




# small\_gicp: Efficient and parallel algorithms for point cloud registration

Kenji Koide <sup>1</sup>✉

<sup>1</sup> National Institute of Advanced Industrial Science and Technology (AIST), Japan ✉ Corresponding author

DOI: [10.21105/joss.06948](https://doi.org/10.21105/joss.06948)

## Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Patrick Diehl](#)  

## Reviewers:

- [@versatran01](#)
- [@abougouffa](#)

Submitted: 18 June 2024

Published: 09 August 2024

## License

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

## Summary

Point cloud registration is a task of aligning two point clouds measured by 3D ranging sensors, for example, LiDARs and range cameras. Iterative point cloud registration, also known as fine registration or local registration, iteratively refines the transformation between point clouds starting from an initial guess. Each iteration involves a proximity-based point correspondence search and the minimization of the distance between corresponding points, continuing until convergence. Iterative closest point (ICP) and its variants, such as Generalized ICP, are representative iterative point cloud registration algorithms. They are widely used in applications like autonomous vehicle localization ([Kim et al., 2022](#)), place recognition ([Wang et al., 2020](#)), and object classification ([Izadinia & Seitz, 2020](#)). Since these applications often require real-time or near-real-time processing, speed is a critical factor in point cloud registration routines.

**small\_gicp** provides efficient and parallel algorithms to create an extremely fast point cloud registration pipeline. It offers parallel implementations of downsampling, nearest neighbor search, local feature extraction, and registration to accelerate the entire process. **small\_gicp** is implemented as a header-only C++ library with minimal dependencies to offer efficiency, portability, and customizability.

## Statement of need

There are several point cloud processing libraries, and PCL ([Rusu & Cousins, 2011](#)), Open3D ([Zhou et al., 2018](#)), libpointmatcher ([Pomerleau et al., 2013](#)) are commonly used in real-time applications owing to their performant implementations. Although they offer numerous functionalities, including those required for point cloud registration, they present several challenges for practical applications and scientific research.

**Processing speed:** A typical point cloud registration pipeline includes processes such as downsampling, nearest neighbor search (e.g., KdTree construction), local feature estimation, and registration error minimization. PCL and Open3D support multi-threading only for parts of these processes (feature estimation and registration error minimization), with the remaining single-threaded parts often limiting the overall processing speed. Additionally, the multi-thread implementations in these libraries can have significant overheads, reducing scalability to many-core CPUs. These issues make it difficult to meet real-time processing requirements, especially on low-specification CPUs. It is also difficult to fully utilize the computational power of modern high-end CPUs.

**Customizability:** Customizing the internal workings (e.g., replacing the registration cost function or changing the correspondence search method) of existing implementations is challenging due to hard-coded processes. This poses a significant hurdle for research and development, where

testing new cost functions and search algorithms is essential.

**small\_gicp:** To address these issues and accelerate the development of point cloud registration-related systems, we designed small\_gicp with the following features:

- Fully parallelized point cloud preprocessing and registration algorithms with minimal overhead, offering up to 2x speed gain in single-threaded scenarios and better scalability in multi-core environments.
- A modular and customizable framework using C++ templates, allowing easy customization of the algorithm's internal workings while maintaining efficiency.
- A header-only C++ library implementation for easy integration into user projects, with Python bindings provided for collaborative use with other libraries (e.g., Open3D).

## Functionalities

**small\_gicp** implements several preprocessing algorithms related to point cloud registration, and ICP variant algorithms (point-to-point ICP, point-to-plane ICP, and Generalized ICP based on distribution-to-distribution correspondence).

- Downsampling
  - Voxelgrid sampling
  - Random sampling
- Nearest neighbor search and point accumulation structures
  - KdTree
  - Linear iVox (supports incremental points insertion and LRU-cache-based voxel deletion) (Bai et al., 2022)
  - Gaussian voxelmap (supports incremental points insertion and LRU-cache-based voxel deletion) (Kenji Koide & Banno, 2021)
- Registration error functions
  - Point-to-point ICP error (Zhang, 1994)
  - Point-to-plane ICP error
  - Generalized ICP error (Segal et al., 2009)
  - Robust kernels
- Least squares optimizers
  - GaussNewton optimizer
  - LevenbergMarquardt optimizer

## Benchmark results

- Single-threaded and multi-threaded (6 threads) small\_gicp::voxelgrid\_sampling are approximately 1.3x and 3.2x faster than pcl::VoxelGrid, respectively.
- Multi-threaded construction of small\_gicp::KdTree can be up to 6x faster than that of nanoflann.
- Single-threaded small\_gicp::GICP is about 2.4x faster than pcl::GICP, with the multi-threaded version showing better scalability.

More details can be found at [https://github.com/koide3/small\\_gicp/blob/master/BENCHMARK.md](https://github.com/koide3/small_gicp/blob/master/BENCHMARK.md).

## Future work

The efficiency of nearest neighbor search significantly impacts the overall performance of point cloud registration. While small\_gicp currently offers efficient and parallel implementations of KdTree and voxelmap, which are general and useful in many situations, there are other

nearest neighbor search methods that can be more efficient under mild assumptions about the point cloud measurement model (e.g., projective search (Serafin & Grisetti, 2015)). We plan to implement these alternative neighbor search algorithms to further enhance the speed of the point cloud registration process. The design of `small_gicp`, where nearest neighbor search and pose optimization are decoupled, facilitates the easy integration of these new search algorithms.

## Acknowledgements

This work was supported in part by JSPS KAKENHI Grant Number 23K16979.

## References

- Bai, C., Xiao, T., Chen, Y., Wang, H., Zhang, F., & Gao, X. (2022). Faster-LIO: Lightweight tightly coupled lidar-inertial odometry using parallel sparse incremental voxels. *IEEE Robotics and Automation Letters*, 7(2), 4861–4868. <https://doi.org/10.1109/LRA.2022.3152830>
- Izadinia, H., & Seitz, S. M. (2020). Scene recomposition by learning-based ICP. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR2020)*, 930–939. <https://doi.org/10.1109/cvpr42600.2020.00101>
- Kenji Koide, S. O., Masashi Yokozuka, & Banno, A. (2021). Voxelized GICP for fast and accurate 3D point cloud registration. *IEEE International Conference on Robotics and Automation (ICRA2021)*, 11054–11059. <https://doi.org/10.1109/ICRA48506.2021.9560835>
- Kim, K., Im, J., & Jee, G. (2022). Tunnel facility based vehicle localization in highway tunnel using 3D LIDAR. *IEEE Transactions on Intelligent Transportation Systems*, 23(10), 17575–17583. <https://doi.org/10.1109/TITS.2022.3160235>
- 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, May). 3D is here: Point cloud library (PCL). *IEEE International Conference on Robotics and Automation (ICRA2011)*. <https://doi.org/10.1109/ICRA.2011.5980567>
- Segal, A., Haehnel, D., & Thrun, S. (2009). Generalized-ICP. *Robotics: Science and Systems*, 2, 435. <https://doi.org/10.15607/rss.2009.v.021>
- Serafin, J., & Grisetti, G. (2015). NICE: Dense normal based point cloud registration. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2015)*, 742–749. <https://doi.org/10.1109/IROS.2015.7353455>
- Wang, H., Wang, C., & Xie, L. (2020). Intensity scan context: Coding intensity and geometry relations for loop closure detection. *IEEE International Conference on Robotics and Automation (ICRA2020)*, 2095–2101. <https://doi.org/10.1109/ICRA40945.2020.9196764>
- Zhang, Z. (1994). Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13(2), 119–152. <https://doi.org/10.1007/BF01427149>
- Zhou, Q.-Y., Park, J., & Koltun, V. (2018). Open3D: A modern library for 3D data processing. *arXiv:1801.09847*. <https://doi.org/10.48550/arXiv.1801.09847>