The Journal of Open Source Software

LiGuard: Interactively and Rapidly Create Point-Cloud and Image Processing Pipelines

Muhammad Shahbaz [1] **and Shaurya Agarwal** [1]

1 University of Central Florida, USA

DOI: 10.21105/joss.06751

Software

- Review
- Repository
- Archive

Editor: Hugo Ledoux

Reviewers:

- @chenzhaiyu
- @tgoelles

Submitted: 16 April 2024
Published: 23 June 2025

Summary

There is a growing interest in the development of lidar-based applications in domains like robotics, autonomous driving, traffic monitoring, infrastructure inspection, and many areas of environmental surveying and mapping. In many cases, these applications rely on solutions that are not end-to-end but rather a sequential combination (a pipeline) of individual data processing functions such as data readers, pre-processors, application-specific algorithms or models, post-processors, visualizers, etc. Creating such multi-step solutions often requires extensive revisions of the proposed processing pipeline, including enabling or disabling certain functions, fine-tuning parameters, and even completely replacing parts of the solution. Such revisions pose challenges to the speed of research and even to the reuse of it, due to interdependencies among functions. To address this issue, `LiGuard` is presented. It is a software framework that provides an easy-to-use graphical user interface (GUI) that helps researchers rapidly build and test their point cloud (and corresponding image) processing pipelines. It allows them to dynamically add/remove/reorder functions, adjust the parameters of those functions, and visualize results in a live, interactive manner compared to other packages where such modifications often require re-executing the program script. Moreover, because it creates all the meta files in structured directories, it allows easy sharing of the created pipelines or the individual functions used in those pipelines.

`LiGuard` features, out of the box, data reading for many common dataset formats, including support for reading calibration and label data. Moreover, it provides an increasing list of commonly used algorithm components ranging from basic data preprocessors to advanced object detection and tracking algorithms. Additionally, it establishes a straightforward standard for adding custom functions/algorithms, allowing users to integrate unique components into their pipelines. For the latest updates on supported features, please see LiGuard's documentation.

Note: `LiGuard` is built on many standard Python libraries and packages for data processing and visualization, including Open3D (Zhou et al., 2018), OpenCV (Bradski, 2000), Numpy (Harris et al., 2020), and SciPy (Virtanen et al., 2020). It is designed to be used interactively and is, therefore, not ideal for processing very large datasets. We provide a utility `liguard_cmd` that processes data slightly faster by removing GUI and by leveraging Dask (Dask Development Team, 2016) for parallel processing. However, since process pipelines are mostly sequential, the output generation is as fast as the slowest step in the pipeline.

Statement of Need

Recent advancements in lidar technology have significantly broadened its applicability across various domains, making it a potential sensor for precise measurement. Innovations such as solid-state lidar and nanophotonics-based devices have enhanced the performance, reliability, and cost-effectiveness of these sensors, enabling their use in diverse fields including autonomous driving, environmental monitoring, and industrial automation (Li et al., 2022). Additionally,

the integration of advanced optics and beam steering technologies has improved the spatial resolution and operational efficiency of lidar systems, facilitating their deployment in complex environments for applications ranging from agriculture to urban planning (Kim et al., 2022; Zhao et al., 2022). As the lidar sensors are becoming more and more mainstream, there is an increasing need for software tools that help standardize rapid research and discovery, and facilitate easy reproducibility of experiments.

`LiGuard` fulfills a critical need for interactive research in lidar data processing, where iterative adjustments and refinements to processing pipelines are essential and are otherwise time consuming. While it is not optimized for large-scale datasets such as pointcloudset (Goelles et al., 2021) or focus operating-system-wide utility and integration such as ROS (Macenski et al., 2022), its user-friendly interface allows researchers to dynamically modify and visualize their workflows, facilitating rapid experimentation and adaptation. Although PDAL (Contributors, 2024), a notable library for creating point cloud processing pipelines, offers much more versatile control over pipelines by providing abstract access, filtering, exploitation, and workflow management capabilities, it lacks interactivity and live parameter tuning which is often crucial in fine-tuning multi-step pipelines. PCL (Rusu & Cousins, 2011), Open3D (Zhou et al., 2018), and others such as laspy (Brown & Montaigu, 2012), pyntcloud (Pyntcloud Development Team, 2021), Pyoints (Lamprecht, 2019), etc., are low-level libraries that offer fine access to underlying data and are designed to be used as a base for the development of versatile 3D applications. `LiGuard` is also primarily built on Open3D for GUI and 3D visualizations.

`LiGuard` segregates the data processing pipeline from the application logic, making it a modular, reusable, and extensible framework. It is, at its core, a combination of five sub-modules, (1) Data I/O, (2) Process Configuration, (3) Inter-Process Data Sharing, (4) Data Processing, and (5) Visualization. During its development, a great focus was put towards minimizing redundant efforts by abstracting common tasks. Therefore, Data I/O, Inter-Process Data Sharing, and Visualization sub-modules are designed to operate seamlessly in the background. While these components can be modified by contributors, they are intentionally abstracted to relieve researchers from the complexities of efficient data reading, process management for interactivity, and visualization tasks. This, in turn, allows researchers to focus on two main aspects of their point-cloud data processing pipelines: the processes they need to execute on the data, and the configuration of those processes. `LiGuard` facilitates this by allowing users to create custom functions along their YAML configuration files. These custom functions can then be used without restarting the application.

Architecture

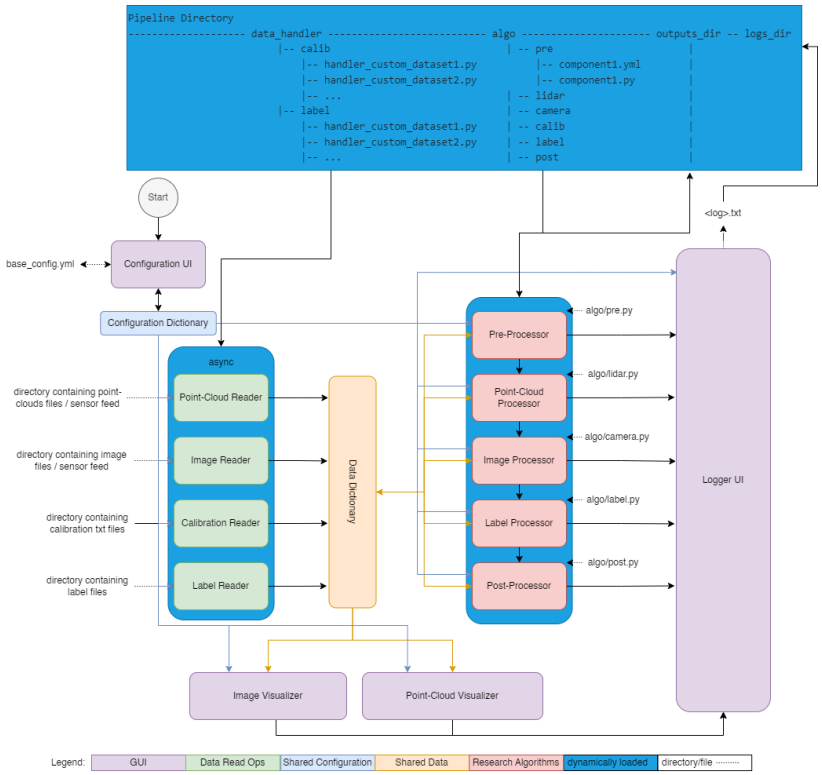

Figure 1: `LiGuard`'s Architecture

`LiGuard` is built on top of the Open3D (Zhou et al., 2018) and OpenCV (Bradski, 2000) libraries, allowing researchers and contributors to leverage the extensive functionalities provided by these libraries. A high-level architecture is presented in Figure 1 consisting of five main components: (1) GUI (purple), (2) Data Handlers (green), (3) Shared Configuration Dictionary (blue), (4) Shared Data Dictionary (orange), and (5) Research Algorithms. The GUI component is responsible for creating an interface for researchers to interact with the framework and for visualizations of both lidar and image data. The Data Handlers component is responsible for reading data from disk/sensor(s). The Shared Configuration Dictionary and Shared Data Dictionary components are responsible for sharing configuration and data, respectively, between different components of the framework. The Research Algorithms component is responsible for implementing the algorithms that process the data. The darker blue rectangular box at the top shows the `pipeline directory` where user-created data handler(s) and algorithm(s) reside along with the meta files (YAML, .yml). Please note that the research algorithms are analogous to the functions mentioned in the summary above.

Contributions

M.S. and S.A. conceived the idea of `LiGuard`. M.S. developed the framework, wrote the documentation, and the manuscript. S.A. reviewed the manuscript and provided feedback. Both authors have read and agreed to the published version of the manuscript.

Acknowledgements

We thank Dr. Karan Sikka for his great support and guidance throughout the development of `LiGuard`.

References

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Brown, G., & Montaigu, T. (2012). Laspy. *URL: Https://Laspy.readthedocs.io/En/Latest*.

Contributors, P. (2024). *PDAL point data abstraction library*. https://doi.org/10.5281/zenodo.10884408

Dask Development Team. (2016). *Dask: Library for dynamic task scheduling*. http://dask.pydata.org

Goelles, T., Schlager, B., Muckenhuber, S., Haas, S., & Hammer, T. (2021). pointcloudset: Efficient analysis of large datasets of point clouds recorded over time. *Journal of Open Source Software*, *6*(65), 3471. https://doi.org/10.21105/joss.03471

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Kim, T.-S., Jung, J., Hong, M., & Jung, Y.-H. (2022). An entropy analysis-based window size optimization scheme for merging LiDAR data frames. *Sensors*. https://doi.org/10.3390/s22239293

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

Li, N., Ho, C. P., Xue, J., Lim, L. W., Chen, G., Fu, Y. H., & T. Lee, L. Y. (2022). A progress review on solid-state LiDAR and nanophotonics-based LiDAR sensors. *Laser & Photonics Review*. https://doi.org/10.1002/lpor.202100511

Macenski, S., Foote, T., Gerkey, B., Lalancette, C., & Woodall, W. (2022). Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, *7*(66), eabm6074. https://doi.org/10.1126/scirobotics.abm6074

Pyntcloud Development Team. (2021). *Pyntcloud*. https://pyntcloud.readthedocs.io/en/latest/introduction.html

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

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., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2

Zhao, S., Chen, J., & Shi, Y. (2022). All-solid-state beam steering via integrated optical phased array technology. *Micromachines*. https://doi.org/10.3390/mi13060894

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