Artificial Intelligence and Computing Innovations

Intelligence Science Publishing

https://www.ispublishing.com

ISSN: 3070-4944

Vo.6, No.2, 2026

Improved Cartographer-Based SLAM with LiDAR Preprocessing for Autonomous Navigation

Corbin Wexler

Southern Illinois University Edwardsville, Edwardsville, USA

cwexler44@siue.edu

Abstract: This study presents an enhanced Cartographer algorithm for simultaneous localization and mapping (SLAM) applied to multi-sensor robotic navigation systems. The proposed approach integrates LiDAR, odometer, and inertial measurement unit (IMU) data within a unified mapping framework to improve localization accuracy and environmental perception. To mitigate common issues such as occlusion, reflection, and measurement noise in LiDAR point clouds, the Point Cloud Library (PCL) is utilized for preprocessing to remove outliers and dropout points. An adaptive filtering mechanism based on the dynamic window method enables real-time obstacle avoidance and trajectory correction during autonomous navigation. The system is simulated under the ROS framework using Rviz and Gazebo environments, where results demonstrate superior mapping clarity and reduced drift compared with the original Cartographer algorithm. Field experiments confirm that the optimized approach effectively eliminates abnormal laser points, enhances robustness, and produces high-precision maps suitable for complex dynamic environments. Overall, this work provides a practical and extensible solution for intelligent robotic navigation and SLAM optimization in multi-source perception scenarios.

Keywords: Cartographer SLAM; Point Cloud Library (PCL); multi-sensor fusion; autonomous navigation; LiDAR preprocessing

1. Introduction

Laser SLAM can be classified into filter based laser SLAM and graph optimization based laser SLAM based on different mathematical optimization frameworks [1]. The filter based laser SLAM are FastSLAM and Gmapping. (1) FastSLAM algorithm is a particle filter based laser SLAM algorithm [2]. It estimates the map and the state of the robot by representing the trajectory of the robot as a set of particles and using particle filters. (2) The Gmapping algorithm can construct maps in real time [3], and the accuracy of the constructed maps ishigh, which is one of the mainstream algorithms, but the algorithm

also has some limitations, i.e., the algorithm relies too much on odometry information. Laser SLAM based on graph optimization include Hectorslam and Cartographer. (1) Hectorslam utilizes the Gaussian Newton method to solve the scan-matching problem, which has higher requirements for sensors. Disadvantages: requires a high update frequency of the radar (LRS) and low measurement noise. (2) The Cartographer algorithm is an optimization of Karto SLAM [4], and its core is the scan matching strategy for local sub-map creation and closed-loop detection of fused multi-sensor data. The front-end scan matching of the algorithm uses a combination of CSM (Correlation Scan Match) and gradient optimization to generate the local sub-map and carry out the local loop detection. The back-end of the algorithm uses branch and bound to improve the detection speed of global closed-loop detection after the sub-map construction is completed [5-6].

2. Design of Autonomous Navigation System for Robots

2.1 SLAM design

Mobile robot uses LiDAR and odometer and other fusion map building technology in order to ensure the accuracy of the map. Each sensor by the coordinates converted to uniform coordinates under the output fusion data to estimate the robot moving position. At the same time, LiDAR constantly scans the surrounding environment data and builds a complete map. SLAM localization map building research process is shown in Figure 1.

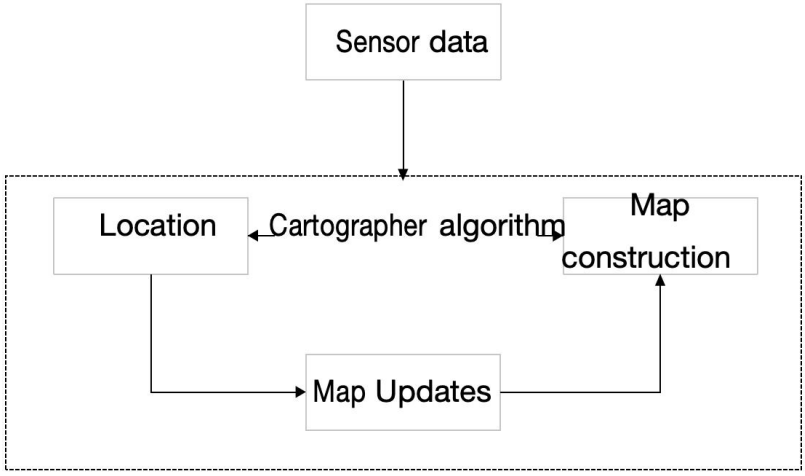


Figure 1. SLAM localization mapping

2.2 Autonomous navigation design

In the process of navigation, a more accurate global map has been established based on SLAM technology in the early stage. The particle filtering method with adaptive Monte Carlo idea can be used for navigation and localization. With the aid of the rotation of the robot's wheels recorded by the encoder's odometer and the robot's rotation angle recorded by the inertial measurement unit (IMU), the robot's moving trajectory position can be accurately localized. For dynamic changes and other dynamic obstacles, the dynamic window method can be used for localized dynamic obstacle avoidance, and the autonomous navigation research flow is shown in Figure. 2.

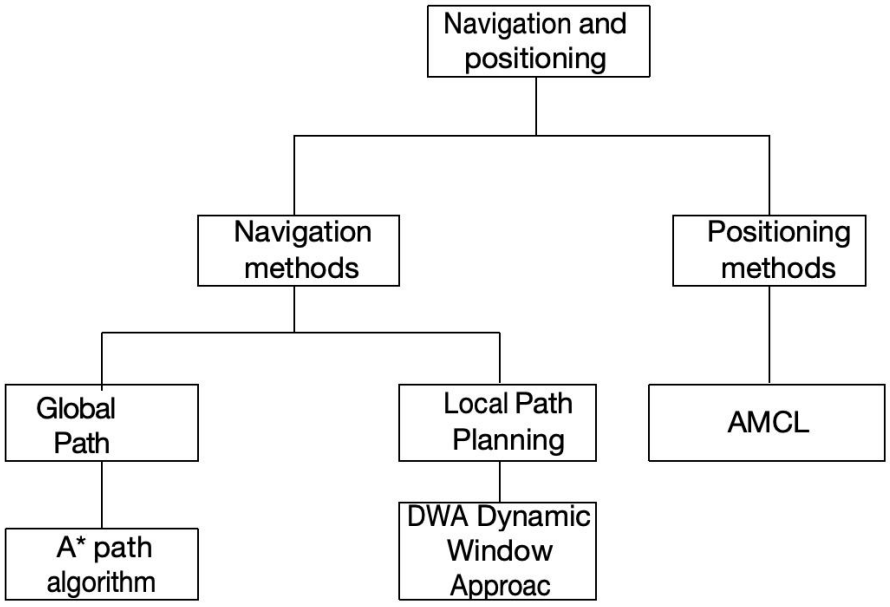


Figure 2. Navigation technology flowchart

3. Improvements to the Cartographer's algorithm

The point cloud information obtained when LiDAR is in the presence of occlusion, foggy weather, reflections, or measurement errors can produce outliers or dropouts (the appearance of nan's points). Wrong LiDAR data can affect the optimization solution. To address the problems of the proposed Cartographer algorithm, the optimization of the LIDAR preprocessing is carried out by means of the Point Cloud Library (PCL) function, which has an interface based on the ROS framework. PCL has an interface based on the ROS framework, through which data can be exchanged between the PCL point cloud library and the ROS system.PCL comes with a function library. PCL comes with a function library. There are conversion functions in the library, which can convert the processed local PCL type data to the .msg format for communication under the ROS framework, and can also visualize the processed LiDAR data with rviz in the ROS system.

The point cloud data is first loaded with the PCL library conversion function and the output is shown in Figure 3. The dropped points (points that appear as nan) in the LiDAR data are filtered out, using the PCL library function std::vector<int> indices needs to save the index labeled by the points that need to be removed first, and the pcl::removeNaN From Point Cloud (*m.cloud,*m.cloud, indices) function to Remove dropped points from the point cloud of LiDAR data, and choose to remove point cloud information with distance < 1cm. When the distance is less than 1cm, there is an occlusion on the LIDAR or a problem with the transmitter module of the LIDAR, and the wrong LIDAR point cloud data needs to be removed, and the PCL library function is used to remove the values with a distance of less than 1cm by the remove Far Point Cloud (laserCloudIn, laser CloudIn,0.01) function to remove the values with a distance of less than 1cm in the LIDAR. The data after pasting the tiny opaque stickers on the laser radar is passed into the PCL library function and the processing results are shown in Figure 4.

```
cloud_src info:
points[]: 49152
width: 49152
height: 1
is_dense: 1
sensor origin (xyz): [0, 0, 0] / orientation (xyzw): [0, 0, 0, 1]
```

Figure 3. PCL library for loading point cloud data

```
after 23354
before 24000
after 23429
before 24000
after 23340
before 24000
after 23453
before 24000
after 23425
```

Figure 4. PCL library for loading point cloud data

Cartographer algorithm is shown in Figure. 5.

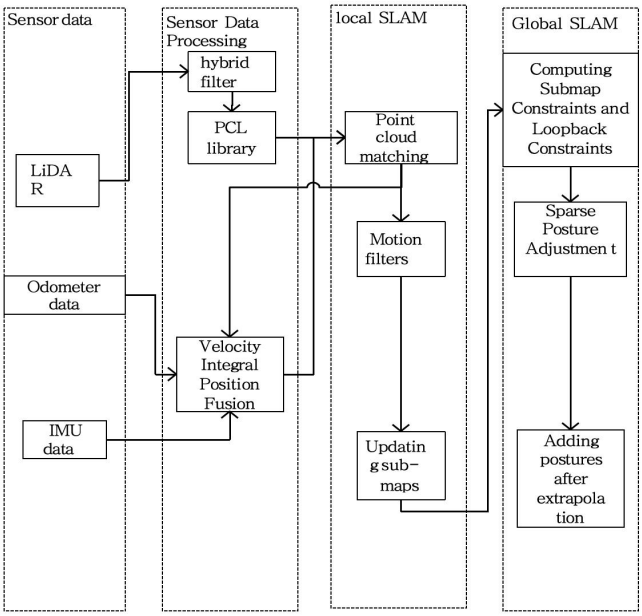


Figure 5. Overall framework of the improved Cartographer's algorithm

4. Simulation and field tests

4.1 Rviz and gazebo simulation

In this chapter, the robot platform is simulated using the Rviz software package under the ROS framework, and the simulation experiment environment is simulated using the gazebo software package under the ROS framework to conduct the simulation comparison experiment. The simulation experiment is carried out in Ubuntu18.04LTS version of linux system, the version of ROS is ROS moledic, in which

the platform for robot simulation is gazebo8.0, firstly, we build the model file of the robot platform, and add the sensor model to complete the conFigureuration. The simulation model of the robot experiment platform is shown in Figure. 6, and the image of gazebo after building the environment is shown in Figure. 7.

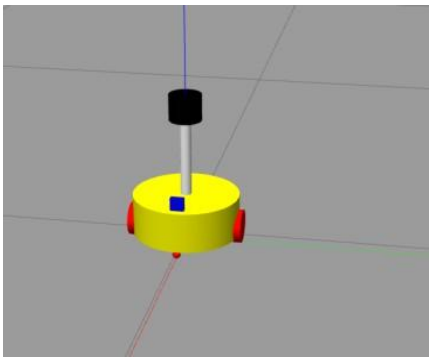


Figure 6. Robot experiment platform simulation model

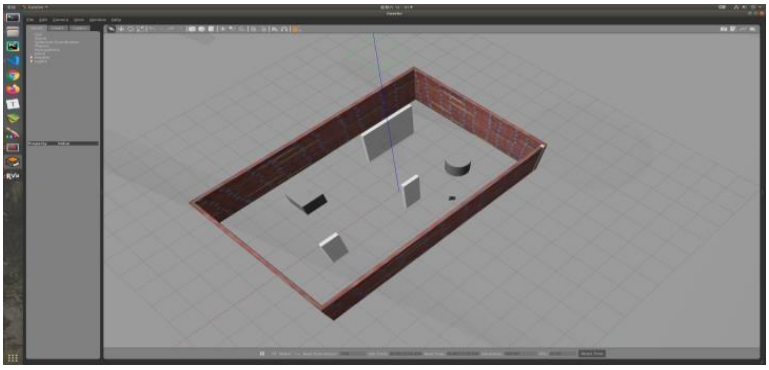


Figure 7. Gazebo build environment

After building the environment the robot is loaded and SLAM simulation experiments are performed. Figure 8 below shows the process of map generation by the robot using the improved Cartographer algorithm. Figure 9 shows the map generated by the improved Cartographer algorithm. The experiment proves that the improved Cartographer algorithm can realize the function of localization and navigation.

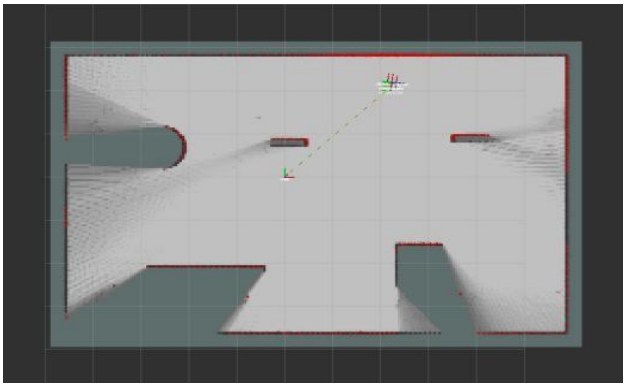


Figure 8. Improvement of Cartographer's Algorithm for Map Generation

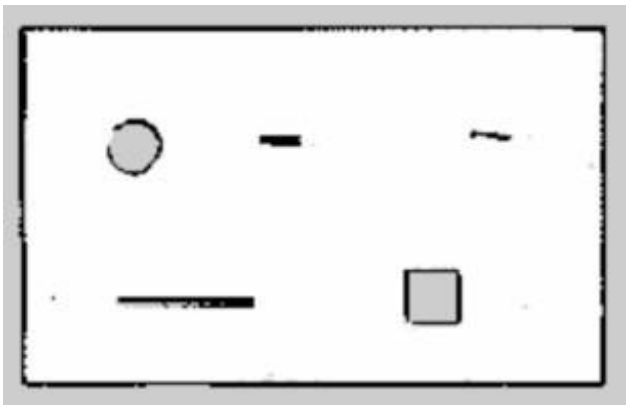
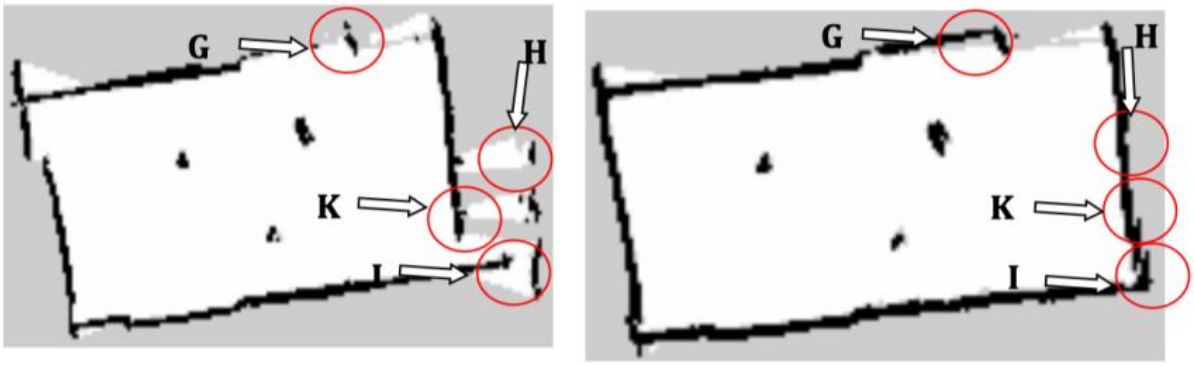


Figure 9. Maps generated after improving Cartographer's algorithm

4.2 Field navigation tests

Figure. a of Figure. 10 is an unimproved Cartographer SLAM algorithm. Figure. b of Figure. 10 shows the graph building effect of the improved Cartographer SLAM algorithm. The robustness of the algorithm is evaluated by comparing the map building effect.



(a)Unimproved Cartographer SLAM algorithm (b)Improved Cartographer SLAM algorithm

Figure 10. SLAM algorithm's map building effect

By comparing the pre- and post-improved Cartographer SLAM algorithm graphs one can notice four distinct differences in the build results. The optimized algorithm is significantly clearer than the pre-optimized algorithm.

(Left) The unimproved Cartographer algorithm fails to remove some outliers, dropouts (points that appear to be nan), and LiDAR data that is too close or too far away. (Right) The improved Cartographer algorithm effectively filters out poor quality LiDAR data.

Figure 10 shows that the red circles are the differences between the Cartographer algorithm before and after the improvement, which are labeled with the letters: G, H, I, and K. It can be seen that there is an obvious laser point escape phenomenon in the circles G, H, I, and K. Due to the fact that the Cartographer algorithm before the improvement is unable to exclude some of the outlying points, dropped points (points that appear to be nan), and points that are too close or far away from the LIDAR data, resulting in the phenomenon of escaping laser line data in (a) Figure. Since the improved

cartographer algorithm excludes some of the outliers, dropped points (points where nan appears) and LIDAR data that are too close or too far away, resulting in the alleviation of the escaping phenomenon in (b). The improved cartographer algorithm has certain superiority.

5. Conclusion

A multi-sensor, multi-mapping algorithm-based SLAM map creation framework for a multi-source integrated robotic navigation system is proposed, which leads to accurate and drift-free mapping across a wide range of complex environments. By fusing LiDAR, odometry, and IMU data through an improved Cartographer algorithm, the system achieves robust localization and precise environmental reconstruction even under conditions of sensor noise, occlusion, and dynamic obstacles. The integration of the Point Cloud Library (PCL) preprocessing and adaptive filtering techniques ensures real-time correction of measurement errors, while the simulation and field test results confirm the method ' s superiority in producing high-resolution, consistent, and stable maps suitable for autonomous navigation and intelligent robotic operations.

References

[1] Smith R, Self M. Estimating uncertain spatial relationships in robotics [J]. Uncertainty in Artificial, 1988, 2: 435-461.

[2] Montemerlo M, Thrun S, Koller D, et al. Fast SLAM: a factored solution to the simultaneous localization and mapping problem [C] // Proc of AAAI National Conference on Artificial Intelligence. Palo Alto, CA: AAAI Press, 2002: 593-598.

[3] Carlone L, Aragues R, Castellanos J A, et al. A fast and accurate approximation for planar pose graph optimization [J]. International Journal of Robotics Research, 2014, 33(7): 965-987.

[4] Weng X W, Li D, Liu J C. Research on 2D laser SLAM based on graph optimization [J]. Automation and Instrumentation, 2019, 34(4): 31-35.

[5] Li M, Wang Y J, Liu X Q. Improvement of Graph-SLAM back end optimization algorithm based on depth-first search branch and bound method [J]. Automation Technology and Application, 2018, 37(9): 8-12.

[6] Li M, Wang Y J, Liu X Q. Improvement of Graph-SLAM back-end optimization algorithm based on depth-first search branch delimitation method [J]. Automation and Instrumentation, 2018, 37(9): 8-12.