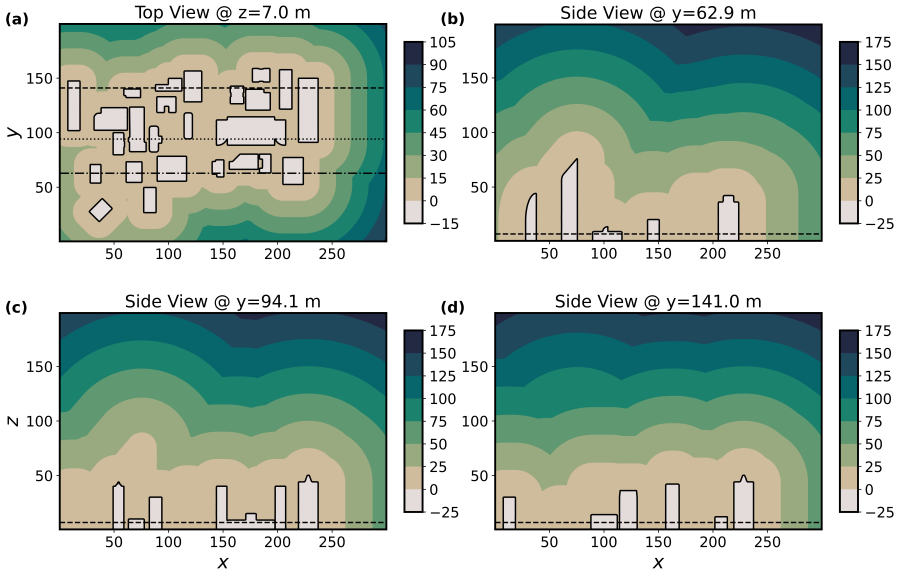

Figure 1: SDF field generated using `stl2sdf` for an urban landscape. Horizontal lines in panel (a) correspond to the locations of panels (b), (c), and (d).

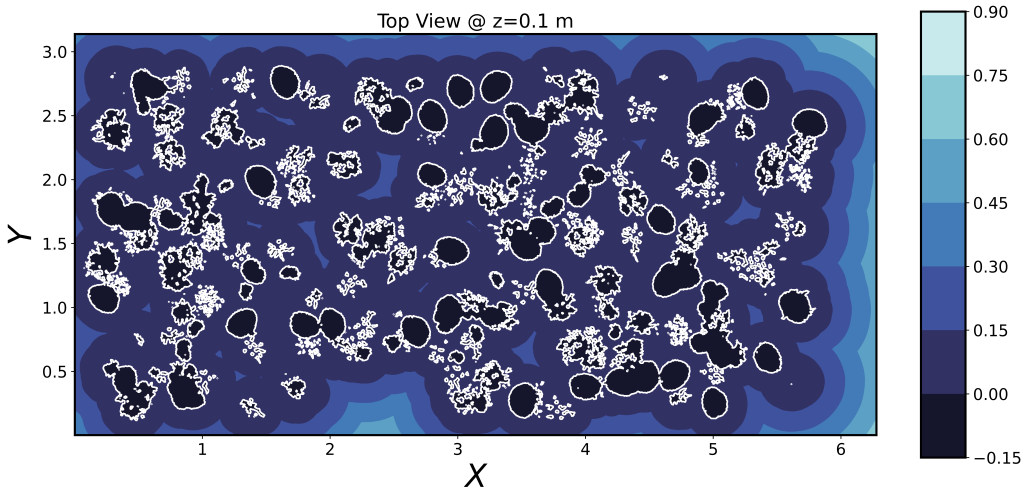

Figure 2: SDF field generated using `stl2sdf` for stochastically generated coral bed.

Acknowledgments

AP would like to thank the support and computing resources within the 3D-Geoinformation research group that enabled the development and testing of `stl2sdf`.

References

Costa, P. (2018). A FFT-based finite-difference solver for massively-parallel direct numerical simulations of turbulent flows. *Computers & Mathematics with Applications*, *76*(8), 1853–1862.

Muñoz-Esparza, D., Sauer, J. A., Shin, H. H., Sharman, R., Kosović, B., Meech, S., García-Sánchez, C., Steiner, M., Knievel, J., Pinto, J., & Swerdlin, S. (2020). Inclusion of building-resolving capabilities into the FastEddy® GPU-LES model using an immersed body force method. *Journal of Advances in Modeling Earth Systems*, *12*(11), e2020MS002141.

Yang, J., & Balaras, E. (2006). An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries. *Journal of Computational Physics*, *215*(1), 12–40.