The Journal of Open Source Software

PointCloudCrafter: Tools for Handling, Manipulating and Analyzing of Point Clouds

Dominik Kulmer [1*] **and Maximilian Leitenstern** [1*]

1 Institute of Automotive Technology, Department of Mobility System Engineering, School of Engineering and Design, Technical University of Munich, Germany. * These authors contributed equally.

DOI: 10.21105/joss.10459

Software

- Review
- Repository
- Archive

Editor: Johan Larsson

Reviewers:

- @jeofo
- @nitishsanghi
- @KennethKwabenaKenney

Submitted: 30 January 2026
Published: 26 June 2026

Summary

PointCloudCrafter is a C++ command-line interface (CLI) toolkit for the processing, manipulation, and analysis of three-dimensional point cloud data. The software provides a collection of compiled executables for integration into automated research pipelines, enabling common point cloud operations without requiring custom code. The toolkit supports the conversion and processing of data from multiple acquisition and storage formats, including ROS 2 bag recordings, binary files, plain-text files, and standard OBJ, PLY and PCD files. A central design feature of PointCloudCrafter is its schema-agnostic handling of per-point attributes. Arbitrary scalar fields associated with individual points are preserved throughout conversion, transformation, filtering, and aggregation steps, ensuring that auxiliary metadata remains available for downstream analysis. Furthermore, it allows the accumulation of multiple point clouds based on their timestamps and transforms. By combining support for robotic middleware data with batch-oriented file processing, PointCloudCrafter facilitates reproducible workflows across a range of scientific and engineering applications that rely on large-scale three-dimensional point cloud data.

State of the field

Three-dimensional point cloud data is widely used in robotics, autonomous driving, remote sensing, and scientific data analysis. In practice, such data is commonly encountered in two distinct representations. During acquisition and experimentation, point clouds are often recorded as time-stamped data streams using robotic middleware such as ROS 2. For benchmarking, archival, and offline analysis, the same data is typically stored and distributed as static files in dataset-specific or standardized formats. In particular, the use of automated pipelines for neural network training often requires the input data to be static files with a specific naming scheme.

The Point Cloud Library (PCL) (Rusu & Cousins, 2011) provides a comprehensive set of data structures and algorithms for three-dimensional processing and serves as the technical foundation of PointCloudCrafter, but its use generally requires direct integration into custom software projects. Graphical tools such as CloudCompare (Girardeau-Montaut, 2026) are well-suited for interactive inspection but are not designed for automated, headless processing of large datasets. Python-based libraries such as Open3D (Zhou et al., 2018) and Pyoints (Lamprecht, 2019) offer accessible interfaces for point cloud manipulation, but performance and input-output overhead can become limiting factors when processing large directories of high-resolution scans. PDAL (PDAL-Contributors, 2024) is a mature framework for large-scale geospatial point cloud processing and provides a flexible pipeline abstraction for working with formats such as LAS and LAZ. While PDAL is well-suited for geospatial analysis and

national LiDAR datasets, it does not natively integrate with robotic middleware or support time-resolved transformations based on sensor pose information. Recent toolkits such as PointClouds.jl (Schmid et al., 2025) provide efficient and flexible programmatic interfaces for point cloud processing within the Julia ecosystem, with a focus on interactive development and integration with geospatial data sources. While well-suited for algorithm development and exploratory analysis, such approaches require users to implement custom processing logic and are not designed as standalone command-line tools for automated conversion of robotic middleware data. Libraries such as libpointmatcher (Pomerleau et al., 2013) focus on modular implementations of point cloud registration algorithms and are typically embedded within state estimation or SLAM systems rather than used for dataset-level preprocessing and format conversion.

Statement of need

Transitioning between streaming and static point cloud representations, or performing large-scale batch operations on point cloud collections, remains a recurring challenge. Existing tools either require custom code integration, lack support for automated headless processing, or do not natively integrate with robotic middleware.

A further limitation of many existing tools is the implicit assumption of fixed-point schemas. Conversion utilities and parsers often restrict data to spatial coordinates and optional color information, discarding non-standard scalar fields. However, many scientific applications depend on additional per-point attributes. Automotive LiDAR sensors provide channels such as intensity, timestamp, and ring index, which are required for calibration, motion correction, and mapping. Furthermore, novel imaging RaDAR sensors export point cloud data with metadata, such as Doppler velocity, RaDAR cross-section, or signal-to-noise ratio. In other domains, including structural biology, point clouds may encode physicochemical properties such as hydrophobicity or electrostatic potential (Bazzano et al., 2025; Parigger et al., 2024). Preserving these attributes across processing steps is essential for reproducibility and downstream analysis.

PointCloudCrafter addresses these needs by providing a lightweight CLI toolkit focused on batch-oriented point cloud conversion and processing. The primary software contribution is a schema-agnostic, command-line-driven framework that unifies middleware-based and file-based point cloud processing while preserving arbitrary per-point attributes. Arbitrary scalar fields refer to dynamically discovered per-point attributes whose names, data types, and memory layout are preserved without enforcing a predefined schema.

Software design

PointCloudCrafter is implemented in C++ and is designed primarily as a command-line toolkit composed of standalone executables for point cloud processing. The software builds on the Point Cloud Library (PCL) (Rusu & Cousins, 2011), which provides the underlying data structures and algorithms for ROS 2 point cloud streaming and for commonly used file formats such as PCD. PointCloudCrafter does not reimplement PCL functionality; instead, it exposes selected operations and extends them through a unified, reproducible command-line interface.

The software is developed using modern C++ design practices, including a clear separation between executable logic and shared processing components. This design improves maintainability and allows new command-line tools to be added in the future without altering existing workflows. Documentation of the codebase and available commands is generated with Doxygen and published as a static website on GitHub.

The toolkit consists of two main executables that target distinct workflows. These executables share a common internal library that implements functionality for loading, saving, analyzing, transforming, filtering, and merging point clouds. Its purpose is to ensure consistent behavior

and reduce code duplication across the executables, while keeping the user-facing interface focused on command-line usage. Internally, all point cloud data is represented using the `PCLPointCloud2` data structure. This representation enables schema-agnostic handling of point clouds by allowing arbitrary per-point attributes to be discovered and preserved at runtime without enforcing a predefined schema.

PointCloudCrafter provides two primary execution modes, *rosbag* and *file*, for streaming and static data workflows, respectively. The *rosbag* executable enables extraction of `sensor_msgs::msg::PointCloud2` topics from ROS 2 bag recordings. This mode integrates with the `TF2` library (Foote, 2013) to perform time-resolved pose lookups based on the recorded transform tree. Each point cloud can be transformed from the sensor coordinate frame into a user-specified fixed reference frame using the pose corresponding to the exact acquisition timestamp. This enables motion-consistent export of point cloud data from dynamic sensor platforms and a timestamp-based fusion of different point cloud sensor topics. The *file* executable focuses on batch-oriented processing of static point cloud files. Supported formats include common text and binary formats such as PLY, PCD, OBJ, and TXT, as well as dataset-specific binary formats used in KITTI (Geiger et al., 2012) and nuScenes (Caesar et al., 2020). Both executables support a common set of processing operations implemented in the shared internal codebase. These operations allow the user to apply rigid-body transformations, merge multiple point clouds into a single representation, analyze timestamps, and perform filtering operations. Filters include geometric cropping using box, spherical, or cylindrical regions, voxel grid downsampling, and statistical or radius-based outlier removal. All processing steps are fully parameterized via command-line options, enabling deterministic and reproducible execution across datasets. Due to the schema-agnostic internal representation, per-point attributes, such as sensor-specific metadata or domain-specific properties, are preserved throughout processing whenever supported by the input and output formats.

A comprehensive overview of the available commands, parameters, and usage examples is provided in the online documentation at https://tumftm.github.io/PointCloudCrafter/usage.html.

Research impact statement

PointCloudCrafter was developed to support the reproducible preprocessing of large-scale point cloud datasets in research environments. The software evolved organically within the Institute of Automotive Technology at the Technical University of Munich to replace highly fragmented processing scripts. Driven by the practical requirements of over a dozen student theses and ongoing research projects, the toolkit's feature set was continuously refined through direct user feedback. This iterative, collaborative development established it as a standard framework for point cloud tasks, from raw data extraction to targeted data alteration.

PointCloudCrafter has been applied across diverse robotics domains to solve specific engineering problems. For example, researchers used it to create an automated pipeline that converts ROS 2 bag recordings into nuScenes- or KITTI-style datasets. These datasets are used to train and evaluate machine learning perception algorithms (Tempfli et al., 2025) and to create a public dataset for a wide range of applications (Kulmer, Tahiraj, et al., 2025). In localization research, the tool was used to extract and compile point cloud datasets for LiDAR odometry benchmarking (Sauerbeck et al., 2023). In mapping, the tool was used for benchmarking and for creating and post-processing large-scale point cloud maps (Kulmer, Leitenstern, et al., 2025; Leitenstern et al., 2025). Additionally, PointCloudCrafter facilitated the development of novel LiDAR calibration approaches by extracting and isolating specific point cloud objects from continuous ROS 2 recordings (Tahiraj et al., 2024, 2026). Beyond these projects, PointCloudCrafter is used for analyses such as inspecting sensor drivers for timestamp anomalies or packet losses that result in incomplete or irregular point clouds.

Although it is difficult to track the exact adoption of open-source utilities, PointCloudCrafter has been rigorously tested across domains such as calibration, localization, mapping, and perception. By providing a middleware-aware, schema-agnostic CLI toolkit, PointCloudCrafter lowers the barrier for converting raw sensor data into analysis-ready datasets. Its focus on ease of use, reproducibility, and preservation of per-point metadata makes it applicable across multiple domains that rely on three-dimensional point cloud data.

Availability

The source code is available at https://github.com/TUMFTM/PointCloudCrafter. The Python package is available at https://pypi.org/project/pointcloudcrafter. The published wheel targets Linux `x86_64` (`manylinux`) with CPython $\geq$ 3.12; users on other platforms, architectures, or Python versions should build from source or use the Docker image. The Docker image is available on the GitHub Container Registry at https://github.com/TUMFTM/PointCloudCrafter/pkgs/container/pointcloudcrafter and can be pulled via `ghcr.io/tumftm/pointcloudcrafter:latest`. The documentation is available at https://tumftm.github.io/PointCloudCrafter. The license is Apache-2.0.

AI usage disclosure

Artificial intelligence tools were used in a limited and transparent manner during the development of this work. Claude Opus 4.5 was used to assist with code structuring, to generate code comments, as well as to write test cases. GitHub Copilot was used to review pull requests. The manuscript text was reviewed for grammar and clarity using OpenAI GPT-5.2 and Grammarly, and selected passages were rephrased to improve readability and flow. All scientific content, software design decisions, and final wording were reviewed and approved by the authors, who retain full responsibility for the work.

Acknowledgements

The authors acknowledge the developers and maintainers of the Point Cloud Library (PCL) (Rusu & Cousins, 2011), upon which PointCloudCrafter is built. We also acknowledge the contributions of student researchers at the Institute of Automotive Technology at the Technical University of Munich who supported the development and evaluation of this software. This work was funded by basic research funds of the Institute of Automotive Technology at the Technical University of Munich.

References

Bazzano, C. F., Alves, L. F. G., Telles, G. P., & Trivella, D. B. B. (2025). Labeled Dataset of X-ray Protein Ligand Images in 3D Point Cloud and Validated Deep Learning Models. *Scientific Data*, *12*, 1726. https://doi.org/10.1038/s41597-025-06002-8

Caesar, H., Bankiti, V., Lang, A., Vora, S., Liong, V., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., & Beijbom, O. (2020). nuScenes: A Multimodal Dataset for Autonomous Driving. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 11618–11628. https://doi.org/10.1109/CVPR42600.2020.01164

Foote, T. (2013). tf: The Transform Library. *2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*, 1–6. https://doi.org/10.1109/TePRA.2013.6556373

Geiger, A., Lenz, P., & Urtasun, R. (2012). Are We Ready for Autonomous Driving? The

KITTI Vision Benchmark Suite. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 3354–3361. https://doi.org/10.1109/CVPR.2012.6248074

Girardeau-Montaut, D. (2026). *CloudCompare [GPL Software]* (Version 2.13). http://www.cloudcompare.org

Kulmer, D., Leitenstern, M., Weinmann, M., & Lienkamp, M. (2025). OpenLiDARMap: Zero-Drift Point Cloud Mapping Using Map Priors. *Proceedings of the 11th International Conference on Vehicle Technology and Intelligent Transport Systems - VEHITS*, 178–188. https://doi.org/10.5220/0013405400003941

Kulmer, D., Tahiraj, I., & Lienkamp, M. (2025). *Autonomous Driver Licence (ADL) Dataset* [Data set]. TUM. https://doi.org/10.14459/2025mp1785105

Lamprecht, S. (2019). Pyoints: A Python Package for Point Cloud, Voxel and Raster Processing. *Journal of Open Source Software*, *4*(36), 990. https://doi.org/10.21105/joss.00990

Leitenstern, M., Alten, M., Bolea-Schaser, C., Kulmer, D., Weinmann, M., & Lienkamp, M. (2025). FlexCloud: Direct, Modular Georeferencing and Drift-Correction of Point Cloud Maps. *Proceedings of the 11th International Conference on Vehicle Technology and Intelligent Transport Systems - VEHITS*, 157–165. https://doi.org/10.5220/0013359600003941

Parigger, L., Krassnigg, A., Hetmann, M., Hofmann, A., Gruber, K., Steinkellner, G., & Gruber, C. C. (2024). CavitOmiX Drug Discovery: Engineering Antivirals with Enhanced Spectrum and Reduced Side Effects for Arboviral Diseases. *Viruses*, *16*(8), 1186. https://doi.org/10.3390/v16081186

PDAL-Contributors. (2024). *PDAL Point Data Abstraction Library* (Version 2.8.3). Zenodo. https://doi.org/10.5281/zenodo.2616780

Pomerleau, F., Colas, F., Siegwart, R., & Magnenat, S. (2013). Comparing ICP Variants on Real-World Data Sets. *Autonomous Robots*, *34*(3), 133–148. https://doi.org/10.1007/s10514-013-9327-2

Rusu, R. B., & Cousins, S. (2011). 3D Is Here: Point Cloud Library (PCL). *2011 IEEE International Conference on Robotics and Automation*, 1–4. https://doi.org/10.1109/ICRA.2011.5980567

Sauerbeck, F., Kulmer, D., Pielmeier, M., Leitenstern, M., Weiß, C., & Betz, J. (2023). Multi-LiDAR Localization and Mapping Pipeline for Urban Autonomous Driving. *2023 IEEE SENSORS*, 1–4. https://doi.org/10.1109/SENSORS56945.2023.10325207

Schmid, M. F., Massey, J. D., & Giometto, M. G. (2025). PointClouds.jl: Fast & Flexible Processing of Lidar Data. *Journal of Open Source Software*, *10*(111), 7952. https://doi.org/10.21105/joss.07952

Tahiraj, I., Edinger, M., Kulmer, D., & Lienkamp, M. (2026). *CaLiV: LiDAR-to-Vehicle Calibration of Arbitrary Sensor Setups*. https://doi.org/10.48550/arXiv.2504.01987

Tahiraj, I., Fent, F., Hafemann, P., Ye, E., & Lienkamp, M. (2024). GMMCalib: Extrinsic Calibration of LiDAR Sensors Using GMM-Based Joint Registration. *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 8555–8562. https://doi.org/10.1109/IROS58592.2024.10801262

Tempfli, L., Rivera, E., & Lienkamp, M. (2025). *VESPA: Towards Un(Human)supervised Open-World Pointcloud Labeling for Autonomous Driving*. https://doi.org/10.48550/arXiv.2507.20397

Zhou, Q.-Y., Park, J., & Koltun, V. (2018). *Open3D: A Modern Library for 3D Data Processing*. https://doi.org/10.48550/arXiv.1801.09847