Triangle Descriptor Loop Detection Method Based on Faster-LIO

Siqi Wang [1], He Huang [1], Junxing Yang [1*], Junyang Bian [1], Shan Jiang [1]

[1] School of Geomatics and Urban Spatial Informatics, Beijing University of Civil Engineering and Architecture, Beijing, China - w2248183443@163.com, huanghe@bucea.edu.cn, yangjunxing@bucea.edu.cn, bjy1066702005@163.com, 2108570023131@stu.bucea.edu.cn

Keywords: SLAM, Triangle descriptor, LIDAR, Position recognition, Loop closure detection.

Abstract:

This paper introduces an enhanced approach to loop closure detection in Simultaneous Localization and Mapping (SLAM) by integrating the Faster-LIO framework with Stable Triangle Descriptors (STD). SLAM, essential in autonomous driving and augmented reality, often encounters cumulative errors affecting mapping accuracy. Traditional detection methods based on sensor data like camera images and LiDAR point clouds struggle with environmental changes that alter scene appearance and geometry. Our approach utilizes LiDAR sensors and STD, exploiting the geometric stability of triangles to maintain robustness against rotational and translational changes. The process involves storing triangle descriptors from key frames in a hash table within the Faster-LIO framework, a voxel-based LiDAR-Inertial Odometry optimized for efficiency and speed. These descriptors are then matched across frames using a voting mechanism to ensure reliable loop closure detection. Validation on the KITTI dataset and a proprietary subterranean parking garage dataset demonstrates that this integration not only enhances loop closure detection but also simplifies computational demands by avoiding complex tree structures. This method shows promise for broader applications in robotics and autonomous systems, with future research focusing on refining the descriptor and expanding its applicability to other sensor modalities.

1. Introduction

Simultaneous Localization and Mapping (SLAM) is a cornerstone technology in fields such as autonomous driving, augmented reality, and virtual reality. Its incremental nature in positioning and mapping inevitably leads to cumulative errors in large-scale mapping. Loop closure detection methods mitigate these errors by establishing constraints between current and historical frames, thus necessitating position recognition to ascertain whether the data collection process has revisited a historical location. Position recognition involves determining whether two sensor measurements (e.g., camera images, LiDAR point clouds) were collected in the same scene.

The extensive use of cameras has led to the development of many vision-based SLAM systems. However, these loop closure detection methods struggle with significant alterations caused by changes in lighting, appearance, or perspective. In contrast, Light Detection and Ranging (LiDAR) sensors capture the structural information of the environment directly, remaining unaffected by changes in lighting and appearance. The advent of low-cost, high-performance LiDAR has further expanded its application in robotics. LiDAR-based position recognition solutions must exhibit rotational and translational invariance,.

Triangles, being more stable and possessing rotational and translational invariance compared to other polygonal shapes, are utilized in this study to encode any three arbitrary keypoints in a scene using Stable Triangle Descriptors, building upon the Faster-LIO framework. This approach enables position [1]recognition through descriptor matching, facilitating loop closure detection by increasing frame-to-frame constraints. In Faster-LIO, voxel point clouds are constructed from key frames to create planar voxels meeting a specific threshold, with non-conforming points designated as boundary voxels. The farthest point on a boundary voxel from planar voxels is projected as a boundary pixel value onto a planar voxel map, selecting the largest pixel value within a defined range (e.g., 4x4) as the keypoint. Kd-tree construction based on these keypoints facilitates the search for adjacent points to form triangle descriptors, which are then stored in a hash table for rapid querying and matching. The hash key values are computed using the dot product of edge lengths and normal projection vectors in the descriptors, ensuring rotational and translational invariance. During the site recognition process, the hash key values of the descriptors in the current key frame are searched, with each matched key frame receiving a vote. Following the processing and querying of all descriptors, the descriptors of the key frames with the highest vote counts are retained for subsequent loop detection steps.

2. Related

2.1 Frame

To address rapid movement and environmental noise, Xu et al. introduced the FAST-LIO framework(Xu and Zhang, 2021), a robust, tightly-coupled laser-inertial odometry system. FAST-LIO integrates LiDAR with IMU using an error state iterative Kalman filter, enhancing computational efficiency by focusing on state dimensions rather than measurement dimensions. FAST-LIO2(Xu et al., 2022) further refines this approach by omitting feature extraction and employing a dynamic kd-Tree structure, ikd-Tree(Cai, Xu, and Zhang, 2021),

* Corresponding author

which offers better performance compared to traditional dynamic data structures.

Extensive testing shows that FAST-LIO2 offers significant improvements in processing speed and robustness, particularly in high-speed environments. However, its real-time mapping capabilities in large-scale scenarios are limited by the maintenance demands of the ikd-Tree structure. Building upon FAST-LIO2, Faster-LIO(Bai et al., 2022) uses an incremental sparse voxel (iVox) data structure, which significantly speeds up point cloud registration without compromising odometry accuracy.

Overall, these advancements in LiDAR SLAM technologies emphasize improvements in speed, accuracy, and adaptability to environmental challenges, yet they also highlight the ongoing challenges in large-scale mapping and real-time data processing.

2.2 key frame

The generation of keyframes significantly impacts the performance of SLAM algorithms; selecting the correct keyframes can conserve computational resources, eliminate redundant data, and enhance both processing speed and accuracy. Existing LiDAR SLAM systems such as LeGO-LOAM and LIO-SAM utilize a fixed mechanism for keyframe generation, where keyframes are chosen based on predefined distance and temporal thresholds between frames.

Place recognition in 3D data is crucial for autonomous robot localization and has been explored through various methodologies. These approaches are typically categorized into three distinct types:

(i) local descriptors that focus on point features;

(ii) global descriptors that capture overall appearance;

(iii) learning-based methods.

Local Descriptors: Bosse and Zlot(Bosse and Zlot, 2013) directly identify keypoints on 3D data using the Gestalt Descriptor, which captures each keypoint's local neighborhood to compute a voting matrix for place recognition. Other descriptors such as PFH(Rusu, Blodow, Marton, and Beetz, 2008), SURFs(Bay, Tuytelaars, and Van Gool, 2006), and SHOT(Salti, Tombari, and Di Stefano, 2014) operate under similar frameworks but tend to be sensitive to the variability in LiDAR point cloud density and noise, lacking invariance to changes in rotation or viewpoint.

Global Descriptors: Giseop Kim and Ayoung Kim(Kim and Kim, 2018) developed Scan Context, a 2D descriptor based on the height of surrounding structures, and V. Nardari et al.(Nardari, Cohen, Chen, Liu, Arcot, Romero, and Kumar, 2021) designed a polygon descriptor for forest environment recognition, while Jiang et al.(Jiang, Zhu, and Liu, 2019) use a triangle feature-based descriptor for 2D SLAM. Global descriptors leverage general appearance attributes such as surface flatness, orientation, and height.

Learning-Based Methods: Contrasting with traditional descriptors, learning-based methods use deep learning to enhance place recognition. SegMap(Dubé, Dugas, Stumm, Nieto, Siegwart, and Cadena, 2017) utilizes semantic features for recognition, whereas OverlapNet(Chen, Läbe, Milioto, Röhling, Vysotska, Haag, Behley, and Stachniss, 2020) employs a neural network for calculating overlaps and estimating relative yaw angles between pairs of 3D scans. These approaches generally require training and benefit from GPU acceleration to manage their computational load.

The Stable Triangle Descriptor introduced here is a global descriptor that integrates three critical points. These descriptors are specifically extracted within a keyframe to accurately represent the relative distribution of key points across the frame. Unlike other global descriptors, this Stable Triangle Descriptor demonstrates enhanced invariance to rotation and translation, offering a distinct advantage in dynamic environments.

In contrast to typical polygon descriptors that are extracted in 2D space, this descriptor operates directly in 3D space. It leverages the most stable and consistent triangle formation found among polygons, thus providing a more reliable and recognizable descriptor. Moreover, this approach enables comprehensive pose estimation with full degrees of freedom, significantly streamlining the registration process by reducing time and enhancing accuracy. This makes the Stable Triangle Descriptor particularly valuable for applications requiring precise and efficient mapping and localization.

3. Method

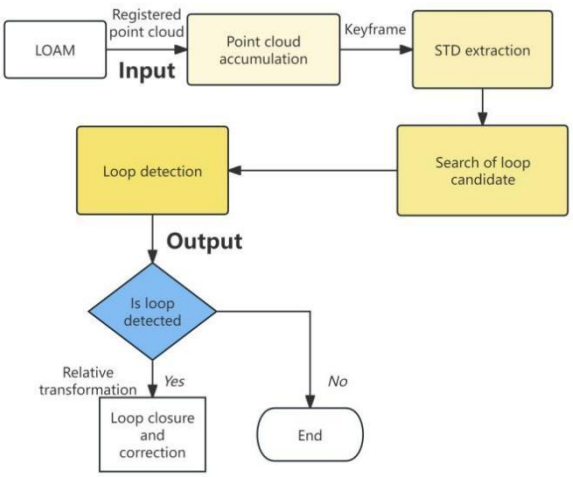

Figure 1 STD flowchart

As depicted in Figure 1, the integration of iVox with Stable Triangle Descriptors (STD) involves leveraging a k-d tree for nearest neighbor searches essential for forming triangle descriptors within STD. A k-d tree is an efficient binary tree structure where each node represents an axis-aligned hyperplane that splits the space into two halves. This tree selects the node to be split as the median point along the longest dimension to ensure compact spatial partitioning, which is particularly effective in solving k-Nearest Neighbors (kNN) problems due to its ability to handle low-dimensional data efficiently when stored in primary memory.

However, despite its efficiency in spatial queries, the k-d tree may experience a loss in performance due to deep branching during searches for nearest neighbors. Such extensive searching does not necessarily improve the accuracy of local plane estimation and can introduce inefficiencies, particularly in LiDAR-Inertial Odometry (LIO) applications where time and computational resources are critical.

In contrast, Faster-LIO utilizes a voxel-based algorithm to constrain the search within a pre-defined boundary, enhancing efficiency by preventing the extensive search depths typical of k-d trees. This approach ensures that even discarded points do not significantly affect the overall residual computations, making it more suitable for real-time applications. The voxel method avoids the extensive construction, iteration, balancing, and removal processes associated with k-d tree nodes, thereby streamlining the computation process.

3.1 iVox

Faster-LIO introduces a sparse, incremental, voxel-based LiDAR-Inertial Odometry (LIO) algorithm, differing from FastLIO2 by utilizing sparse voxels instead of k-d trees. The choice for sparse incremental voxels is motivated by two main reasons: First, unlike k-d trees, which may traverse distant branches to find a potential nearest neighbor, the search range in sparse voxel-based algorithms is confined within a preset limit, with the omission of such nearest neighbors having negligible impact on the majority of residuals. Second, voxels are more efficient since the construction, iteration, balancing, and removal of k-d tree nodes can impede LIO performance.

In the iVox framework, the point clouds are strategically stored within sparse voxel structures. This storage is facilitated through the use of a sparse hash map, which is designed to efficiently maintain records of voxels that encompass at least one point. This method optimizes memory usage by ensuring that only those voxels which are non-empty are indexed, thereby enhancing the computational efficiency and speed of access during point cloud processing tasks.

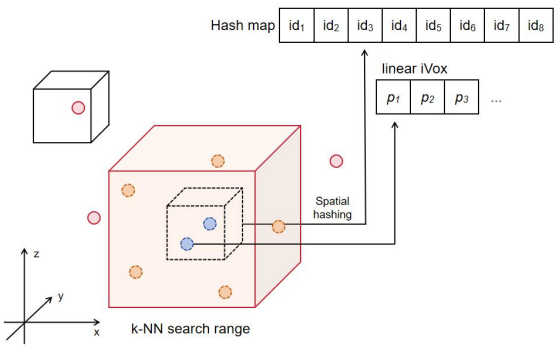

Figure 2 Diagram illustrating the mapping of 3D points to the same one-dimensional hash index.

Using the following hash function:

$$P=[p_x,\ p_y,\ p_z]^T \tag{1}$$

$$v=[p_x,\ p_y,\ p_z]^T \tag{2}$$

$$idv=\text{hash}(v)=(v_x n_x)\text{xor}(v_y n_y)\text{xor}(v_z n_y)\text{mod}N \tag{3}$$

(1)Represents a point is position in three-dimensional space.

(2) In the context, v denotes the voxel index, and s represents the size of the voxel.

(3)*idv* refers to the method used to calculate the hash value of vector v for spatial hashing. It represents the voxel index is hash value. The final modulus operation ensures that the resultant hash value falls within the size N of the hash table. This modulus operation guarantees that, even if the number obtained through the XOR operation is very large, the final hash value will be confined within the valid index range of the hash table.

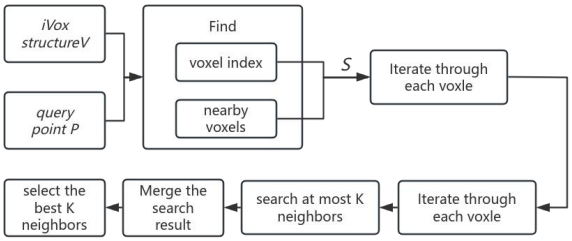

Figure 3 k-NN search flowchart

The k-NN search in the iVox framework follows a predefined range and is divided into three critical steps. Considering the iVox structure V and a query point P, the process unfolds as follows: 1) Identify the voxel index and surrounding voxels (which may include 6, 18, or 26 voxels); collectively, these voxels are referred to as S. 2) Iterate through each voxel in S, searching for up to K nearest neighbors within each voxel. 3) Aggregate all search results and select the optimal K neighbors. It is noteworthy that step 2 can be parallelized for each voxel. However, since parallel processing has already been implemented at the point cloud level, there is no need for separate parallel searches within each voxel. The k-NN search within iVox is straightforward and efficient, although it may not be as strict as tree-based algorithms, it is sufficiently robust for LIO applications.(Bai, Xiao, Chen, Wang, Zhang, and Gao, 2022)

3.2 boundary voxels and descriptor

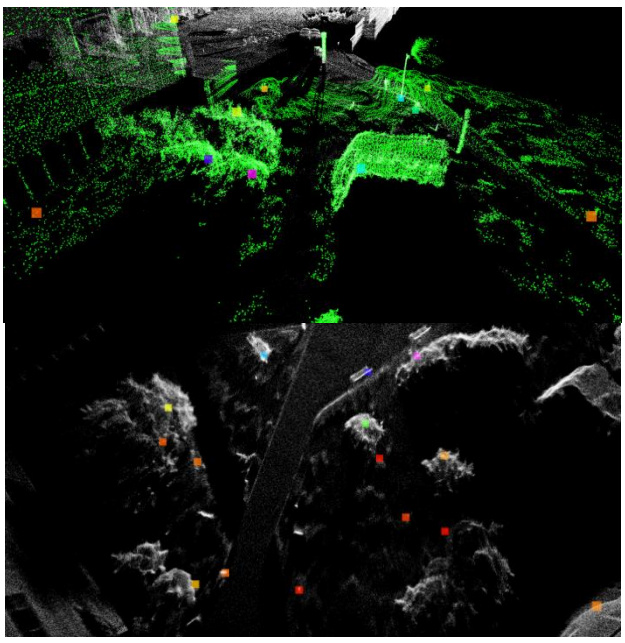

Figure 4 boundary voxels

Boundary voxels are employed as a method for extracting key points from 3D point clouds.As shown in Figure 4, these are the boundary voxels during the experimental process.In this process, the entire point cloud is first divided into voxels of a specified size. Each voxel contains a set of points, and the covariance matrix of these points is calculated to determine whether they form a plane. If it is established that a plane is formed, this

voxel is designated as a planar voxel. Subsequently, a plane is initialized, starting with any planar voxel, and is expanded by searching adjacent voxels. If neighboring voxels share the same planar normal vector as the current plane and are within a certain distance threshold, they are added to the plane. Conversely, if adjacent voxels do not lie on the same plane, they are added to the list of boundary voxels of the expanding plane.

The formula for calculating the covariance matrix of points is as follows:

$$\bar{p} = \frac{1}{N}\sum_{i=1}^{N} p_i;\ \Sigma = \frac{1}{N}\sum_{i=1}^{N} (pi - \bar{p})(pi - \bar{p})^{T} \qquad (4)$$

The first part, computes the mean vector of the point set, where pi represents the i -th point in the dataset, and N is the total number of points. The mean is derived by taking the arithmetic average of all the data points.

The second part,represents the covariance matrix, which measures the extent of variation between each point pi and the mean point $\bar{p}$. The difference between each data point pi and the mean $\bar{p}$ is first computed, then this difference vector is multiplied by its own transpose vector. The sum of these products, normalized by the total number of points N,yields the covariance matrix. The diagonal elements of the covariance matrix indicate the variance within each dimension, while the off-diagonal elements represent the covariance between different dimensions.

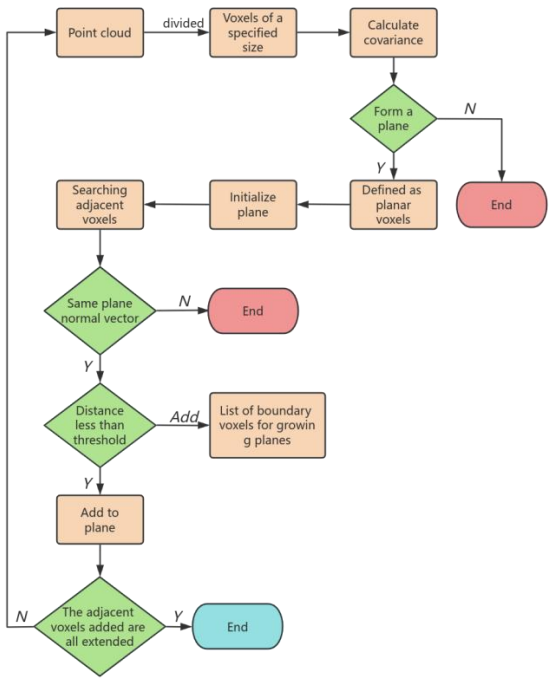

Figure 5 Boundary Voxel Search Flowchart

The Stable Triangle Descriptor is an innovative tool for 3D position recognition, represented by a six-dimensional vector that includes the lengths of a triangle's three sides and the angles between the normals of adjacent planes at each triangle vertex, as shown in Figure 1. The STD employs a triangle to encode any three key points within a scene, offering enhanced stability compared to other descriptors. Its primary advantage lies in the determinacy of its shape by either side lengths or angles, maintaining invariance under rigid transformations.

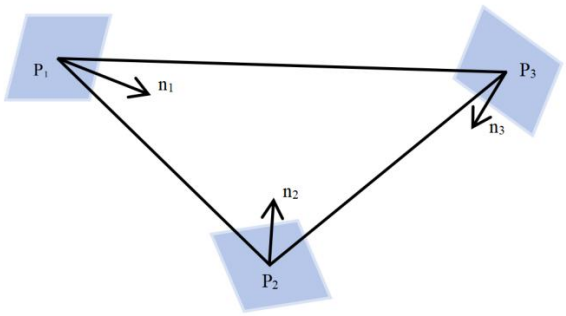

Figure 6 A standard triangle descriptor

The process begins with the extraction of triangle descriptors from key frames, utilizing a hash table as the database for efficient storage and matching. Frames are ranked based on descriptor matching scores, with the top ten frames selected as candidates. These candidates undergo geometric verification to identify valid loops, and relative transformations between the loop and candidate frames are established upon loop detection. As depicted in Figure 1, following the registration of point cloud data collected via LiDAR Odometry and Mapping (LOAM)(Zhang and Singh, 2014), the comprehensive STD process commences. It starts with the accumulation of point cloud data, followed by the extraction of triangle descriptors from key frames. Plane boundary detection involves identifying all planar regions and their boundaries within the point cloud.

Key point extraction and assembly entail locating points within the point cloud that possess unique geometric characteristics, which are vital for subsequent data processing steps such as point cloud registration and feature matching. These key points are matchable across datasets captured from various viewpoints or moments in time. The process continues with querying loop candidates, integrating historical point clouds into the hash table, and performing stable triangle descriptor matching and scoring against the current point cloud. Loop closure and correction are achieved through RANSAC and geometric verification if a loop is detected; otherwise, the process concludes without loop closure.(Yuan, Lin, Zou, Hong, and Zhang, 2023)

3.3 Methodology

Building on the foundation of Faster-LIO, this paper uses a novel application of Stable Triangle Descriptors (STDs) for encoding any three arbitrary keypoints within an environment for position recognition. This method enhances the constraints between the current and historical frames, facilitating point cloud correction.

4. Experiment

The KITTI dataset is a renowned repository in the domain of computer vision, predominantly utilized for research in autonomous vehicular technologies. This dataset encompasses a diverse array of sensor data, including outputs from stereo cameras, Light Detection and Ranging (LiDAR) scanners, Global Positioning System (GPS), and Inertial Measurement Units (IMU). The KITTI dataset was collaboratively developed by the Karlsruhe Institute of Technology in Germany and the Toyota Technological Institute, providing a comprehensive suite of high-resolution sensor data critical for the advancement of perception, navigation, and autonomous driving systems. The dataset has been pivotal in benchmarking the performance of a

multitude of computer vision algorithms, specifically in the tasks of object detection, tracking, and scene understanding within the dynamic context of real-world driving environments.

In the experimental section, in addition to utilizing the KITTI dataset, we have also employed a self-collected subterranean parking garage dataset for our experiments.

This study utilized the AGILE mobile platform as the experimental apparatus. The platform is equipped with a nine-axis IMU for precise inertial navigation, along with four infrared obstacle avoidance sensors and wheel motors for propulsion, capable of achieving a maximum speed of 5 meters per second and a payload capacity of 50 kilograms. The control system employs a high-performance industrial computer with an i7 CPU and 16GB of memory, supporting the ROS operating system to ensure robust computing power and stability. Additionally, the experimental platform is equipped with HESAI's Pandar40 LiDAR, enabling 360-degree omnidirectional scanning, high-density point cloud output, and remote distance measurement, facilitating efficient, accurate, and comprehensive environmental perception.Our experimental equipment is as shown in Figure 7.

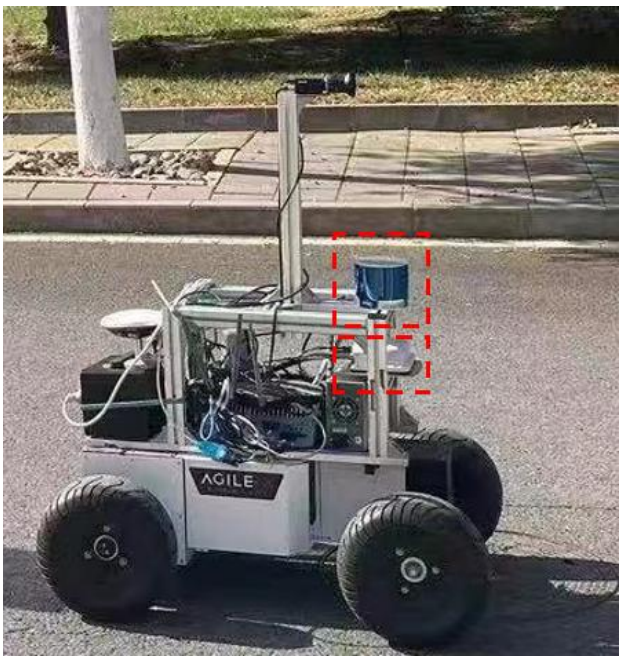

Figure 7 Experimental equipment

On the software side, this experimental platform can run on Ubuntu 20.04 and is integrated with ROS, providing powerful software support and a wide range of algorithm libraries.

4.1 Experimental and Data Comparisons

In the course of our experiments, we initially conducted validation on the KITTI dataset, which provides ground truth data. Within the KITTI dataset experiments, the ground truth is represented by the grey dashed lines. It is evident from Figures 8 and 9 that our experiments demonstrate an enhanced performance in loop closure detection compared to FasterLIO, particularly noticeable within the KITTI_00 dataset. It can be distinctly observed that the trajectory formed by our method adheres more closely to the ground truth, indicating a superior accuracy of our approach.

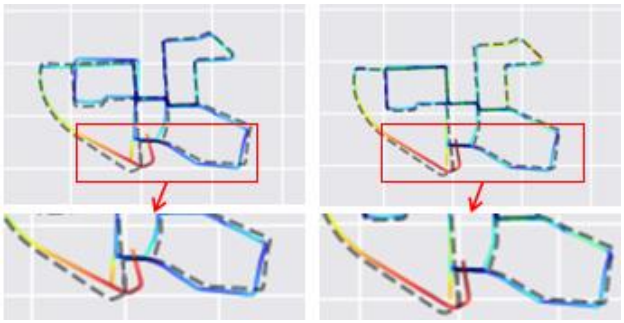

Figure 8 Comparison of the effects of dataset KITTI_00,The visualization of Faster-lio is on the left, and ours is on the right.

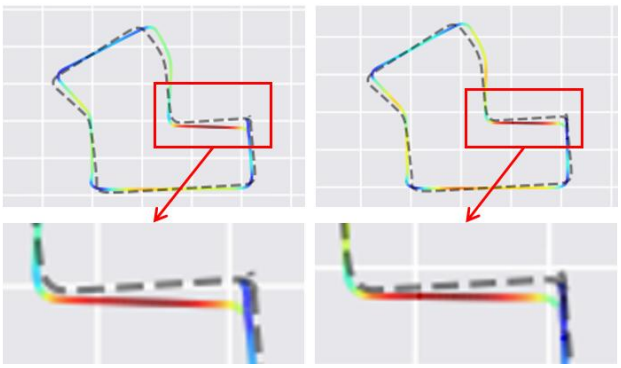

Figure 9 Comparison of the effects of dataset KITTI-07,The visualization of Faster-lio is on the left, and ours is on the right.

kitti	00				
	mean	std	rmse	max	min
fasterlio	18.18	10.67	21.08	54.48	2.17
ours	13.86	6.80	15.44	33.38	1.20

Table 1 KITTI_00 experimental data

kitti	07				
	mean	std	rmse	max	min
fasterlio	8.42	3.94	9.29	18.97	0.99
ours	7.53	3.57	8.33	15.76	0.41

Table 2 KITTI_07 experimental data

4.2 a self-collected subterranean parking garage dataset

In our proprietary dataset of point clouds collected from an underground parking garage, we conducted experiments using our enhanced methodology. Given the absence of ground truth in the subterranean parking environment, we could only assess the relative accuracy. In Figure 10, panels a, b, and c represent the three distinct point cloud datasets from our collected underground parking scenarios. Panel d depicts a magnified section within dataset c, where it is distinctly noticeable that upon the formation of a loop closure, our method successfully identified the Stable Triangle Descriptors in our self-collected dataset and accomplished matching between the current frame and historical frames. Furthermore, it is clearly observed that our approach results in a substantial correction effect.

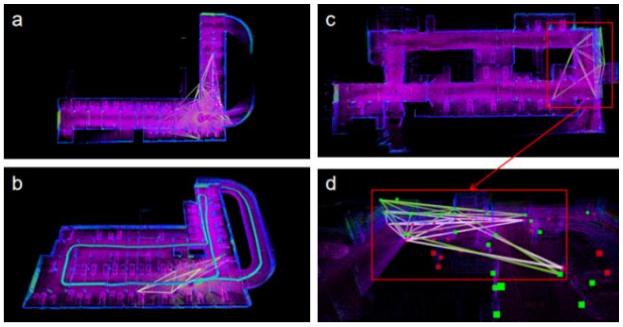

Figure 10 Experimental results and local details of three different underground parking garages.

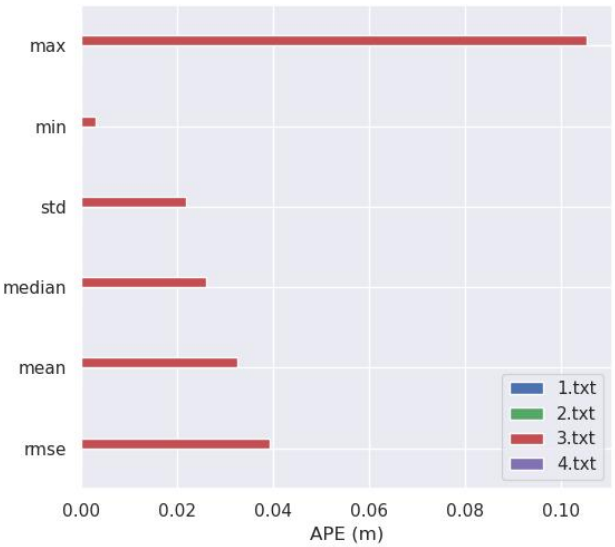

Figure 11 Figure 10c presents the absolute positional errors from four experimental trials conducted on the subterranean parking garage dataset.

In the challenging environment of an underground parking facility where no ground truth data is available, a set of four controlled experiments was conducted to rigorously evaluate the reliability and consistency of our localization methodology. The outcomes of these experimental runs are presented in Figure 11, which elucidates the performance across six pivotal statistical metrics. These metrics encompass the maximum and minimum recorded errors, offering an in-depth understanding of the performance boundaries. The standard deviation elucidates the variability within our dataset, and the median value serves as a metric for central tendency, which is less susceptible to the influence of outliers as compared to the arithmetic mean. The mean error is indicative of the average performance level. Critically, the root mean square error amalgamates a holistic measure of variance, emphasizing larger deviations. Remarkably, only one experimental instance indicated a notable deviation, underscoring the robustness of our approach amidst the intrinsic challenges associated with an underground setting.

5. Conclusion

This study examines the combined use of Faster-LIO and Stable Triangle Descriptor (STD) matching, performing numerous experiments on datasets featuring loop closures, both from public sources and those meticulously compiled by our research group. Empirical outcomes indicate that integrating these methods could improve the refinement process inherent in loop closure detection and increase the precision of position identification. Utilizing the geometric consistency provided by STDs, the approach under investigation shows significant promise in enhancing the precision of environmental mapping and navigation within autonomous systems.

Acknowledgements

Acknowledgements of support for the project/paper/author are welcome. Note, however, that for the paper to be submitted for review all acknowledgements must be anonymized.

References

Xu, W., Zhang, F., 2021. FAST-LIO: A Fast, Robust LiDAR-Inertial Odometry Package by Tightly-Coupled Iterated Kalman Filter. IEEE Robotics and Automation Letters. 6(2), 3317-3324.

Xu, W., Cai, Y., He, D., et al., 2022. FAST-LIO2: Fast Direct LiDAR-Inertial Odometry. IEEE Transactions on Robotics. 38(4), 2053-2073.

Cai, Y., Xu, W., Zhang, F., 2021. ikd-Tree: An Incremental KD Tree for Robotic Applications [EB/OL]. Accessed: October 28, 2023. https://doi.org/10.48550/arXiv.2102.10808

Bai, C., Xiao, T., Chen, Y., et al., 2022. Faster-LIO: Lightweight Tightly Coupled Lidar-Inertial Odometry Using Parallel Sparse Incremental Voxels. IEEE Robotics and Automation Letters. 7(2), 4861-4868.

Bosse, M., Zlot, R., 2013. Place recognition using keypoint voting in large 3D LiDAR datasets. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation, pp. 2677-2684.

Rusu, R. B., Blodow, N., Marton, Z. C., et al., 2008. Aligning point cloud views using persistent feature histograms. In Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3384-3391.

Bay, H., Tuytelaars, T., Van Gool, L., 2006. SURF: Speeded up robust features. In Proceedings of the European Conference on Computer Vision. Springer, pp. 404-417.

Salti, S., Tombari, F., Di Stefano, L., 2014. SHOT: Unique signatures of histograms for surface and texture description. Computer Vision and Image Understanding. 125, 251-264.

Kim, G., Kim, A., 2018. Scan context: Egocentric spatial descriptor for place recognition within 3D point cloud map. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, pp. 4802-4809.

Nardari, G. V., Cohen, A., Chen, S. W., et al., 2021. Place recognition in forests with urquhart tessellations. IEEE Robotics and Automation Letters. 6(2), 279-286.

Jiang, B., Zhu, Y., Liu, M., 2019. A triangle feature based map-to-map matching and loop closure for 2D graph SLAM. In Proceedings of the 2019 IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 2719-2725.

Dubé, R., Dugas, D., Stumm, E., et al., 2017. Segmatch: Segment based place recognition in 3D point clouds. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, pp. 5266-5272.

Chen, X., Läbe, T., Milioto, A., et al., 2020. OverlapNet: Loop Closing for LiDAR-based SLAM. In Proceedings of Robotics: Science and Systems (RSS), 2020.

Zhang, J., Singh, S., 2014. LOAM: Lidar Odometry and Mapping in Real-time. In Proceedings of the Robotics: Science and Systems, July 2014. DOI: 10.15607/RSS.2014.X.007.

Yuan, C., Lin, J., Zou, Z., et al., 2023. STD: Stable Triangle Descriptor for 3D place recognition. IEEE Robotics and Automation Letters.