LiGuard: A GUI-powered Python Framework for Processing Point-Cloud Data for Intelligent Transportation Systems

04 April 2024

1 Summary

There is a growing interest in the development of LiDAR-based applications for Intelligent Transportation Systems (ITS) primarily in autonomous driving, traffic monitoring, and road safety domains. However, to conduct research in these areas, researchers often need to develop custom software for their case specific requirements, leading to duplication of efforts and lack of standardization. To address this issue, a GUI-powered Python framework, `LiGuard` is presented. `LiGuard` allows building dynamic point-cloud (and accompanying image) data processing pipelines by decoupling research algorithms from framework's source code. This dynamic design facilitates users in focusing on researching novelties rather than the intricacies of feature-complete software development. The decoupled design also aids ablation studies by providing researchers the ability to dynamically execute algorithms using a user-friendly GUI. Additionally, it allows them to incorporate editable parameters of their algorithms into the framework's interface, making it easier to fine-tune their algorithms interactively. `LiGuard` supports both individual (live) and bulk (batch) processing, thus facilitating testing and visual inspections on selected test cases before processing large-scale data.

2 Statement of Need

The research on point-cloud data is challenging as the available libraries such as PCL (Rusu and Cousins 2011), Open3D (Zhou, Park, and Koltun 2018), and other (Brown and Montaigu 2012; Pyntcloud Development Team 2021; Hunter 2007; Lamprecht 2019), etc. focus single point-cloud frames, and although some efforts such as (Goelles et al. 2021) attend to processing bulk data, the focus is still developing a python package that can be imported to custom, often hard-coded scripts, thus imposing limitations on reusability and scalability.

`LiGuard` tackles the issue by segregating 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 is put towards abstracting away the tasks that foster duplication of efforts, therefore, Data I/O, Inter-Process Data Sharing, and Visualization sub-modules are developed in such a way that they, although allow modifications from development contributors, are running behind the curtains for researchers to relieve them of the burden of reading data efficiently, creating logic to keep the processing as interactive as possible (often requiring in-depth understanding of multi-threading), and visualizing (to some extent the) outcomes before starting bulk process loop, etc. This, in turn, allows researchers to focus on two main things 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 (a) providing YAML formatted configuration files that are later loaded into GUI allowing easy manipulations of the parameters, and (b) allowing researchers to directly create corresponding (to configuration) functions that are automatically called along with configuration parameters and data.

3 Architecture

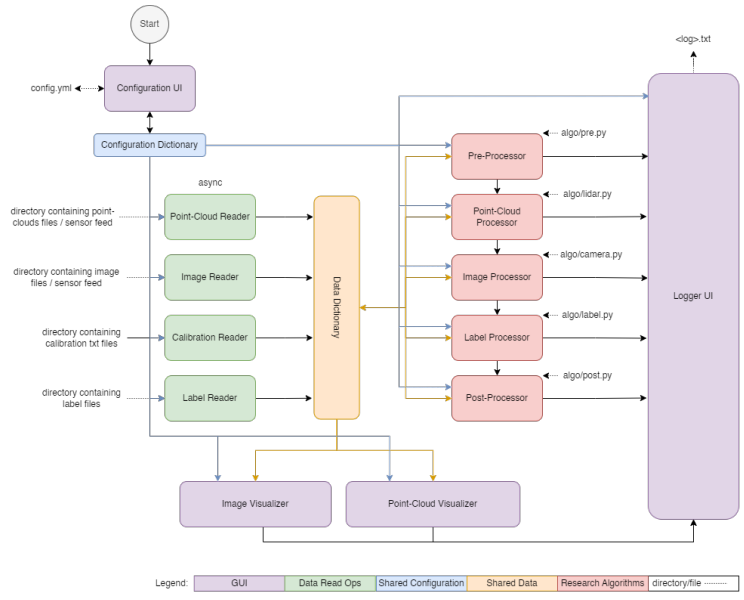

Figure 1: `LiGuard`'s Architecture

`LiGuard` is built on top of the Open3D (Zhou, Park, and Koltun 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 (related to the five sub-modules mentioned in Section 2): (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 a user-friendly 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 between different components of the framework. The Research Algorithms component is responsible for implementing the algorithms that process the data. Please note that the research algorithms are analogous to the data processes mentioned in Section 2.

4 Usage

A tutorial with technical details is available under `Usage` at our `Github` repository. There, we demonstrate the capabilities of `LiGuard` by implementing a simple pipeline of processing samples from KITTI (Geiger et al. 2013) dataset. The pipeline incorporates reading point-cloud, image, and (KITTI's standard) label data, followed by cropping point-clouds, removing out-of-bound and low point-density labels, and storing the processed dataset in OpenPCDet(Team 2020) standard.

5 Contributions

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

6 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, Grant, and T Montaigu. 2012. "Laspy." *URL: Https://Laspy.readthedocs.io/En/Latest.*

Geiger, Andreas, Philip Lenz, Christoph Stiller, and Raquel Urtasun. 2013. "Vision Meets Robotics: The Kitti Dataset." *The International Journal of Robotics Research* 32 (11): 1231–37.

Goelles, Thomas, Birgit Schlager, Stefan Muckenhuber, Sarah Haas, and Tobias Hammer. 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.

Hunter, J. D. 2007. "Matplotlib: A 2D Graphics Environment." *Computing in Science & Engineering* 9 (3): 90–95. https://doi.org/10.1109/MCSE.2007.55.

Lamprecht, Sebastian. 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.

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

Rusu, Radu Bogdan, and Steve Cousins. 2011. "3D is here: Point Cloud Library (PCL)." In *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China: IEEE.

Team, OpenPCDet Development. 2020. "OpenPCDet: An Open-Source Toolbox for 3D Object Detection from Point Clouds." https://github.com/open-mmlab/OpenPCDet.

Zhou, Qian-Yi, Jaesik Park, and Vladlen Koltun. 2018. "Open3D: A Modern Library for 3D Data Processing." *arXiv:1801.09847*.