The Journal of Open Source Software

Boost.Geometry: A Generic C++ Geometry Library

Tinko Sebastian Bartels [3], **Vissarion Fisikopoulos** [2], **Barend Gehrels**[1], **Menelaos Karavelas**[1], **Bruno Lalande**[1], **Mateusz Łoskot**[1], **and Adam Wulkiewicz**[1]

1 Boost C++ Libraries **2** National Technical University of Athens, Greece **3** The Chinese University of Hong Kong, Shenzhen

DOI: 10.21105/joss.10328

Software

- Review
- Repository
- Archive

Editor: Hugo Ledoux

Reviewers:

- @kenohori
- @ioannisman

Submitted: 26 February 2026
Published: 19 May 2026

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

Computational geometry underpins numerous scientific and engineering domains, including GIS, robotics, computer graphics, CAD, and astronomy. Foundational algorithms such as convex hulls, intersection tests, and point-in-polygon queries are covered in standard texts (Berg et al., 2008); however, robust and efficient implementations are crucial for real-world applications where numerical robustness and performance matter.

`Boost.Geometry`[1] is an open-source C++ library providing generic, extensible, and standards-compliant geometric computation. It defines geometry *concepts* and implements algorithms in a dimension- and coordinate-system-agnostic manner. Since its inclusion in Boost C++ libraries (2011), a widely used peer-reviewed open-source collection of portable C++ components, it has been used in a range of C++ scientific and industrial software.

`Boost.Geometry` builds on several Boost libraries, among others: `Boost.Range` as the foundational abstraction for generic sequence iteration, `Boost.Core` for low-level utilities, `Boost.Config` for compiler and platform portability, `Boost.Math` as the numerical backbone for geometric and geodetic algorithms, and `Boost.MPL` for compile-time type manipulation in tag dispatching.

Statement of need

C++ developers working in GIS, robotics, computer graphics, CAD, simulation, and scientific computing frequently require geometry algorithms that are both correct and efficient across multiple coordinate systems. Existing solutions either target a single coordinate system, lack extensibility for user-defined types, or impose runtime overhead through dynamic dispatching. `Boost.Geometry` addresses these needs by providing:

- A generic programming architecture via concepts, type traits, and template metaprogramming, enabling zero-overhead abstraction
- Strategy-based dispatch enabling coordinate-system-specific algorithms
- Support for Cartesian, spherical, and geographic coordinate systems, including CRS transformations, projections, and spatial indexing
- OGC-compliant geometry types and spatial predicates (Herring, 2011)
- Header-only distribution, simplifying deployment and integration into existing codebases without additional build dependencies
- Seamless adaptation of user-defined geometry types without modifying existing data structures

[1]Authors are listed in alphabetical order by surname.

State of the field

A similar library is GEOS (GEOS contributors, 2025), a C++ port of the Java Topology Suite (JTS) that implements the OGC Simple Features standard for GIS applications and serves as the core geometry engine for major geospatial tools like PostGIS (PostGIS contributors, 2025). GEOS is widely used in GIS, but it is centered on its own runtime geometry model and API. By contrast, `Boost.Geometry` is designed as a generic C++ library: users can work with adapted user-defined types, write against stateless free functions, and integrate geometry algorithms into existing C++ code without introducing a separate object hierarchy.

CGAL (Fabri et al., 2000) is a comprehensive C++ library covering a broader range of geometric algorithms, including triangulations, Voronoi diagrams, mesh generation, and geometry processing, with a strong emphasis on robustness via exact geometric predicates and constructions. Compared with CGAL, `Boost.Geometry` is more narrowly focused on the geometry operations and predicates commonly needed in GIS, geospatial software, and related scientific applications. Its added value lies in combining this focus with support for Cartesian, spherical, and geographic coordinate systems, as well as spatial indexing, within a single generic programming framework.

Coordinate transformations and CRS handling are commonly performed using PROJ (PROJ contributors, 2024), which users often combine with geometry libraries for projections and datum transformations. Similarly, specialized libraries such as libspatialindex (Libspatialindex contributors, 2024) are frequently used for spatial indexing. `Boost.Geometry` does not aim to replace such specialized tools in full; rather, it provides commonly needed projections, coordinate-system-aware algorithms, and an R-tree implementation for spatial and nearest-neighbor queries within the same header-only library, which can simplify integration in some workflows.

GeographicLib (Karney & GeographicLib contributors, 2013) and Karney's work on geodesics (Karney, 2013) provide state-of-the-art algorithms for ellipsoidal geodesic problems. `Boost.Geometry` builds on these geodetic methods and incorporates them into a broader geometry library, allowing users to apply geographic calculations as part of higher-level operations such as distance, area, and buffering. This integration is useful in applications that need geodetic accuracy without first projecting data into a planar coordinate system.

In summary, these libraries each excel in their respective domains, but `Boost.Geometry` occupies a distinct position by combining generic C++ type adaptation, coordinate-system-aware geometric algorithms, geodetic support, and integrated spatial indexing in a single header-only library.

Software design

`Boost.Geometry` provides:

- **Geometry models:** points, boxes, rings, polygons, multi-geometries
- **Algorithms:** area, distance, length, centroid, convex hull, within, intersects, union, intersection, simplify, buffer, transformations and projections
- **Spatial indexing:** R-tree implementation supporting spatial and nearest-neighbor queries

`Boost.Geometry` is a generic, header-only C++14 library built around stateless free functions with template-based dispatch via concepts, traits, and geometry tags, so users can adapt their own geometry types to the `Boost.Geometry` interfaces (e.g., via registration macros) without changing their data structures and with no runtime overhead.

Algorithms are structured as coordinate-system-agnostic core logic combined with coordinate-system-specific strategies selected from the input geometry's declared coordinate-system tag (e.g., Cartesian vs. spherical vs. geographic). In the spherical model, computations follow

great-circle arcs; in the geographic model, an ellipsoidal Earth model is used for more accurate geodetic results. Notably, algorithms such as `buffer` operate directly in the geographic or spherical domain without requiring a prior projection to a planar coordinate system, preserving geodetic accuracy throughout.

`Boost.Geometry` provides a range of geographic strategies that trade accuracy for speed: fast, efficient closed-form formulas for common use cases, and more accurate series-expansion methods based on Karney's geodesic algorithms (Fisikopoulos, 2019; Karney, 2013).

Numerical robustness can cause silent failures in standard GIS algorithms (Bartels & Fisikopoulos, 2021; Kettner et al., 2004). `Boost.Geometry` mitigates these problems using filtered predicates (Devillers & Pion, 2003) that perform fast approximate tests and fall back to exact evaluations when needed, ensuring correct topology and predicate results.

The source code is distributed under the Boost Software License. Detailed documentation and reference material are available on the Boost website (Gehrels et al., 2024). Contributor guidelines, documentation tooling, talks, and videos are collected on the project wiki. The repository's `test/` directory contains an extensive suite of unit and regression tests. The repository also provides examples in multiple forms: standalone tutorial-style programs in `example/`, documentation-integrated snippets in `doc/src/examples/`, and spatial-index-specific material in `index/example/`. Continuous integration on the Boost project provides broad platform and compiler coverage; `Boost.Geometry` also runs project-specific CI on GitHub Actions and CircleCI.

Research impact statement

`Boost.Geometry` has demonstrated significant adoption across scientific and industrial domains since its inclusion in Boost (2011). The examples below are indicative rather than exhaustive, and highlight the diversity of settings in which the library has been used.

MySQL uses `Boost.Geometry` as the geometry engine for spatial SQL operations (Zhao, 2014), demonstrating use in production database systems.

In spatial data management research, Hecatoncheir (Georgiadis et al., 2025), a distributed in-memory spatial data management library, uses `Boost.Geometry` for geometry comparisons. The system reports orders-of-magnitude speedups over Apache Sedona, illustrating the use of `Boost.Geometry` within a high-performance distributed setting.

In scientific computing, the lifex finite-element library (Bucelli, 2025) uses `Boost.Geometry`'s R-tree implementation for efficient nearest-neighbor searches in computational physics simulations, demonstrating applicability in numerical scientific computing beyond GIS.

The library has also been used in terrain analysis in real-time strategy video games (Richoux, 2022) for simplifying obstacle contours and pruning Voronoi diagrams, utilizing the simplify and R-tree implementations respectively.

In crowd simulation research, Vermeulen et al. (2017) used and evaluated `Boost.Geometry`'s R-tree for k-nearest-neighbour searching.

In robotics, Ashtekar & Dutta (2023) used `Boost.Geometry` for geometrical computations in a push-recovery controller for a bipedal robot balancing on offset planes.

In naval and polar engineering, Metrikin (2014) integrated `Boost.Geometry` into a simulation framework for stationkeeping of a vessel in floating sea ice, leveraging its polygon geometry operations such as intersection and difference. A related study by Yang et al. (2021) used `Boost.Geometry` in a numerical model to investigate the effect of ice floe shape on ship resistance under low-concentration broken ice condition.

Acknowledgements

We acknowledge contributions from the many developers who have submitted new features, bug reports, fixes, and feature requests to `Boost.Geometry` over the years, and support from the Boost C++ Libraries community.

AI usage disclosure

No generative AI tools were used in the development of this software or supporting materials. Generative AI was used in a limited role to suggest wording improvements during proofreading of this manuscript. All AI-suggested edits were reviewed and confirmed by the authors. The AI tool used for this assistance was GitHub Copilot with GPT-5.4.

References

Ashtekar, V., & Dutta, A. (2023). Push recovery control of a bipedal robot standing on two offset planes in double leg stance. *Proceedings of the 2023 6th International Conference on Advances in Robotics (AIR '23)*, 1–8. https://doi.org/10.1145/3610419.3610469

Bartels, T., & Fisikopoulos, V. (2021). Fast robust arithmetics for geometric algorithms and applications to GIS. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLVI-4/W2-2021*, 1–8. https://doi.org/10.5194/isprs-archives-XLVI-4-W2-2021-1-2021

Berg, M. de, Cheong, O., Kreveld, M. van, & Overmars, M. (2008). *Computational geometry: Algorithms and applications* (3rd ed.). Springer.

Bucelli, M. (2025). The lifex library version 2.0. *ACM Trans. Math. Softw.*, *51*(4). https://doi.org/10.1145/3748817

Devillers, O., & Pion, S. (2003). Efficient exact geometric predicates for Delaunay triangulations. *Proceedings of the 5th Workshop on Algorithm Engineering and Experiments*, 37–44. https://inria.hal.science/inria-00344517

Fabri, A., Giezeman, G.-J., Kettner, L., Schirra, S., & Sch"onherr, S. (2000). On the design of CGAL, a computational geometry algorithms library. *Software: Practice and Experience*, *30*(11), 1167–1202. https://doi.org/10.1002/1097-024X(200009)30:11%3C1167::AID-SPE337%3E3.0.CO;2-B

Fisikopoulos, V. (2019). Geodesic algorithms: An experimental study. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLII-4/W14*, 45–47. https://doi.org/10.5194/isprs-archives-XLII-4-W14-45-2019

Gehrels, B., Lalande, B., Łoskot, M., Wulkiewicz, A., & others. (2024). *Boost.Geometry documentation*. https://www.boost.org/libs/geometry/

Georgiadis, T., Michalopoulos, A., Dimitropoulos, D., Tsitsigkos, D., & Mamoulis, N. (2025). Hecatoncheir: Scaling up and out spatial data management. *Proceedings of the 33rd ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL '25)*, 844–847. https://doi.org/10.1145/3748636.3762796

GEOS contributors. (2025). *GEOS computational geometry library*. Open Source Geospatial Foundation. https://libgeos.org/

Herring, J. (2011). *OpenGIS implementation standard for geographic information – Simple Feature Access – Part 1: Common architecture* (OGC 06-103r4). Open Geospatial Consortium. https://www.ogc.org/standards/sfa

Karney, C. F. F. (2013). Algorithms for geodesics. *Journal of Geodesy*, *87*(1), 43–55.

https://doi.org/10.1007/s00190-012-0578-z

Karney, C. F. F., & GeographicLib contributors. (2013). *GeographicLib*. https://geographiclib.sourceforge.io/

Kettner, L., Mehlhorn, K., Pion, S., Schirra, S., & Yap, C. (2004). Classroom examples of robustness problems in geometric computations. *Algorithms–ESA 2004: 12th Annual European Symposium, Bergen, Norway, September 14-17, 2004. Proceedings 12*, 702–713. https://doi.org/10.1016/J.COMGEO.2007.06.003

Libspatialindex contributors. (2024). *Libspatialindex - R-tree spatial index library*. https://libspatialindex.org/

Metrikin, I. (2014). A software framework for simulating stationkeeping of a vessel in discontinuous ice. *Modeling, Identification and Control*, *35*(4), 211–248. https://doi.org/10.4173/MIC.2014.4.2

PostGIS contributors. (2025). *PostGIS – Spatial and geographic objects for PostgreSQL*. https://postgis.net/

PROJ contributors. (2024). *PROJ - cartographic projections and coordinate transformations library*. https://proj.org/

Richoux, F. (2022). Terrain analysis in StarCraft 1 and 2 as combinatorial optimization. *2022 IEEE Congress on Evolutionary Computation (CEC)*. https://doi.org/10.1109/CEC55065.2022.9870230

Vermeulen, J. L., Hillebrand, A., & Geraerts, R. (2017). A comparative study of k-nearest neighbour techniques in crowd simulation. *Computer Animation and Virtual Worlds*, *28*(3–4), e1775. https://doi.org/10.1002/cav.1775

Yang, B., Sun, Z., Zhang, G., Wang, Q., Zong, Z., & Li, Z. (2021). Numerical estimation of ship resistance in broken ice and investigation on the effect of floe geometry. *Marine Structures*, *75*, 102867. https://doi.org/10.1016/j.marstruc.2020.102867

Zhao, J. (2014). *MySQL 5.7 GIS enhancements*. https://dev.mysql.com/blog-archive/making-use-of-boost-geometry-in-mysql-gis/