Design and Implementation of ROS-Based Autonomous Mobile Robot Positioning and Navigation System

ZHU Jian-jun
College of Information & Control Engineering
Jilin Institute of Chemical Technology
Jilin, China
e-mail: zjj099@163.com

Li Xu
College of Information & Control Engineering
Jilin Institute of Chemical Technology
Jilin, China
e-mail: 929415785@qq.com

***Abstract*—Aiming at the problem of positioning and navigation of mobile robots indoors, this paper designs and implements related functions based on Robot Operating System and laser radar. The core controller of the robot adopts an industrial computer, and is equipped with a laser radar to acquire information about the surrounding environment and develop related algorithms on the Robot Operating System. Realize Simultaneous Localization and Mapping function based on Cartographer algorithm, path planning based on optimal path algorithm and navigation function based on particle filter algorithm. The real machine experiment proves that the system can efficiently perform real-time synchronous positioning and map construction and navigation functions.**

Keywords-robot operating system;simultaneous localization and mapping;lidar;indoor position and navigation

I. INTRODUCTION

Simultaneous Localization and Mapping (SLAM) technology is mainly used for robots to construct and locate maps in unfamiliar environments. That is to say, the mobile robot uses its own sensor to obtain environmental information in an unfamiliar environment, realizes its own positioning and incrementally constructs an environmental map[1].

II. SIMULTANEOUS LOCALIZATION AND MAPPING

SLAM uses two mathematical models, the motion equation and the observation equation.Turn the time of continuous motion into discrete moments t=1,…,k,the position of the robot at each moment is represented by x，recorded as $x_1,\ldots,x_k$，This series of points constitutes the route for the robot to move. The equation of motion is expressed by the function f as:

$$x_k = f(x_{k-1}, u_k, w_k) \tag{1}$$

Among them u_k is the reading of the motion sensor,w_k is noise.

The observation equation describes that when the robot sees a certain target point y_j in position x_k, an observation data $z_{k,j}$ is generated. The observation equation is expressed as a function h:

$$z_{k,j} = h(y_j, x_k, v_{k,j}) \tag{2}$$

Where v_{kj} is the noise in this observation.

III. SYSTEM DESIGN BASED ON ROS AND LIDAR

The physical object of this design is shown in Figure 1. The main hardware modules are: Intel i3 industrial control host, Stm32 drive board, TB6612FNG motor drive module, RPLIDAR A2 laser radar and motor. The hardware architecture of each module is shown in Figure 2:

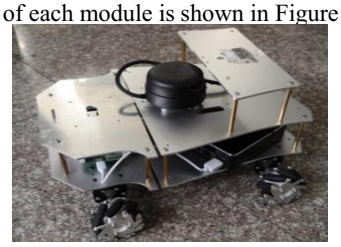

Figure 1. Autonomous mobile robot

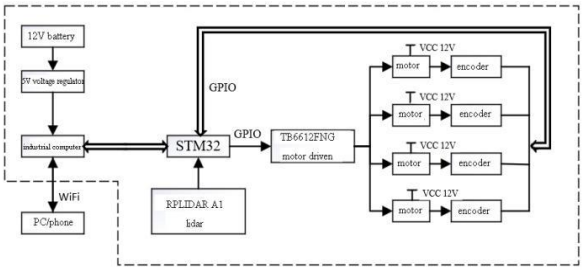

Figure 2. Autonomous mobile robot hardware architecture

A. Hardware Core Module

1) Industrial computer

The autonomous mobile robot is controlled by the industrial computer core. It is a microcomputer designed and produced by Soarsea, equipped with Intel I3 dual-core processor and running Linux operating system. Install the Ubuntu 16.04 operating system and use ROS to write the minimum system control robot.

2) Lidar

A laser radar is a sensor that performs distance determination without contact. When the laser beam is reflected back onto the target, the reflected beam is processed to obtain an environmental image. Traditional laser sensor systems include: a distance sensor that measures a feature point, a two-dimensional scanning plane of lidar, and a three-dimensional lidar [2, 3]. This design adopts the two-dimensional laser radar RPLIDAR A2 developed by Siwei Technology Co., Ltd., which adopts laser triangulation technology, equipped with RPVision 2.0 high-speed visual ranging engine, and rotates clockwise to 8000 times per second within a radius of 18m in the surrounding environment 360 degree detection to obtain an environmental image.

2473-3636/19/$31.00 ©2019 IEEE
DOI 10.1109/DCABES48411.2019.00060

B. Software Core Module

1) Robot operating system

Robot Operating System (ROS) is an emerging robotic operating system that uses a distributed architecture and is a robot service framework for Linux environments[4]. The process is reflected in peer-to-peer. It mainly includes: nodes, messages, topics, and services. A node is an executable file. A message is a type of ROS data that is used to subscribe or post to a topic [5]. ROS has a number of feature packs that enable common functions, allowing people to focus on researching and improving basic algorithms when designing robots, greatly improving the efficiency of robot development[6-8].

2) System software design

ROS has a number of proven feature packages that can be used directly, including the Cartographer software package for robotic positioning and mapping, navigation kits for autonomous navigation and obstacle avoidance. Among them, Cartographer is an open source SLAM library containing ROS interface developed by Google in recent years. The main sensor relies on lidar and inertial navigation (IMU) for 2D and 3D internal SLAM. Its ROS framework is shown in Figure 3:

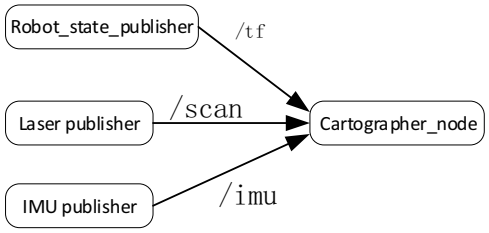

Figure 3. Cartographer's ROS framework

Among them, Cartographer_node is the main node, where /scan is the data obtained by lidar scanning, and /imu is the data released by IMU.

C. SLAM Algorithm

In 2D SLAM, Cartographer divides two-dimensional map drawing into two parts, local and global. The pose ξ of the robot consists of three parameters, translation (x,y) and rotation ξ_θ, obtained by laser scanning, and is expressed as $\xi=(\xi_x, \xi_y, \xi_\theta)$.

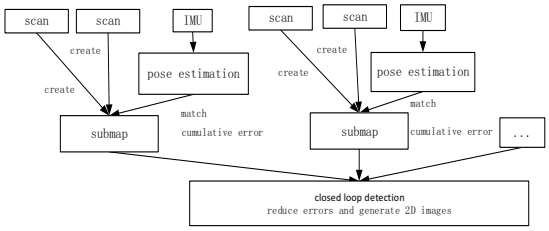

Figure 4. Cartographer algorithm flow

As shown in Figure 4, scan is the laser data in the current pose when the map is created. The scan set forms a submap called submap, which is also the basic unit of closed loop detection. When new laser information is inserted into the map, the closed loop is found if the estimated pose of the laser scan is close to the pose of the laser scan of the submap in the map.

1) Scan matching

Before the scans are inserted into the submap, the Ceres-based scan matcher method is needed to optimize the scanning pose. Convert it to a nonlinear least squares problem, as shown in Equation 2.1.

$$\arg\min_{\xi} \sum_{k=1}^{K} (1 - M_{smooth}(T_\xi h_k))^2 \tag{2.1}$$

In the above formula, T_ξ is a conversion coefficient of the laser information h_k from the laser coordinate system to the submap coordinate system. Function $M_{smooth}: R^2 \rightarrow R$ is a smoothing of the probability values assigned in the submap using bicubic interpolation.

2) Loop closing

a) Optimization

Closed loop optimization is represented as a nonlinear least squares problem as well as scan matching. As shown in Equation 2.2:

$$\arg\min_{\Xi^m, \Xi^s} \frac{1}{2} \sum_{ij} \rho(E^2(\xi_i^m, \xi_j^s; \Sigma_{ij}, \xi_{ij})) \tag{2.2}$$

Among them, $\Xi^m = \{\xi_i^m\}_{i=1,\ldots,m}$ and $\Xi^s = \{\xi_j^s\}_{j=1,\ldots,m}$ are the poses of submap and scan in the world coordinate system, respectively, and are optimized according to the given constraints. These constraints take the form of a relative pose ξ_{ij} and an associated covariance matrix $\sum_{ij}$. For a pair of submap E and scan F, pose ξ_{ij} describes the location of the scan match in the submap coordinate system. The covariance matrix $\sum_{ij}$ can be evaluated using the Cedes covariance. The corresponding residual E calculation formula is shown in 2.3 and 2.4:

$$E^2(\xi_i^m, \xi_j^s; \Sigma_{ij}, \xi_{ij}) = e(\xi_i^m, \xi_j^s; \xi_{ij})^T \Sigma_{ij}^{-1} e(\xi_i^m, \xi_j^s; \xi_{ij}) \tag{2.3}$$

$$e(\xi_i^m, \xi_j^s; \xi_{ij}) = \xi_{ij} - \begin{pmatrix} R_{\xi_i^m}^{-1}(t_{\xi_i^m} - t_{\xi_j^s}) \\ \xi_{i;\theta}^m - \xi_{j;\theta}^s \end{pmatrix} \tag{2.4}$$

The purpose of using the loss function is to reduce the impact of outliers added to the optimization problem on the system.

b) Branch-and-bound scan matching(BBS)

The Cartographer algorithm uses another method called Branch-and-bound scan matching for closed-loop optimization.

$$\xi^* = \arg\max_{\xi \in w} \sum_{k=1}^{K} M_{nearest}(T_\xi h_k) \tag{2.5}$$

Where w represents the size of a search window, and $M_{nearest}$ is the extension of the nearest grid point of the parameter in M_{smooth} to the corresponding pixel set R^2. Choosing the right step size can improve the calculation accuracy and efficiency. If the maximum scan range d_{max} does not exceed a single pixel δ_θ, then the angular step

length F can be obtained according to the Pythagorean theorem:

$$d_{\max} = \max_{k=1,..,K} \|h_k\| \tag{2.6}$$

$$\delta_\theta = \arccos(1 - \frac{r^2}{2d_{\max}^2}) \tag{2.7}$$

Calculate the integer step size based on the size of the search line window and the corner window:

$$w_x = \left\lceil \frac{W_x}{r} \right\rceil, w_y = \left\lceil \frac{W_y}{r} \right\rceil, w_\theta = \left\lceil \frac{W_\theta}{\delta_\theta} \right\rceil \tag{2.8}$$

$$\overline{w} = \{-w_x, \ldots, w_x\} \times \{-w_y, \ldots, w_y\} \times \{-w_\theta, \ldots, w_\theta\} \tag{2.9}$$

$$w = \{\xi_0 + (rj_x, rj_y, \delta_\theta j_\theta) : (j_x, j_y, j_\theta) \in \overline{w}\} \tag{2.10}$$

The relationship between scan and submap in loopback detection and loopback optimization is shown in Figure 5.

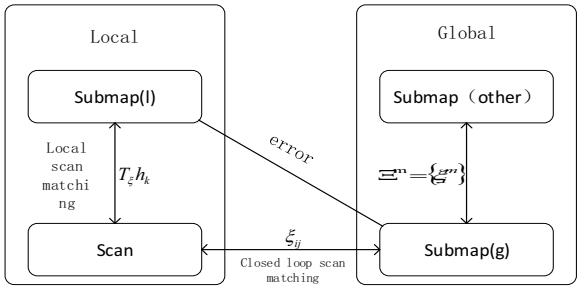

Figure 5. Scan and submap relationship

D. Navigation System Function Realization

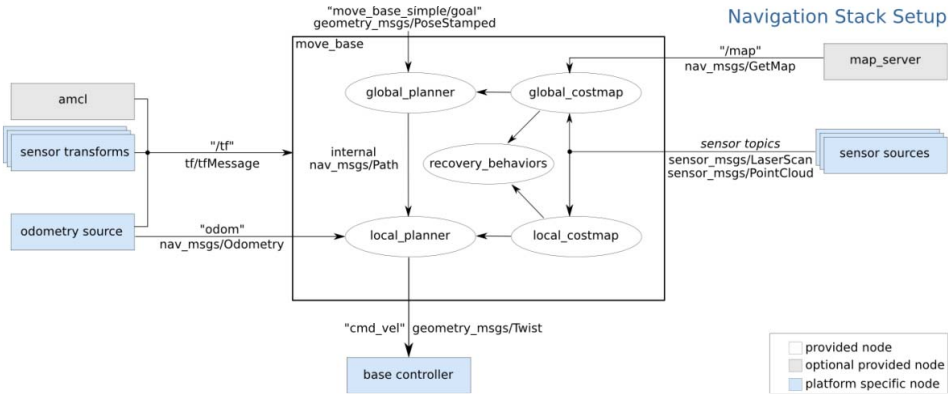

Figure 6. Overall navigation package framework

In Figure 6, sensor transforms 12 converts the sensor data into coordinate information on the control center; sensor sources here are robot navigation sensor data inputs, generally only laser sensor data and point cloud data; odometry source publishes odometer The base controller is responsible for encapsulating the previous data into specific line speed and angular velocity information and distributing it to the hardware platform; the map_server node is used to read the data information of the map. The recovery_behaviors node allows the robot to recover after a navigation failure. Four configuration files are required before the move_base node is run. These files define a set of related parameters, including sensor reading range, obstacle information threshold, robot radius, map update frequency, and speed and acceleration of robot movement. The four configuration files are:

- Basic local planner configuration/base_local_planner_params.yam
- Common cost map configuration/costmap_common_params.yaml
- Global cost map configuration/global_costmap_params.yaml
- Locall cost map configuration/local_costmap_params.yaml

IV. EXPERIMENTAL RESULTS AND ANALYSIS

Experimental steps:①start bringup.launch to start the industrial computer;②start rplidar.launch wake up radar; ③start cartographer_slam's launch file; and start rviz;④start keyboard control to control car movement: rosrunteleop_twist_keyboardteleop_twist_keyboard.py;⑤ Control the robot to move the map in the area;⑥Click the 2D Nav Goal icon and select the target point, the robot will plan a feasible path and avoid obstacles based on the detected obstacle information.

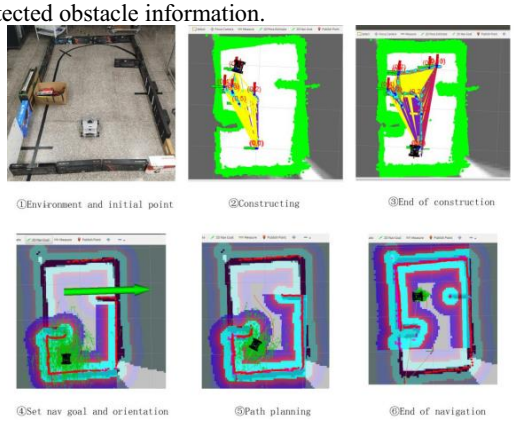

Figure 7. Experimental result

According to the experimental results, the system has a good mapping effect, a clear outline, and can reach the set target point without collision.

V. CONCLUSION

This design uses the industrial computer as the core controller, equipped with RPLIDAR A2 laser radar to obtain environmental information, supplemented by ROS development related algorithms. The theoretical principle and algorithm flow of the Cartographer algorithm are studied and analyzed. The Cartographer algorithm references the concept of a submap, which enables front-end matching to match the current frame to the currently created submap. Loopback detection is used, and branch and bound are used to optimize search, improve efficiency, and achieve real-time loopback. The real machine measurement shows that the system functions are well realized, and the real-time positioning, mapping and navigation functions can be efficiently performed.

REFERENCES

[1] Yuan Quande. Vision-based multi-robot collaborative SLAM research[D]. Harbin:Harbin Institute of Technology, 2016.

[2] Khan S, Wollherr D, Buss M.,"Modeling laser intensities for simultaneous localization and mapping," IEEE Robotics and Automation Letters, 2016, pp.692-699.

[3] Artamonov S, Gryaznoy N, Kupenyuk V, et al. "Selection of scanners for use in lidar systems," Journal of Optical Technology, 2016, pp.549-555.

[4] Huang Kaihong, Yang Xingrui , ZengZhiwen. Construction of ROS Outdoor Mobile Robot Software System J. Robotics and applications, 2013(4):37-41.

[5] Zhang Kaiwei. Open source robot operating system[M]. Beijing:Science Press, 2012.

[6] Fairchild C, Harman T L., "ROS Robotics By Example,"Packt Publishing, 2016.

[7] AaronMartinez, EnriqueFernANdez, "ROS robot programming," Beijing:Mechanical Industry Press, 2014.

[8] Carol Fairchild, "ROS robotics by example:learning to control wheeled, limbed, and flying robots using ROS Kinetic Kame". 2017.