Academic Journal of Science and Technology
ISSN: 2771-3032 | Vol. 2, No. 2, 2022

Test of Cartographer based on Raspberry PI

Fan Zhang

Department of Automation, North China University of Technology, Beijing, China

Abstract: The Raspberry Pi-based AI car uses a 4-wheeled Ackerman wheeled robot as a motion platform and is equipped with a high-performance lidar. In the ROS system under Ubuntu18, the test has passed the cartographer mapping algorithm. After experiments, it is found that the cartographer algorithm can also present a good and superior mapping effect under the Raspberry Pi platform.

Keywords: Cartographer, SLAM.

1. Introduction

Based on a 4-wheeled Ackerman robot carrying Raspberry PI, this project successfully tested and tested the feasibility of Cartographer's graph construction algorithm on its platform through the construction of dependency, environment and other necessary conditions.

2. Algorithm Theory

The algorithm is divided into two parts: Local SLAM (front-end detection) and Global SLAM (back-end closed-loop).

There are four sensor Data input here: Range Data (LiDAR, camera, etc.), Odometry Pose(Odometry Data), IMU Data, FixedFramePose(which I understand to mean determined location).

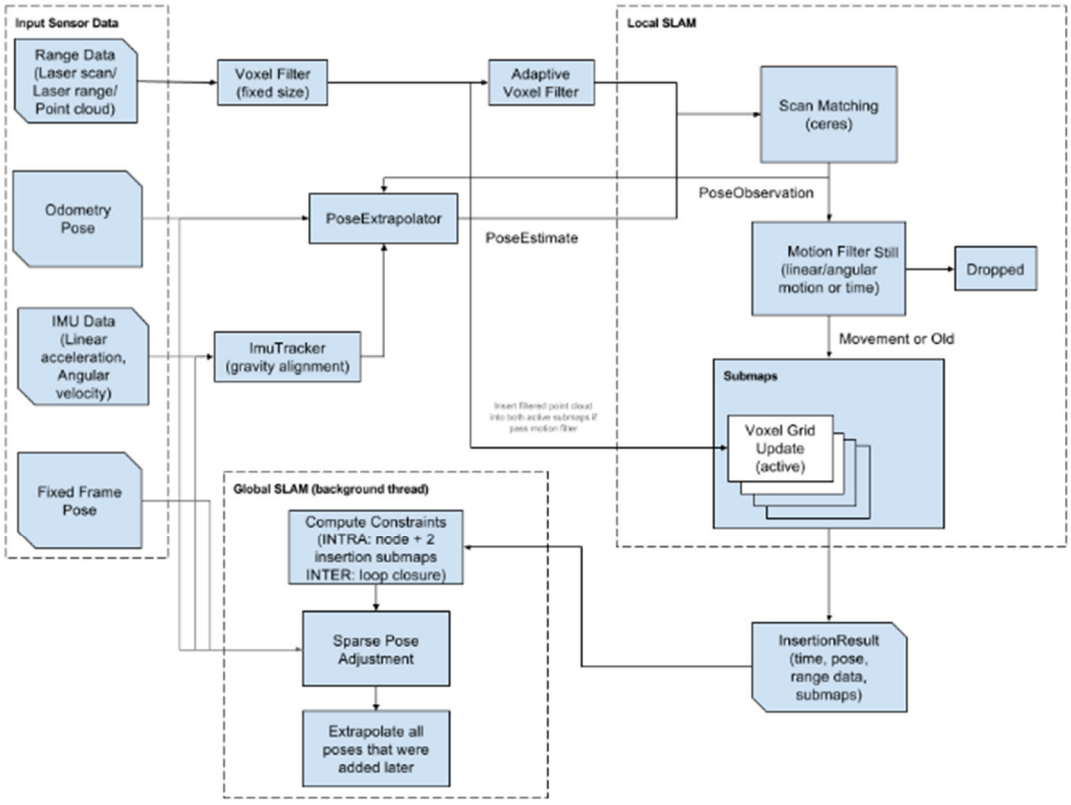

Figure 1. Cartographer block diagram

The odometry and data are entered into the PoseExtraPolator together with the IMU data, where dead reckoning is done. Given a position estimate obtained by the odometry and IMU, the specific method is very simple: calculate the average speed with the odometry more than twice, calculate the average angular velocity with the IMU data more than twice, and then calculate the position and attitude at the next moment. Then it is given to Scan Matching as the initial value of Scan Matching.

The Range Data is then filtered by voxel (a filtering method) and adaptive voxel filtering, and then entered into scanMatching as the observed value, Here, scanMatching uses the CERES-optimized scanMatching method introduced

in the paper of Real-time Loop Closure in 2D LIDAR SLAM to obtain the optimal position estimation. After Motion Filter, As the optimal position estimation, submap is constructed.ScanMatching method more interesting here, matches the probability of each grid point double cubic spline interpolation is smooth, and then build optimization problem as follows, M that behind the probability, probability value of 0 to 1 of a number, so M the, the greater the 1 - the smaller M, the highest goal is to obtain a probability optimization function.

$$arg\,min \sum_{k=1}^{k} \left(1 - M_{smooth}(T_{\varepsilon} h_k)\right)^2 (CS)$$

The Local SLAM section gets a lot of submaps, which are then fed into the Global SLAM section for optimization.

Cartographer's main theory is to eliminate the accumulated errors caused by the composition process through closed-loop detection.The basic unit used for closed-loop detection is SubMap.A submap is made up of a certain number of Laser Scans.When a Laser Scan is inserted into its corresponding submap, its optimal position in the submap is estimated based on the existing Laser Scan and other sensor data of the submap.The error accumulation of Submap creation in a short period of time is considered to be small enough.However, as more and more submaps are created over time, the error accumulation between submaps will become larger and larger.Therefore, it is necessary to properly optimize the pose of these submaps through closed-loop detection to eliminate these accumulated errors, which transforms the problem into a pose optimization problem.When the construction of a submap is completed, that is, no new Laser Scan will be inserted into the submap, the submap will be added to the closed-loop detection.The closed loop detection takes into account all the submaps that have been created.When a new Laser Scan is added to the map, if the estimated pose of the Laser Scan is close to the pose of a Laser Scan of a submap in the map, the closed loop will be found by some Scan match strategy.The Scan match strategy in Cartographer searches for a possible match of the Laser Scan by taking a window near the estimated pose of the Laser Scan newly added to the map. If a good enough match is found, Then the closed-loop constraint of this matching will be added to the position pose optimization problem.The key contents of Cartographer are the creation of local submap fused with multi-sensor data and the implementation of Scan match strategy for closed-loop detection.

The code of Google open source contains two parts: Cartographer and Cartographer_ROS.Cartographer is mainly responsible for processing data from radar, IMU and odometer and building maps based on these data, which is the underlying implementation of Cartographer's theory.Cartographer_ros obtains the sensor data based on the communication mechanism of ROS, converts them into the format defined in Cartographer and transfers them to Cartographer for processing. At the same time, Cartographer publishes the processing results for display or saving. Is an upper-layer application based on Cartographer.

3. Environment Building

1. Install dependencies

2 Download the installation package

3. The compilation

4. Test

3.1. The general framework of Cartgrapher's algorithm

A total of three software packages were downloaded to install Cartographer, among which Cartographer is our research target and Cartographer_ROS is its encapsulation in ROS environment.Ceres-solver is an optimization library, which is used by Cartographer to solve the optimization problems in the process of updating local map and global optimization.

The RosNode tool is used to view the nodes running in the current system, or the RQT_Graph tool is used to obtain the subscription relationship diagram between nodes and topics as shown in the right figure.

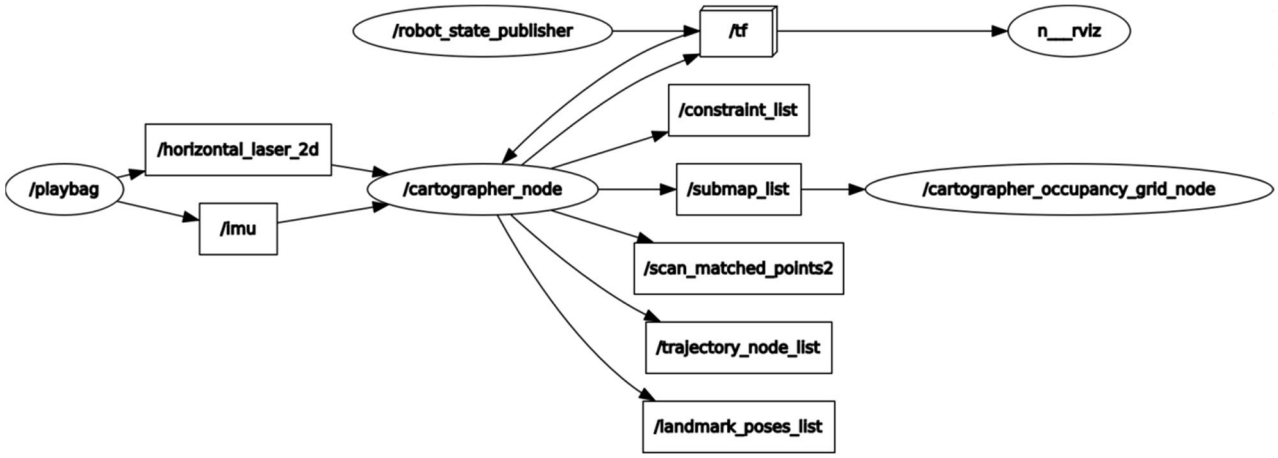

Figure 2. Node graph

There are a total of 6 nodes in the system.

/rosout is the node used by the ROS system to manage log output and has no relation to the actual business logic

Both /rviz and /playbag can be found in the launch script. /cartographer_node and/cartographer_Occupancy_grid_node are the key to the whole diagram.

/robot_state_publisher is mainly used to maintain the coordinate transformation relationship between sensor and robot body.

/ Cartographer_node subscribed to/horizontal_LASer_2D and/IMU topics, which are the LiDAR and IMU data played back through/PlayBag, respectively.

Based on the input sensor data, the/Cartographer_node completes the localization and subgraph construction and publishes the /submap_list topic.

/ Cartographer_Occupancy_grid_node will subscribe to the /submap_list theme and build the occupancy raster map according to the subgraph.

3.2. Official ROS package -Cartographer_ROS

3.2.1. Source directory of Cartographer_ROS

The files in this directory are some explanatory properties and Docker configuration under different ROS environment, not table.The Docs directory contains documentation, the Jenkins directory contains configuration related to the continuous integration tool Jenkins, and scripts contains some automation scripts related to installation.The only directories associated with Cartographer are cartographer_ros, Cartographer_ros_MSgs and Cartographer_rviz.Among them, many MSG and SRV files are defined in Cartographer_ros_MSgs, which define various message types for Cartographer.Cartographer_rviz is a visualization-related project.Cartographer_ros here, which is the main body of the official ROS package.

3.2.2. Source directory of Cartographer_ROS Launch script

Three nodes are loaded and run, coordinate transformation is maintained, Cartographer system is run for positioning and mapping.

As can be seen from the figure, "RoBOT_state_publisher" maintains the coordinate transformation of the laser sensor and IMU mounted horizontally and vertically with respect to the base link.

"Cartographer_node" maintains the coordinate relationship among MAP, odometer and robot base.

The script backpack_2d.launch loads the node

" cartographer_node", which specifies a configuration file through a series of parameters. This node does the positioning and subgraph construction.Finally,

the "cartographer_occupancy_grid_node" is loaded to merge the subgraphs into an occupancy raster map and publish it via ROS themes.

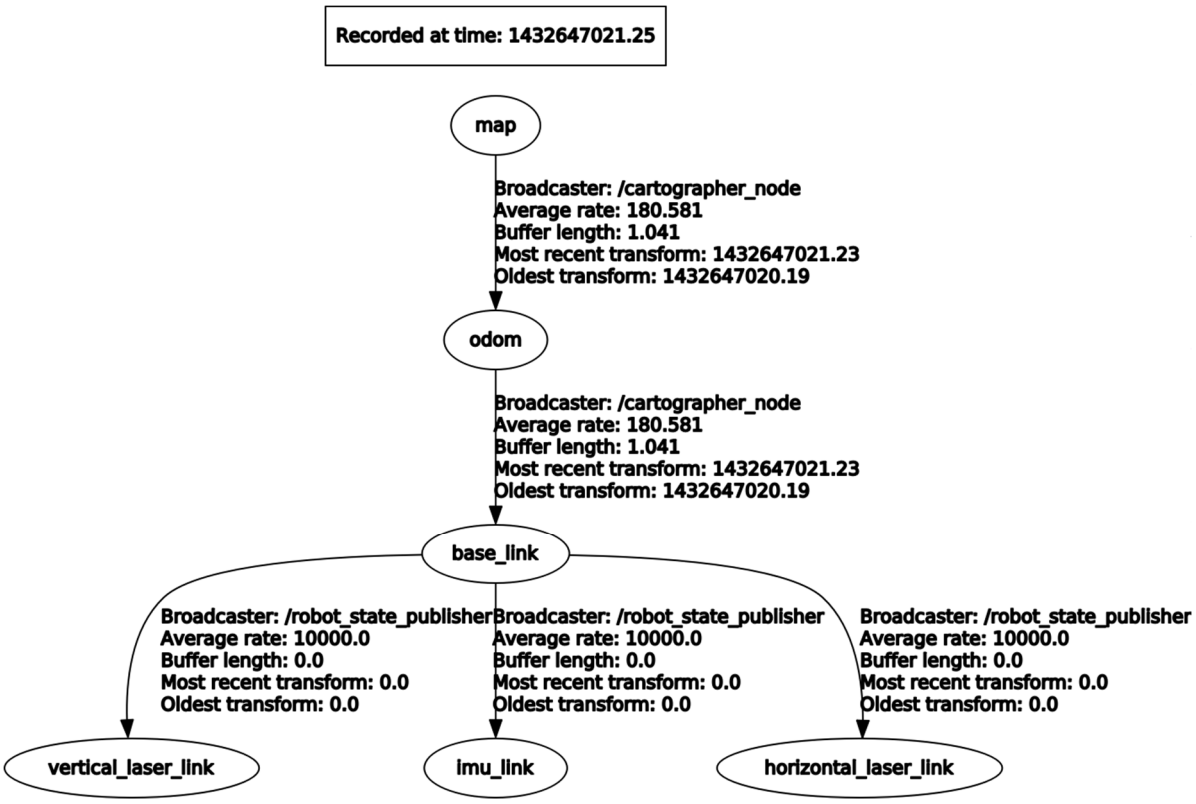

Figure 3. TF tree

The simulation test results are shown below

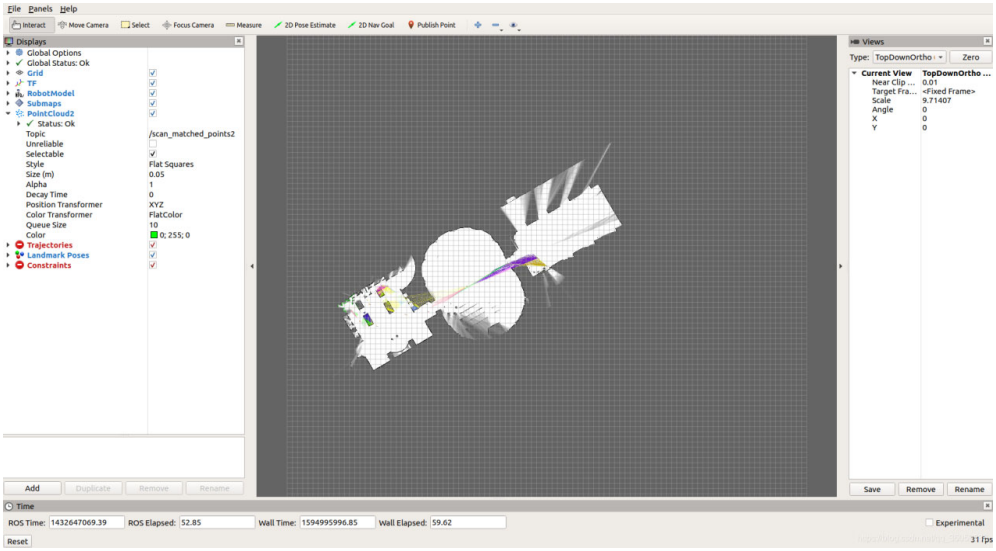

Figure 4. Simulation renderings

4. Three-platform Verification

Experimental results show that the algorithm can be implemented on this platform.

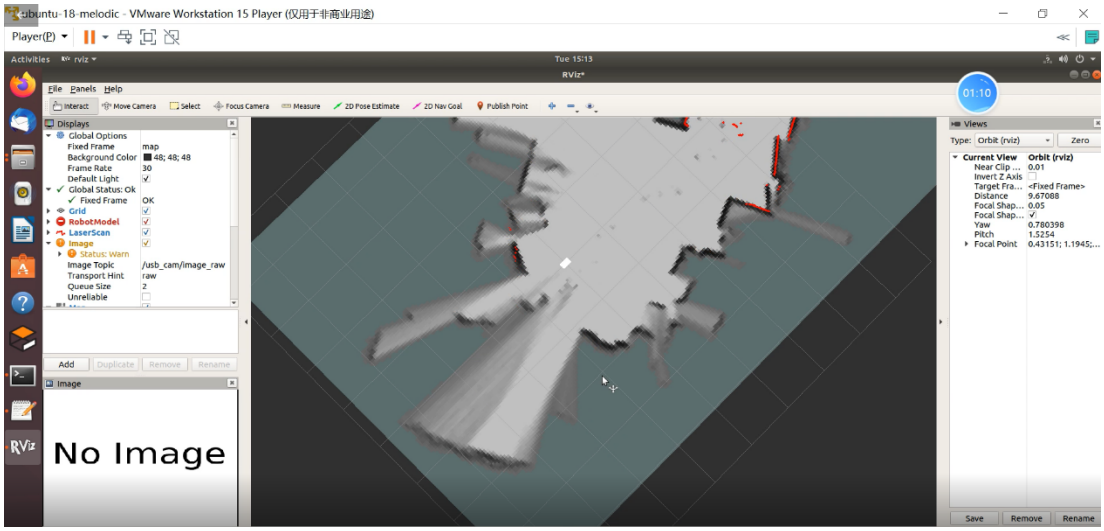

Figure 5. Actual renderings

References

[1] Abdurahman Dwijotomo et al. Cartographer SLAM Method for Optimization with an Adaptive Multi-Distance Scan Scheduler[J]. Applied Sciences, 2020, 10(1)

[2] Hoshi Masahiko and Hara Yoshitaka and Nakamura Sousuke. Graph-based SLAM using architectural floor plans without loop closure[J]. Advanced Robotics, 2022, 36(15) : 715-723.

[3] Eldemiry Amr et al. Autonomous Exploration of Unknown Indoor Environments for High-Quality Mapping Using Feature-Based RGB-D SLAM[J]. Sensors, 2022, 22(14) : 5117-5117.