The Journal of Open Source Software

VAST: the Void Analysis Software Toolkit

Kelly A. Douglass[1], **Dylan Veyrat**[1], **Stephen W. O'Neill, Jr.**[2], **Segev BenZvi**[1], **Fatima Zaidouni**[1], **and Michaela Guzzetti**[1, 3, 4]

1 University of Rochester **2** Independent Researcher **3** Smith College **4** University of Washington

DOI: 10.21105/joss.04033

Software

- Review
- Repository
- Archive

Editor: Dan Foreman-Mackey

Reviewers:

- @changhoonhahn
- @lavaux

Submitted: 17 December 2020
Published: 05 January 2022

Summary

Voids are expansive regions in the universe containing significantly fewer galaxies than surrounding galaxy clusters and filaments. They are a fundamental feature of the cosmic web and provide important information about galaxy physics and cosmology. For example, correlations between voids and luminous tracers of large-scale structure improve constraints on the expansion of the universe as compared to using tracers alone. However, what constitutes a void is vague and formulating a concrete definition to use in a void-finding algorithm is not trivial. As a result, several different algorithms exist to identify these cosmic underdensities. Our Void Analysis Software Toolkit, or VAST, provides Python 3 implementations of two such algorithms: **VoidFinder** and **V2**.

Statement of Need

Analyzing the next generation of cosmological surveys will require the ability to process large volumes of data (for example, at least 30 million galaxies and quasars from the Dark Energy Spectroscopic Instrument (Levi et al., 2013)). To more efficiently process such large datasets, this Python 3 implementation of **VoidFinder** includes a Cythonized (Behnel et al., 2011) version of the algorithm which also allows for multi-process void-finding. The void-finding algorithm in the `Vsquared` package uses the `scipy.spatial` (Virtanen et al., 2020) submodule for fast computation of the Voronoi tessellation and convex hulls involved in the algorithm.

VoidFinder

`vast.voidfinder` is a software package containing a Python 3 implementation of the Void Finder algorithm (El-Ad & Piran, 1997) that is based on the algorithm's Fortran implementation by Hoyle & Vogeley (2002). Motivated by the expectation that voids are spherical to first order, this algorithm defines voids as the unions of sets of spheres grown in the underdense regions of the large-scale structure. It begins by removing all isolated tracers from the catalog of objects, defined as having significantly (1.5σ) larger than average third-nearest neighbor distances. The remaining tracers are then placed on a grid, and spheres are grown from the centers of the empty cells until they are bounded by four tracers on their surfaces. All spheres larger than a specified radius (typically around 10 Mpc/h) are considered possible maximal spheres – the largest sphere that can fit in a given void region. Filtering through these candidate maximal spheres by order of decreasing radius, no maximal sphere can overlap by more than 10% of its volume with any other previously identified (larger) maximal sphere. After the maximal spheres are identified, the remaining holes are combined with these maximal spheres to enhance the void structure if they overlap exactly one maximal sphere by at least 50% of its volume. The union of a set of spheres (one maximal and the remaining smaller holes) defines a void region.

V2

`vast.Vsquared` is a software package for finding voids based on the ZOBOV (ZOnes Bordering On Voidness) algorithm (Neyrinck, 2008), which grows voids from local minima in the density field. This algorithm first produces a Voronoi tessellation of the catalog of large-scale tracers, and the volumes of the Voronoi cells are used to identify local density minima. Zones are then built from density minima in the distribution of cells using a watershed transform, where each cell is linked to its least dense neighbor. Finally, voids are formed from these by identifying low-density boundaries between adjacent zones and using them to grow unions of weakly divided zones. This list of voids is then typically pruned to remove void candidates unlikely to be true voids.
`Vsquared` includes several of the different void-pruning methods that exist, including methods from other ZOBOV implementations such as VIDE (Sutter et al., 2012) and REVOLVER (Nadathur et al., 2018).

VoidRender: a 3D Visualization of voids

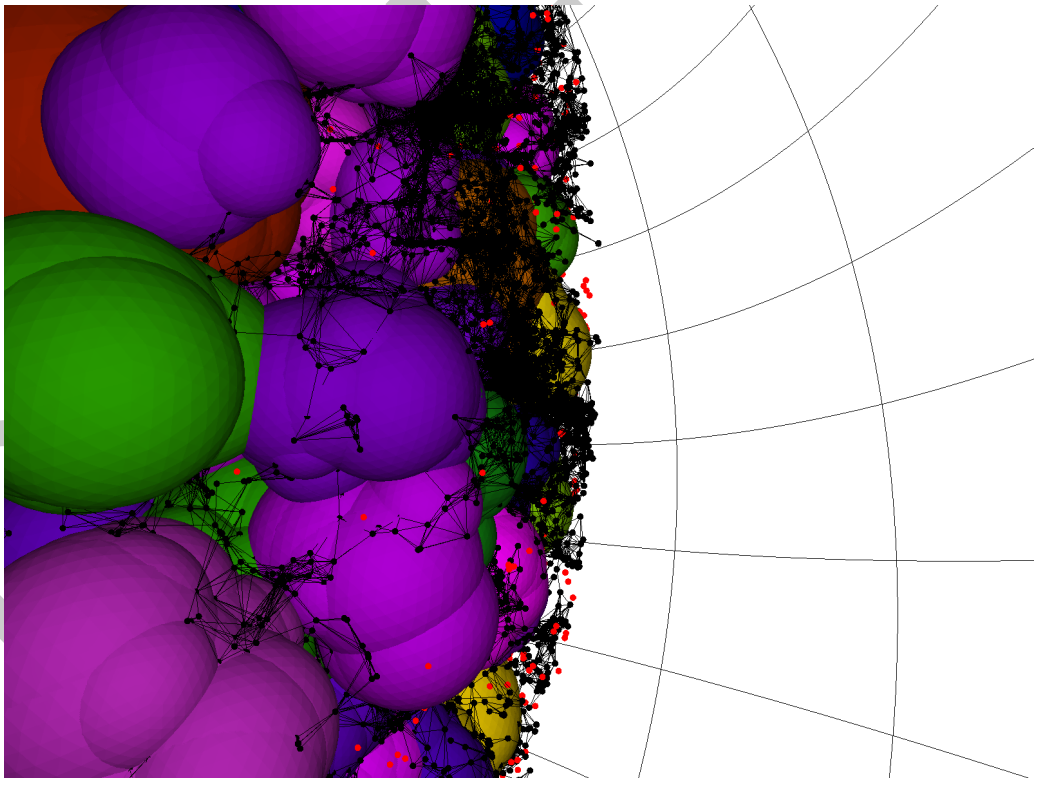

Figure 1: VoidRender visualization of the output from SDSS DR7 (Abazajian et al., 2009).

In order to aid in assessing the quality of the VoidFinder algorithm, the `vast.voidfinder.viz` package includes a `VoidRender` class (`from vast.voidfinder.viz import VoidRender`) which utilizes a combination of `OpenGL` and the python `vispy` package to enable real-time 3D rendering of the VoidFinder algorithm output. This 3D visualization allows the user to explore 3D space in a video-game-esque manner, where the w-a-s-d-style keyboard controls function to give the user a full range of motion: forward/backward/left/right/up/down translation and pitch/yaw/roll rotation.

Each void hole of the VoidFinder output is rendered to the screen using the icosadehral sphere approximation, where the depth of the approximation is configurable and higher approximation depths yield a finer and finer grained triangularization of each sphere. In addition, VoidRender includes an option to remove the interior walls of each void, which is approximated by removing the triangles from the triangluarization where all three vertices of a given triangle fall within the radius of an intersecting sphere. This option aids in visually inspecting the properties of joining spheres.

The galaxy survey upon which the output voids are based may also be included within the visualization, where each galaxy is represented by a small dot since the radius of even the largest galaxy is negligibly small compared to the radius of the smallest void. For visual purposes, the mouse scroll wheel may be used to enlarge or shrink the galaxy dot size. By passing the appropriate portions of the galaxy survey to different parts of the VoidRender keyword parameters, wall galaxies may be displayed in black and void galaxies may be displayed in red. Additionally, in order to help visualize the clustering of wall galaxies, another VoidRender option plots a thin black line between a galaxy and its K nearest neighbors, yielding a denser spider-web look for those galaxies which cluster together, as can be seen in Figure 1.

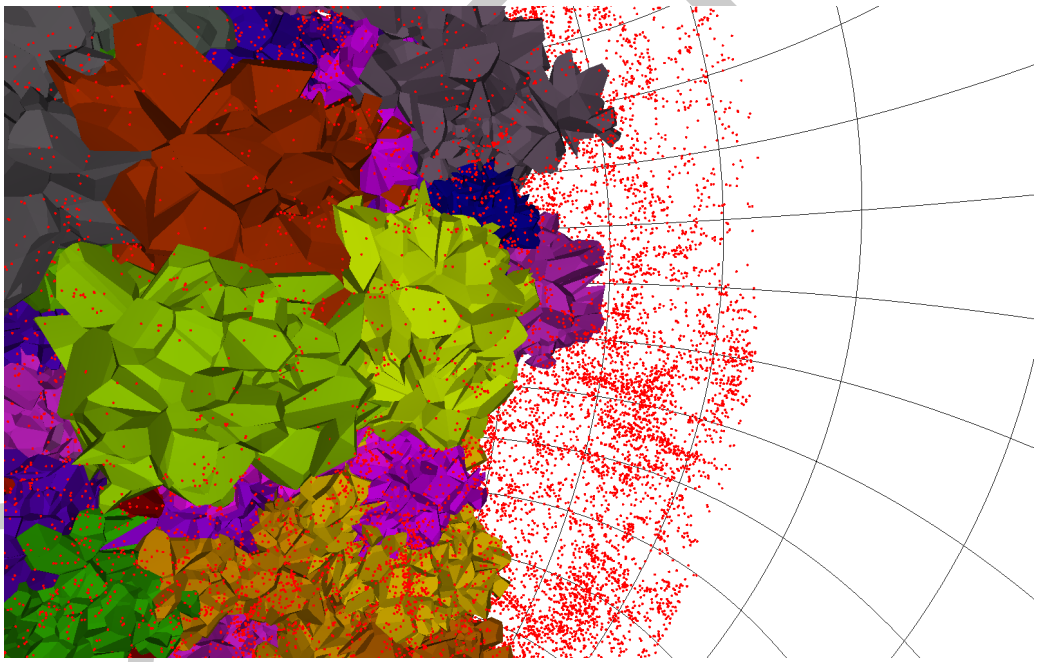

Figure 2: `Vsquared` visualization of the output from SDSS DR7.

An animated example of the VoidRender visualization can be found on YouTube. VoidRender can be utilized to produce screenshots or videos such as this example if a user's environment includes the `ffmpeg` library. `Vsquared` also includes an `OpenGL` and `vispy` based visualization for its output. The surfaces of voids found by the ZOBOV algorithm are made up of convex polygons, and are rendered exactly in 3D. Controls for movement and production of screenshots and videos are identical to those of VoidRender. An example of the `Vsquared` visualization is shown in Figure 2.

Acknowledgements

SB and DV acknowledge support from the DOE Office of High Energy Physics under award number DE-SC0008475. MG was supported by the NSF Research Experience for Undergrad-

uates program under award number 1757062. FZ acknowledges support from the University of Rochester Research Innovation Grants program.

References

Abazajian, K. N., Adelman-McCarthy, J. K., Agüeros, M. A., Allam, S. S., Prieto, C. A., An, D., Anderson, K. S. J., Anderson, S. F., Annis, J., Bahcall, N. A., & al., et. (2009). The seventh data release of the sloan digital sky survey. *The Astrophysical Journal Supplement Series*, *182*(2), 543–558. https://doi.org/10.1088/0067-0049/182/2/543

Behnel, S., Bradshaw, R., Citro, C., Dalcin, L., Seljebotn, D. S., & Smith, K. (2011). Cython: The best of both worlds. *Computing in Science Engineering*, *13*(2), 31–39.

El-Ad, H., & Piran, T. (1997). Voids in the Large-Scale Structure. *ApJ*, *491*(2), 421–435. https://doi.org/10.1086/304973

Hoyle, F., & Vogeley, M. S. (2002). Voids in the point source catalogue survey and the updated zwicky catalog. *ApJ*, *566*(2), 641.

Levi, M., Bebek, C., Beers, T., Blum, R., Cahn, R., Eisenstein, D., Flaugher, B., Honscheid, K., Kron, R., Lahav, O., McDonald, P., Roe, N., Schlegel, D., & DESI collaboration, representing the. (2013). *The DESI experiment, a whitepaper for snowmass 2013*.

Nadathur, S., Bautista, J., Carter, P., Hotchkiss, S., Percival, W., Radinovic, S., & Winther, H. (2018). *REVOLVER*. https://github.com/seshnadathur/Revolver/

Neyrinck, M. C. (2008). ZOBOV: a parameter-free void-finding algorithm. *Mon. Not. Roy. Astron. Soc.*, *386*, 2101. https://doi.org/10.1111/j.1365-2966.2008.13180.x

Sutter, P. M., Wandelt, B., Lavaux, G., Weinberg, D., Hamaus, N., & Pisani, A. (2012). *Public Cosmic Void Catalog*. http://www.cosmicvoids.net/

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Jarrod Millman, K., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … Contributors, S. 1. 0. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/https://doi.org/10.1038/s41592-019-0686-2