Research And Design of A 3D Reconstruction Algorithm with Lio Algorithm

Fangzheng Tian *

School of Automation, Southeast University, Nanjing, 211102, China

* Corresponding Author Email: fangzheng_tian@outlook.com

Abstract. Based on existing research results, this article uses the WheelTev Ackermann-type robot to implement the Fast-Lio and related improved algorithms and initializes system parameters under the condition of tight coupling of laser radar and inertial navigation unit data. At the same time, the obtained point cloud map is subjected to backend loop detection to remove graphic distortion generated during mapping and achieve 3D structure reproduction in multiple semi-open environments. Selected scenes with structural features in the campus of Southeast University were used to test the algorithm's effectiveness, and the robot was able to successfully complete mapping requirements and achieve automatic cruising and return in industrial scenes, demonstrating the algorithm's reliability in industrial scenes. The algorithm was also tested on some open-source datasets, and the experimental results showed improvements in some performance parameters, especially in computational load and robustness compared to existing algorithms.

Keywords: Mobile robot, 3D reconstruction algorithm, Point cloud map, Lio.

1. Introduction

In the past few decades, the development of underground space has been an important topic in society construction. the redevelopment of many tunnels which may have been abandoned for a few decades, faces many difficulties. In order to meet the needs of subsequent engineering planning and the reuse of underground space, these abandoned tunnels need to be measured internally to obtain more accurate three-dimensional structural information. However, for tunnels built a long time ago, their construction drawings, model structures, and other data may have become inaccurate for various reasons, and even lost, therefore, it is a significant engineering problem to obtain the true Three-Dimensional (3D) structures of these tunnels. And the Simultaneous Localization and Mapping (SLAM) algorithm originated in the field of robotics, and its essence lies in achieving both localization and mapping of the external environment through interaction with sensors. Most SLAM algorithms are designed to run offline which means without the Robot Operating System (ROS) system and generate point cloud models that are viewable, thus proving to be effective in reconstruction work. However, the lighting conditions in tunnel environments are poor and the environmental textures are less distinct, which are not suitable for optical sensors such as binocular cameras that can provide depth of field. Instead, they are suitable for inertial sensors and Lidar. Algorithms based on data fusion from these two sensors are generally referred to as Lidar-Inertial Odometry and Mapping (LioDAR), which can provide high-precision positioning and mapping services in special environments such as tunnels.

Besl proposed a method for scan registration based on Iterated Closest Points (ICP), which forms the foundation of data fusion methods for laser odometry [1]. ICP performs well in dense point clouds, which require a large number of environmental features such as obstacles. However, for sparse point clouds, there are few data points suitable for ICP. Zhang combined the ICP method with point-edge distance to develop the framework for laser odometry and mapping, known as the famous LOAM technology [2]. And in the Currently, popular algorithms such as Fast-Lio2 focus on optimizing point cloud calculations by using special data structures to reduce system computation load while improving accuracy [3]. Additionally, the tightly coupled filtering iterative expansion method has also been widely applied and achieved good results [4, 5].

This article focuses on the analysis of existing algorithms for 3D mapping and automated mapping, particularly in semi-open environments. The article proposes improvements based on the existing research, specifically optimizing the point cloud processing using the Incremental voxels (iVox) data structure. The experiments conducted were carried out mostly in semi-open classical environments within Southeast University and some of the classic SLAM open-source data sets. The Southeast University (SEU) campus scenes were extracted from locations such as the Wusi Building and the underground parking lot. Open-source datasets were employed to evaluate the algorithm's modeling abilities, robustness, computational speed, precision, and other performance metrics in more complex and general environments. The results showed that the proposed algorithm has improved performance in some evaluation metrics when compared to other existing algorithms. The data collected from these experiments were used for algorithm improvement, run, and test on a mobile development platform.

In Chapter 2, this article introduced the structure of the experimental platform used and the state estimation method of the algorithm, as well as the map update method based on the iVox structure. Chapter 3 focuses on introducing the experimental results and analyzing them.

2. Methodology of the system

2.1. Framework Overiew

This article is based on the Ackerman Robot, combined with its original functions, and further developed to design data collection and recording functions and automatic exploration functions. With the mobile robot platform being responsible for implementing data recording and automatic exploration functions based on the ROS system. The overall framework of the system is shown in Figure 1.

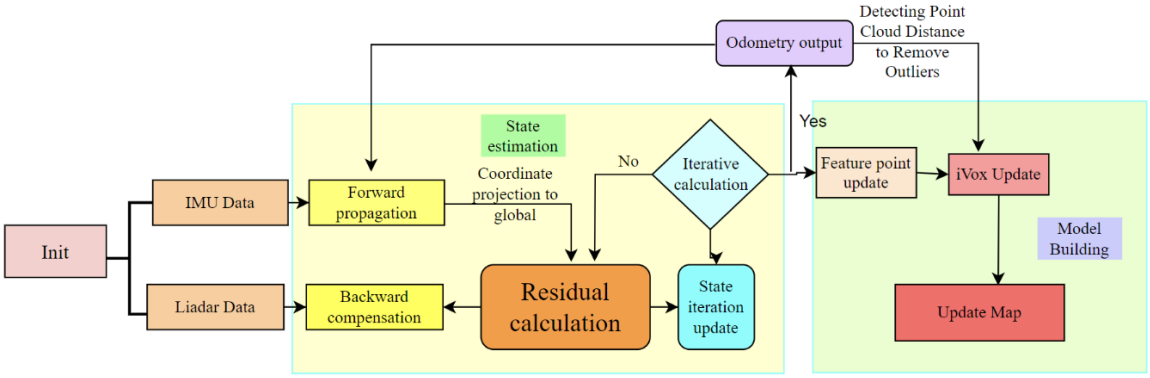


Fig. 1 The over framework of the system (Photo/Picture credit: Original)

While the data processing function is the focus of this design. The mobile platform is responsible for data collection, and the mobile development platform is responsible for data processing. The data collection information mainly involves recording Inertial Measurement Unit (IMU) data and Light Detection and Ranging (LIDAR) data to generate a Rosbag file. This file type can be run offline in various SLAM algorithms to generate a 3D point cloud file that can be viewed [6]. During measurement, three nodes are run on ROS. One node is responsible for sending a read register command to the inertial measurement unit through a Universal Serial Bus (USB) serial port to obtain angular acceleration information, encapsulating the angular velocity information into a topic in ROS, and publishing it. The last node subscribes to the topics published by the first two nodes and saves them. The node data subscription relationships during measurement are shown in Figure 2.

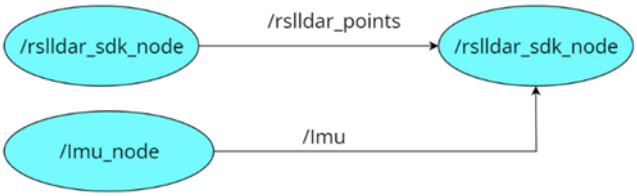


Fig. 2 ROS Node Graph (Photo/Picture credit: Original)

In order to enable robots to automatically navigate and complete tasks in work scenarios, this paper proposes an automatic exploration system based on the combination of Gmapping algorithm and RRT_exploration algorithm. The SLAM algorithm focuses on solving the two sub-problems of mapping and localization, and then constructs the map through the robot's known trajectory and observer data.

During this process, the Gmapping algorithm combines the data from laser radar scans and IMU and processes them using the Rao-Blackwellized Particle Filters (RBPF) particle filtering algorithm to obtain the proposal distribution function. It also uses the maximum likelihood estimation and keyframe determination methods to reduce invalid data, lower computational load, and reduce the number of resampling times, ensuring that resampling only occurs when the particle dispersion is less than the threshold. Finally, the new grid map is estimated based on the calculated robot trajectory [7].

Rapid-exploration Random Tree (RRT_exploration) is a search algorithm based on the RTT path planning algorithm [8]. This algorithm tends to prioritize exploring unexplored areas, making it suitable for map construction in unknown spaces in this project. In the entire system, this algorithm is used for boundary point detection. By establishing multiple independent fast exploring random trees, the global boundary and local boundary are detected separately. After filtering, suitable boundaries are selected for exploration and passed to the robot task allocation module to guide the robot's actions [9].

2.2. State Estimate

The Lio system utilizes the same state estimation method as the Fast-Lio system, involving both forward and backward propagation, and introduces a new point cloud storage structure, iVox, used in the Faster-Lio system, to improve computational speed and reduce the computational load of mobile robot systems, ultimately achieving better experimental results [10]. The state estimation process is divided into forward and backward propagation, and in order to adapt the theoretical continuous model to real-world conditions, a zero-order hold is introduced to maintain signal constancy between two consecutive sampling instants over the IMU sampling period Δt, defining the operator $\oplus$, and representing the relationship between the two states of the corresponding discrete system with the following equation 1.

$$\mathbf{x}_{i+1} = \mathbf{x}_i \oplus (\Delta t \mathbf{f}(\mathbf{x}_i, \mathbf{u}_i, \mathbf{w}_i)) \tag{1}$$

In the equation, i represents the temporal index corresponding to the IMU measurement value, and the definitions of the other variables are as follows, so $\mathbf{x}_{i+1}$ represents the state at time $i+1$, In translation, where $\mathbf{x}_i$ represents the state of the corresponding time series, u represents the input, and w represents the state, the estimated covariance matrix can be calculated in the tangent space, thus achieving the state estimation of equation using an extended Kalman filter in an iterative scenario. The forward propagation process is used to handle data generated by an IMU and continuously update and calculate the predicted equation of the state using the ESKF algorithm. Essentially, the ESKF algorithm is a special form of the Kalman filter algorithm which can estimate system states. Its characteristic is the reduction of non-linear errors through multiple iterations. In the algorithm, the state includes nominal state, state error, and system estimated state, which satisfy the following relationship:

$$\boldsymbol{X}_t = \hat{\boldsymbol{X}}_t \boxplus \delta \boldsymbol{X}_t \tag{2}$$

The left side of the equation represents the nominal state, and the $\boxplus$ operation on the right side indicates that all quantities in the state equation satisfy the additive law, while the positional operation follows the corresponding situation in the Lie algebra, that is, $\boldsymbol{\delta R} = \boldsymbol{Exp}(\delta \boldsymbol{\theta})$. The estimation error and system error conform to the multiplication law. The specific corresponding relationships of each quantity are as follows: the robot's position, velocity, pose, angular velocity measured by the IMU, angular acceleration, and gravitational constant, corresponding to the state $x_t = [\boldsymbol{p}_t, \boldsymbol{v}_t, \boldsymbol{R}_t, \boldsymbol{b}_{at}, \boldsymbol{b}_{\omega t}, \boldsymbol{g}_t]^{\mathrm{T}}$:

$$\begin{aligned} p &= \hat{p} + \delta p \\ v &= \hat{v} + \delta v \\ \boldsymbol{R} &= \boldsymbol{R}\delta\boldsymbol{R} \\ b_g &= \hat{b}_g + \delta b_g \\ b_a &= \hat{b}_a + \delta b_a \\ g &= \hat{g} + \delta g \end{aligned} \tag{3}$$

In the forward propagation process, the change process of the state process quantity x̂ satisfies the state update error, and the expression for the next time state error based on the previous time state error and noise can be calculated in the system.

$$\delta \boldsymbol{X}_{\boldsymbol{t+1}} = f(\boldsymbol{x}, \boldsymbol{\delta x}, \boldsymbol{\mu}_m, \boldsymbol{i}) = \boldsymbol{F}_x(\boldsymbol{x}, \boldsymbol{\mu}_m) * \delta \boldsymbol{X}_{\boldsymbol{t}} + \boldsymbol{F}_i * \boldsymbol{i} \tag{4}$$

The $\boldsymbol{F}_x$ and $\boldsymbol{F}_i$ are the partial derivatives of the state error function $f(\boldsymbol{x}, \boldsymbol{\delta x}, \boldsymbol{\mu}_m, \boldsymbol{i})$ with respect to the noise term $\boldsymbol{i}$ and the state error term $\boldsymbol{\delta x}$, respectively, and their forms are as follows:

$$F_i = \frac{\partial f}{\partial \mathbf{i}}\Big|_{\mathbf{x}, \mathbf{u}_m} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \mathbf{I} & 0 & 0 & 0 \\ 0 & \mathbf{I} & 0 & 0 \\ 0 & 0 & \mathbf{I} & 0 \\ 0 & 0 & 0 & \mathbf{I} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{F}_{\mathbf{x}} = \frac{\partial f}{\partial \delta \mathbf{x}}\Big|_{\mathbf{x}, \mathbf{u}_m} = \begin{bmatrix} \mathbf{I} & \mathbf{I}\Delta t & 0 & 0 & 0 & \\ 0 & \mathbf{I} & -\mathbf{R}[\mathbf{a}_m - \mathbf{a}_b]_{\times} \Delta t & -\mathbf{R}\Delta t & 0 & \mathbf{I}\Delta t \\ 0 & 0 & \mathbf{R}^{\top}\{(\boldsymbol{\omega}_m - \boldsymbol{\omega}_b)\Delta t\} & 0 & -\mathbf{I}\Delta t & 0 \\ 0 & 0 & 0 & \mathbf{I} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{I} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5}$$

The forward propagation process of the system can be completed by continuously occurring within a scanning time, until the end of the scan, the obtained propagation state and covariance can be used to correct the relationship between the actual situation and the propagated state, because the covariance at this time is the covariance of the error between the two [11].

The purpose of the backward propagation is to remove the motion distortion generated by the system. Even after time offset correction, it cannot guarantee that the timestamps of the radar and IMU are completely consistent. In fact, the result of the initialization time offset correction is obtained through interpolation calculation, which inevitably has a small error compared to the actual situation. This kind of error is not obvious in a few cycles, and can be accepted by the system, but after a period of accumulation, it is easy to reduce the system robustness and produce motion distortion. This kind of distortion caused by the mismatch of the reference system of the subject is a common problem in SLAM systems. In this paper, the backward propagation method is adopted to push forward from a

certain moment and correct the point cloud state of the previous moment, so as to obtain a point cloud with high confidence. The expression of the point cloud in the world coordinate system corresponding to Lidar after estimation and compensation in the backward propagation is shown below.

3. iVox and Map update

The full name of iVox is Incremental Sparse Voxels [12]. In most SLAM solutions, it is necessary to continuously compute and maintain a local map based on the surrounding point cloud information near the current position point. This local map is generally used to provide services for registering point clouds in scans [9]. One of the key points in various algorithm designs is how to control the appropriate number of point clouds in the map. If the number of point clouds is too large, it will increase the computational load of the onboard computer and reduce the processing speed, resulting in a decrease in the effective frame rate of the system. If the number of point clouds is too small, it will lack the necessary information for building the map. Therefore, the local map should be as sparse as possible, which means that it should not be too dense in some parts and should be distributed reasonably in various positions, in order to maximize the use of information in the point cloud and maximize the efficiency of point cloud data storage.

Therefore, using a special spatial data structure to maintain the local map is a very effective approach. For example, in the Fast-Lio2 algorithm, the ikd-Tree, which is the data structure of the octree, is used to maintain the grid map. Compared with other similar algorithms, Fast-Lio effectively reduces the computational load of the algorithm. However, the octree map voxelizes all the maps around the robot, but in semi-open environments, most of the environments around the robot are undoubtedly air, and in the octree map, an empty voxel is required to maintain this part of the map, which is completely meaningless and wasteful of resources [13].

A sparse voxel map only creates and maintains data where it is needed, that is, where the point cloud is determined. However, in this case, the voxel cannot be represented linearly in the program using simple data structures such as linked lists or arrays, because the creation and distribution of voxels are uncertain, and there is no continuous space to arrange these voxels.

However, in the development of computer data structures, there is no shortage of data structures with nonlinear characteristics specifically designed to deal with this problem. The hash table structure can uniquely determine the index based on the spatial position of the voxel by inputting the three-dimensional coordinates as parameters into the hash function and placing the pointer value of the corresponding voxel at the index address.

Hash table, also known as a hash map, is a relatively abstract data structure that is implemented through a combination of a hash function and a underlying data structure. The underlying data structure is usually an array, and the data stored in the table is accessed by mapping specific key values to specific positions. This structure can improve the efficiency of data retrieval.

After the hash function calculates the array address corresponding to a certain key value, the data corresponding to the key value is stored in the array, achieving a dimensionality reduction of the data storage format. Whether it is adding, searching, or retrieving data, the data address can be indexed by the hash function and ensure a one-to-one correspondence between the two. Std:unordered_map in the C++ standard library is a typical data container for hash table structure. In the context of SLAM, the key value is obtained from the position of the point cloud, i.e., the key value of the point cloud comes from the spatial coordinates of the voxel. As long as the three-dimensional coordinates of the voxel are multiplied by a large integer and then subjected to a bitwise operation, the one-to-one correspondence between the key value and the hash array can be ensured. In the restoration process, the voxel can be calculated by dividing the coordinates of the point p by the edge length of the voxel and rounding down. This is the spatial hash function, and iVox adopts this design, which is expressed in the following formula.

$$\mathbf{p} = [p_x, p_y, p_z]^T, \mathbf{v} = \frac{1}{s}[p_x, p_y, p_z]^T \tag{6}$$

$$id_v = \text{hash}\,(v) = (v_x n_x)\ \text{xor}\ (v_y n_y)\ \text{xor}\ (v_z n_z) mod N \tag{7}$$

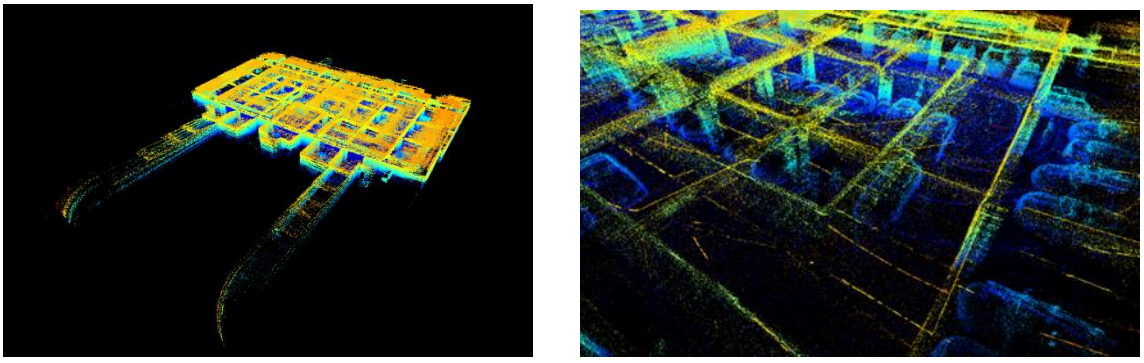
The problem of hash collision is common in the design process of hash tables, where in special cases the mapping between key values and the corresponding address in the hash array cannot be achieved one-to-one. However, in iVox, the algorithm maintains a limited number of voxels, and the hash table is used to measure the proportion of data size to array capacity, the load factor is always within an acceptable range. Additionally, the space hash function is designed to have anti-collision ability, thus iVox has strong anti-collision ability. Even if collisions occur, they often occur in a distant voxel with a small number, which will not have a significant impact on the system's robustness and can be ignored.

Compared with other data structures, iVox's incremental map update is simpler because after the voxel calculation is completed, the point cloud data can be directly added to the corresponding voxel grid. If it is in PHC format, the curve position corresponding to PHC can be added [14]. The entire process only needs to give the linear position of the point cloud through the spatial hash function.

In iVox, traversing the entire iVox local map is also slow, so similar to algorithms such as Fast-Lio2, it is necessary to update and process historical point clouds during the calculation process. However, this kind of update in iVox is passive, rather than actively deleting after each frame calculation. Because the LRU cache of the local map is fixed, the structure of iVox can record the voxels that have been recently used during the nearest neighbor search, and the point clouds that are far away from the query on the spatial distance will naturally be moved to the end. In this way, in the process of the local map autonomously releasing LRU, the point clouds that are not queried are automatically released, effectively increasing the calculation speed of the algorithm. At the same time, the deletion is for the entire voxel, and even the internal points will be deleted. This local map caching strategy will also make the local map follow the movement of the vehicle, greatly reducing the actual required calculation amount.

4. Experiment Result and Analysis

This article selects internal campus scenes of Southeast University for 3D modeling effects and conducts scene restoration and analysis after modeling. The paper demonstrates the modeling effect of the underground parking lot in Figure 3.



(a) Overall rendering (b) Partial rendering

Fig. 3 modeling effect of the underground parking lot (Photo/Picture credit: Original)

As shown in the figure, the mapping effect of data in the underground parking lot on the mobile development platform is good overall. The internal structure of the underground parking lot and the parking of cars are clearly visible, and the point, line, and surface structures of the underground space are accurately identified. There is no obvious motion distortion, and under long-distance detection, there is no obvious odometer drift, indicating that the system has good robustness and verifies its effectiveness in real-life scenarios.

The algorithm records data as shown in Figure, and even with relatively outdated Central Processing Unit (CPUs) mounted on the mobile robot platform, it can still achieve a frame rate of

nearly 150Hz. This indicates that the system has a fast response speed, low CPU performance requirements, and is suitable for use on mobile robot platforms. At the same time, the time consumption of each step in the overall process of the algorithm was recorded, and it can be seen that the time consumption of each addition of the iVox point cloud is very small in the overall time consumption of the system. This demonstrates the effectiveness of this data structure in algorithm optimization.

In addition, the experiment also selected the Wusi Building with more diverse 3D structures and more interference for modeling, as shown in the figure 4.

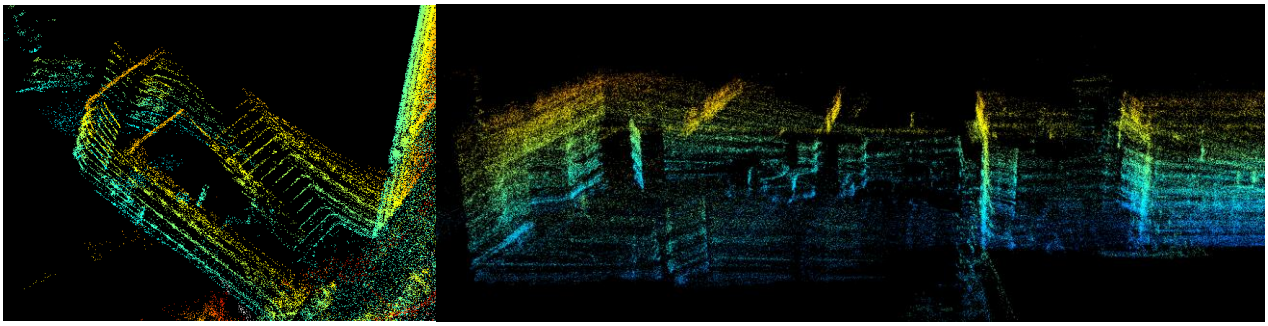


Fig. 4 Modeling effect of Building 54 (Photo/Picture credit: Original)

In order to test the response and accuracy effects of the algorithm compared to existing common algorithms, this paper uses multiple common open-source algorithms on some open-source datasets on a mobile testing platform and obtains the following two tables for performance analysis of the algorithms.

Table 1. Comparison of algorithm running speeds

Map ID	this paper (ms)	Faster-Lio(ms)	Fast-lio2(ms)	LiLi-OM [15](ms)
Nclt_2	6.85	8.26	16.33	22.43
Nclt_4	9.03	10.58	16.89	22.39
Ulhk_1	6.19	4.06	14.55	14.54
Ulhk_2	5.22	4.98	14.32	15.03
Liosam_1	8.16	7.18	12.71	-----

As shown in Table 1, the time consumed for each pose calculation during algorithm execution was tested and obtained by adding code to measure timestamps in the algorithm. From the experimental results, the algorithm proposed in this paper using the iVox structure and Faster-Lio have a significant advantage over other algorithms in terms of pose calculation speed. However, the PHC structure used in this paper's algorithm is only superior to the Faster-Lio algorithm in some cases, which is consistent with the theoretical analysis that PHC retrieval is more advantageous than linear structures only when the point cloud is large, and the index is complex. This provides guidance for selecting the optimal application scenario for the algorithm.

Table 2. Comparison of accuracy among different algorithms

Map ID	this paper (%)	Faster-Lio(%)	Fast-lio2(%)	LIO-SAM(%)
Nclt_2	0.34	0.36	0.36	0.43
Nclt_4	0.33	0.35	0.35	0.37
Ulhk_1	1.50	1.55	1.48	1.87
Ulhk_2	1.69	1.68	1.62	1.46

As shown in Table 1, the translation into English shows the percentage error produced by different algorithms after running for every hundred meters in a specific dataset. As this study's algorithm and Fast-Lio and Faster-Lio algorithms do not have a loop closure detection process, the loop closure detection in Lio-SAM was manually turned off. The conclusion drawn from this study indicates that

there is no significant decrease in measurement accuracy with a noticeable reduction in computational load. This suggests that the algorithm has a significant advantage over most current open-source algorithms. However, if further improvements in accuracy are desired, it is necessary to consider adding a loop closure detection process.\

5. Conclusion

This article is based on existing open-source algorithms to analyze and design a set of autonomous patrol navigation and 3D scanning and reconstruction algorithm systems for semi-open indoor and outdoor spaces. The system is based on Gmapping & RRT_exploration for automatic mapping and navigation while ensuring map traversal as much as possible. On this basis, a forward and backward propagation process coupling IMU and Lidar data was designed for point cloud state estimation, and iVox data type was introduced into the map iteration process. Space hash functions and PHC methods were used for nearest neighbor querying, realizing the design and theoretical analysis of the entire SLAM system.

This article also sets up three experimental groups to verify the feasibility and robustness of the system in practical scenarios. It designs accuracy and operation speed comparison groups to compare the system with currently popular open-source algorithms. The results confirm that the system as a whole has good feasibility and some performance improvements compared to mainstream algorithms such as Fast-Lio2 and Lio-SAM.

The portability of this work is good and has achieved results that are in line with expectations in functional verification experiments. In subsequent experiments, the robot's environmental adaptability can be further improved according to different environmental scenarios, and backend loop detection and other steps can be added to improve its operational accuracy. The work is easily portable and prepares for the effective development of future work.

References

[1] Ke Sun, Kartik Mohta, Bernd Pfrommer, et al. Robust stereo visual inertial odometry for fast autonomous flight. IEEE Robotics and Automation Letters, 2018, 3(2): 965-972.

[2] Dingkang Wang, Connor Watkins, Huikai Xie. MEMS mirrors for LiDAR: A review. Micromachines, 2020, 11(5): 456.

[3] Zheng Liu, Fu Zhang, Xiaoping Hong. Low-cost retina-like robotic lidars based on incommensurable scanning. IEEE/ASME Transactions on Mechatronics, 2021, 27(1): 58-68.

[4] P.J. Besl, Neil D. McKay.Method for registration of 3-D shapes. Sensor fusion IV: control paradigms and data structures. Spie, 1992, 1611: 586-606.

[5] Ji Zhang, Sanjiv Singh. LOAM: Lidar odometry and mapping in real-time. Robotics: Science and Systems. 2014, 2(9): 1-9.

[6] Tixiao Shan, Brendan Englot. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018: 4758-4765.

[7] Yashar Balazadegan Sarvrood, Siavash Hosseinyalamdary, Yang Gao. Visual-LiDAR odometry aided by reduced IMU. ISPRS international journal of geo-information, 2016, 5(1): 3.

[8] Xingxing Zuo, Patrick Geneva, Woosik Lee, Yong Liu, Guoquan Huang et al. Lic-fusion: Lidar-inertial-camera odometry.2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2019: 5848-5854.

[9] Wu Xu, Fu Zhang. Fast-lio: A fast, robust lidar-inertial odometry package by tightly coupled iterated kalman filter. IEEE Robotics and Automation Letters, 2021, 6(2): 3317-3324.

[10] Wei Xu, Yixi Cai, Dongjiao He, Jiarong Lin, Fu Zhang et al. Fast-lio2: Fast direct lidar-inertial odometry. IEEE Transactions on Robotics, 2022, 38(4): 2053-2073.

[11] Giorgio Grisetti, Cyrill Stachniss, Wolfram Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. IEEE transactions on Robotics, 2007, 23(1): 34-46.

[12] Chunge Bai, Tao Xiao, Yajie Chen, Haoqian Wang, Fang Zhang, Xiang Gao. Faster-LIO: Lightweight tightly coupled LiDAR-inertial odometry using parallel sparse incremental voxels. IEEE Robotics and Automation Letters, 2022, 7(2): 4861-4868.

[13] Hue-Ling Chen, Ye-In Chang. All-nearest neighbors finding based on the Hilbert curve. Expert Systems with Applications, 2011, 38(6): 7462-7475.

[14] Kailai Li; Meng Li; Uwe D. Hanebeck. Towards high-performance solid-state-lidar-inertial odometry and mapping. IEEE Robotics and Automation Letters, 2021, 6(3): 5167-5174.

[15] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, Daniela Rus. Lio-sam: Tightly coupled lidar inertial odometry via smoothing and mapping. 2020 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, 2020: 5135-5142.