The Journal of Open Source Software

abspy: A Python package for 3D adaptive binary space partitioning and modeling

Zhaiyu Chen [1]

1 Technical University of Munich, Germany

DOI: 10.21105/joss.07946

Software

- Review
- Repository
- Archive

Editor: Matthew Feickert

Reviewers:

- @preetishkakkar
- @tuelwer

Submitted: 18 February 2025
Published: 06 June 2025

Summary

Efficient and robust space partitioning of 3D space underpins many applications in computer graphics. *abspy* is a Python package for adaptive binary space partitioning and 3D modeling. At its core, *abspy* constructs a plane arrangement by recursively subdividing 3D space with planar primitives to form a linear cell complex that reflects the underlying geometric structure. This adaptive scheme iteratively refines the spatial decomposition, producing a compact representation that supports efficient 3D modeling and analysis. Built on robust arithmetic kernels and interoperable data structures, *abspy* exposes a high-level Python API that lets researchers prototype advanced 3D pipelines such as surface reconstruction, volumetric analysis, and feature extraction for machine learning.

Statement of need

In many scientific and engineering workflows, raw 3D data must be transformed into compact volumetric representations for visualization, simulation, and analysis of complex 3D environments. Sensor-acquired point clouds and photogrammetric meshes often contain hundreds of millions of irregular, noisy samples that are expensive to store, transmit, and query, and that provide little inherent structure. Converting such data into a well-defined spatial decomposition, in which each cell represents a coherent geometric region, reduces memory usage, spares downstream algorithms from ad-hoc filtering and enables exact spatial predicates such as intersection, containment, and adjacency. The central challenge is to obtain this decomposition without inheriting the aliasing artefacts of coarse voxel grids or the combinatorial explosion produced by exhaustive plane arrangements (Edelsbrunner et al., 1986).

How can 3D space be partitioned into a minimal yet meaningful set of units that faithfully capture geometric detail? Adaptive binary space partitioning (BSP) (Murali & Funkhouser, 1997) addresses this challenge by dynamically subdividing 3D space according to the data's intrinsic features, constructing a plane arrangement that conforms to geometric boundaries. Despite its theoretical advantages, a high-level implementation suitable for modern research use cases has been missing. *abspy* fills this gap with a modern Python API for adaptive partitioning that delivers reliable and efficient 3D modeling. Researchers can thus tackle large-scale and complex reconstruction challenges with great reliability and precision. Unlike general-purpose libraries such as CGAL (Fabri & Pion, 2009) and Open3D (Zhou et al., 2018), which mainly supply low-level geometric utilities, *abspy* uniquely integrates adaptive BSP and exact arithmetic within a single framework, facilitating direct integration into research workflows tailored for compact, dynamic 3D modeling tasks and beyond.

Figure 1 illustrates how *abspy* uses user-supplied or extracted planar primitives to recursively subdivide space into a convex cell complex. Beginning with user-supplied planes, we insert each plane only into the BSP-tree leaves whose bounding boxes it intersects. This local, on-demand splitting keeps every cell convex, updates adjacency on the fly, and, by skipping unrelated

plane pairs, avoids the prohibitive cost of exhaustive arrangements. The result is a compact cell complex. Initially developed for research purposes, *abspy* has since evolved to support diverse 3D applications, particularly for compact surface reconstructions. It has been employed in several research projects and featured in scientific publications (Chen et al., 2022, 2024; Sulzer & Lafarge, 2024) as well as graduate students' projects.

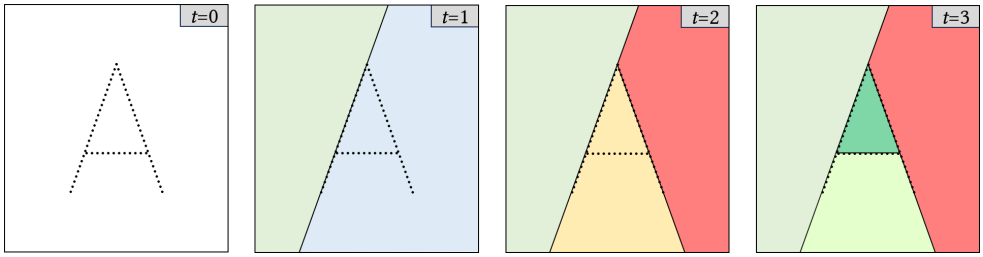

Figure 1: A 2D illustration for adaptive binary space partitioning. The ambient space is recursively partitioned into a cell complex with the insertion of planar primitives.

Overview of features

The core features of *abspy* include:

- **Planar primitive manipulation:** *abspy* extracts planar primitives from point clouds or meshes, and refines or perturbs the primitives to control noise, ensuring a robust representation of the input geometric structures.
- **Linear cell complex construction:** Using adaptive binary space partitioning, *abspy* recursively subdivides 3D space into a cell complex that reflects the data's spatial layout. This structured decomposition facilitates efficient spatial queries and further processing.
- **Dynamic graph generation:** The package iteratively constructs and updates a BSP-tree as a dynamic graph, supporting operations like connectivity analysis and neighborhood searches.
- **Robust spatial operations:** By leveraging the rational ring from SageMath (The SageMath Developers, 2025), *abspy* performs exact geometric computations that avoid inaccuracies associated with floating-point arithmetic, leading to reliable intersection tests and robust boundary determinations.
- **Surface reconstruction:** *abspy* identifies boundaries between interior and exterior cells to reconstruct polygonal surfaces using graph cuts, enabling the generation of surface models suitable for visualization and analysis.
- **Ease of integration:** With a Pythonic interface, comprehensive documentation, and practical examples, *abspy* integrates smoothly into existing research workflows. Its data structures are designed for interoperability with tools such as NetworkX (Hagberg et al., 2008) and Easy3D (Nan, 2021), facilitating further development.

Figure 2 showcases a few geometric representations that *abspy* can provide. From an input point cloud, for example, the library can manipulate points grouped by their supporting planar primitives, build a linear cell complex with its adaptive partitioning routine, extract a watertight polygonal surface from that complex through its graph-cut API, and sample representative points from the cells with the built-in samplers. These representations allow users to choose whichever level of abstraction best fits their application or to combine several of them as needed. The documentation of *abspy* contains tutorials for the package, its API reference, and file format specifications.

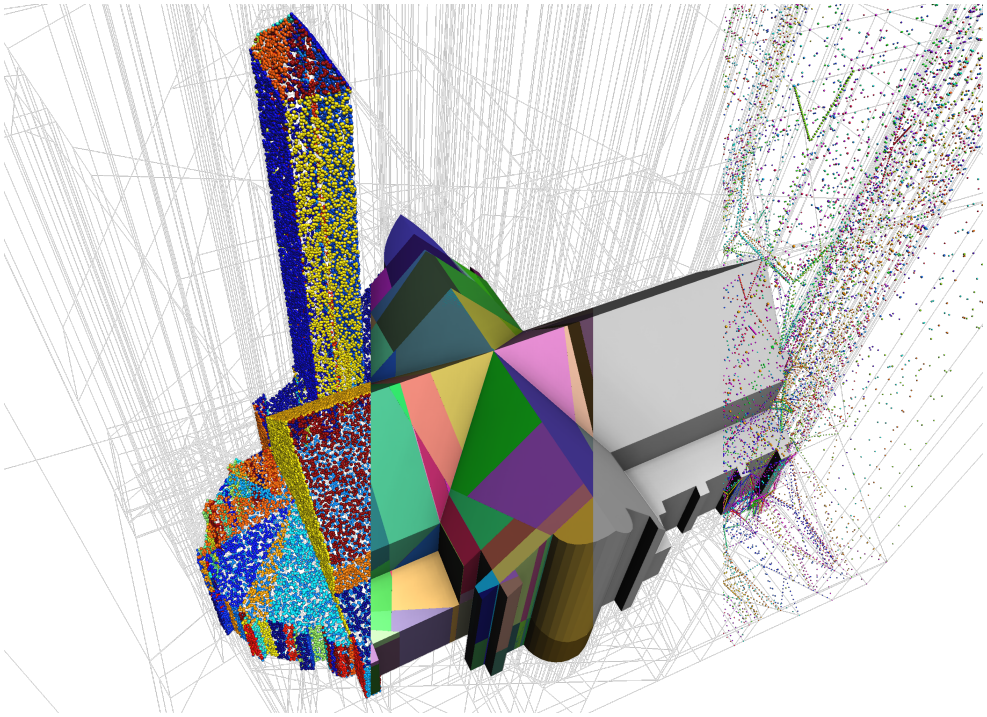

Figure 2: An overview of representations supported by *abspy*, shown left to right: points grouped and colored by planar primitives; cell complex of colored polyhedral volumes generated by adaptive space partitioning; reconstructed surface mesh; and sampled representative points from each cell volume, colored to match their cells.

Acknowledgements

We thank Liangliang Nan, Hugo Ledoux, Yuqing Wang, Qian Bai, and the JOSS reviewers for bug fixes, testing, and feedback. This work was supported in part by TUM Georg Nemetschek Institute under the AI4TWINNING project.

References

Chen, Z., Ledoux, H., Khademi, S., & Nan, L. (2022). Reconstructing compact building models from point clouds using deep implicit fields. *ISPRS Journal of Photogrammetry and Remote Sensing*, *194*, 58–73. https://doi.org/10.1016/j.isprsjprs.2022.09.017

Chen, Z., Shi, Y., Nan, L., Xiong, Z., & Zhu, X. X. (2024). PolyGNN: Polyhedron-based graph neural network for 3D building reconstruction from point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, *218*, 693–706. https://doi.org/10.1016/j.isprsjprs.2024.09.031

Edelsbrunner, H., O'Rourke, J., & Seidel, R. (1986). Constructing arrangements of lines and hyperplanes with applications. *SIAM Journal on Computing*, *15*(2), 341–363. https://doi.org/10.1109/sfcs.1983.11

Fabri, A., & Pion, S. (2009). CGAL: The computational geometry algorithms library. *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 538–539. https://doi.org/10.1145/1653771.1653865

Hagberg, A., Swart, P. J., & Schult, D. A. (2008). *Exploring network structure, dynamics, and function using NetworkX*. Los Alamos National Laboratory (LANL), Los Alamos, NM

(United States). https://doi.org/10.25080/tcwv9851

Murali, T., & Funkhouser, T. A. (1997). Consistent solid and boundary representations from arbitrary polygonal data. *Proceedings of the Symposium on Interactive 3D Graphics*, 155–162. https://doi.org/10.1145/253284.253326

Nan, L. (2021). Easy3D: A lightweight, easy-to-use, and efficient C++ library for processing and rendering 3D data. *Journal of Open Source Software*, *6*(64), 3255. https://doi.org/10.21105/joss.03255

Sulzer, R., & Lafarge, F. (2024). Concise plane arrangements for low-poly surface and volume modelling. *Computer Vision – ECCV 2024*, 357–373. https://doi.org/10.1007/978-3-031-72904-1_21

The SageMath Developers. (2025). *SageMath* (Version 10.6.beta3). Zenodo. https://doi.org/10.5281/zenodo.8042260

Zhou, Q.-Y., Park, J., & Koltun, V. (2018). Open3D: A modern library for 3D data processing. *arXiv Preprint arXiv:1801.09847*.