Robot Patrolling in My Home

Zhaofeng Tian

Gy8217

I. Problem Description

In recent years, robotic industry has developed rapidly. Intelligent mobile robots are gaining increasing attention from the public, including patrol robot, cleaning robot, and service robot, etc. Among these robots, they all have the function of localization, map building and path planning. These three functions are all classified in the field of navigation, and also have a relationship of mutual dependence. In order to complete the navigation task, the robot needs to know the position on the map. Moreover, the robot also needs to know how to arrive to the destination on the best path. Sometimes the robot needs to perform tasks like patrolling among several waypoints. Therefore, it is a challenge to develop a multi-waypoints patrolling navigation system. This project implements the patrolling function on a mecanum-wheel- based robot with ROS stacks and python coding.

II. Literature Review

In order to complete the patrolling navigation task, the robot needs a related environment map. In [1, 2], a method named grid mapping (Gmapping) was utilized to build the map. The function of simultaneous localization and mapping (SLAM) is included in Gmapping. In [3], an indoor mapping method using the Kinect sensor was proposed. The Kinect sensor can reduce costs compared to the laser sensor. However, the accuracy of the Kinect sensor is worse than the laser sensor. Thus, the map built with the Kinect sensor needs to modify for the navigation system. In the research areas of path planning, A* is a popular graph search algorithm. In [4, 5], the map was divided into many grids. The best path can be obtained by calculating the weight of each grid. Dijkstra algorithm [6, 7] is also a widely utilized for path planning. The difference between A* and Dijkstra algorithm lies in the area of calculation. In Dijkstra algorithm, all grids in the map are the calculation area. In A* algorithm, the destination is the priority search direction. However, A* algorithm has a disadvantage that it may cause the robot to walk too close to the wall or obstacle. In [8], an improved A* algorithm was proposed to overcome this shortcoming. In [9, 11, 12, 13], Monte Carlo localization (MCL) method was proposed for the robot localization. By calculating weights through particle filter, the robot's position can be obtained on the map. This algorithm does not need to receive Wi-Fi signals, and it can avoid the situation of poor Wi-Fi signal. However, MCL may cause the kidnapped robot problem. Some particles with too high weight values will take the robot to the wrong position. In [10], augmented Monte Carlo localization (AMCL) was proposed to solve this problem by adjusting the amount of the particles.

III. Architecture Details

This is the node architecture of navigation stack on my robot shown as below:

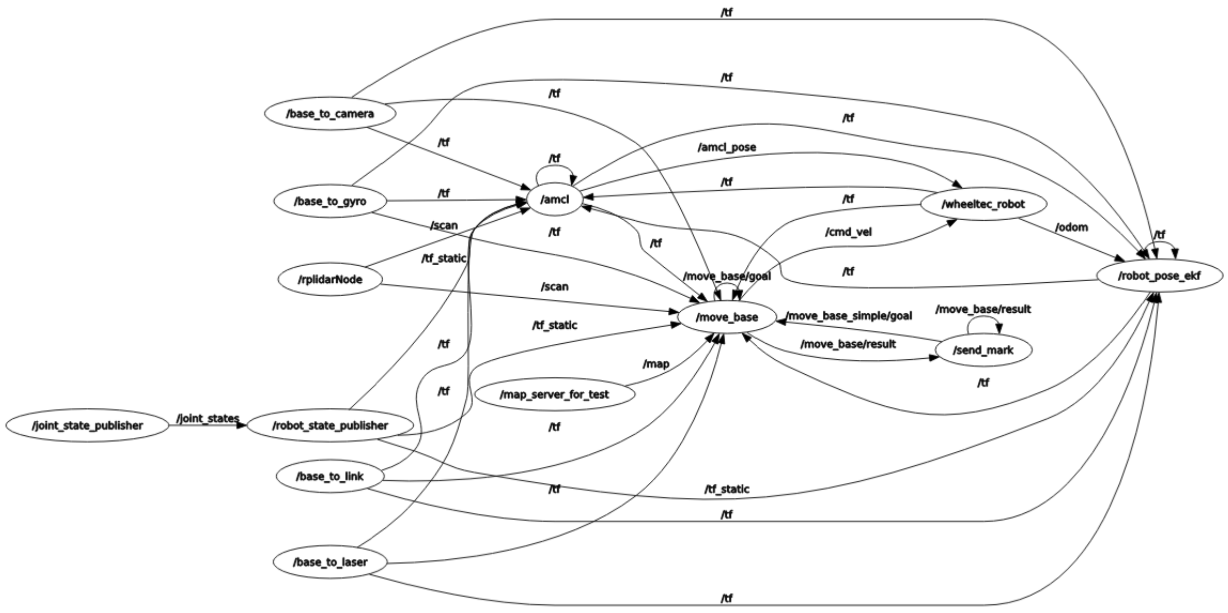


Fig.1 Navigation nodes

1. Localization method:

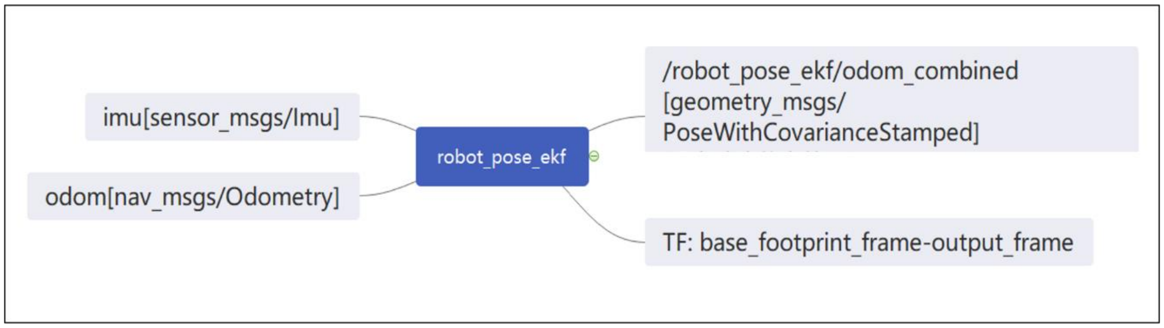

imu[sensor_msgs/Imu]
odom[nav_msgs/Odometry]
robot_pose_ekf
/robot_pose_ekf/odom_combined [geometry_msgs/ PoseWithCovarianceStamped]
TF: base_footprint_frame-output_frame

Fig.2 Localization node

The ROS navigation stack is using a package called “robot_pose_ekf” to implement the localization. As we can see, it subscribes to “odom” and “imu” topics, and publish to “TF” (the coordinate relationship between the map and the car) and “combined_odom” topics. Meanwhile, “AMCL” will help localization as an assistant package to compensate for the accumulated errors of the odometer.

2. Move_base package:

Look at the following picture, it shows the structure of the move_base package and its connection to other nodes. Look from the top to the bottle, the move_base node subscribe to the navigation target and then publish “cmd_vel” that uses Twist message to the base controller of robot platform. The “AMCL” node could publish TF messages to compensate the accumulated errors of the odometer; The sensor transform node will publish the TF from “odom_combined” to “base_footprint”.

Additionally, the costmap can be adaptively updated with map information from map_server and sensor information from sensors on the robot.

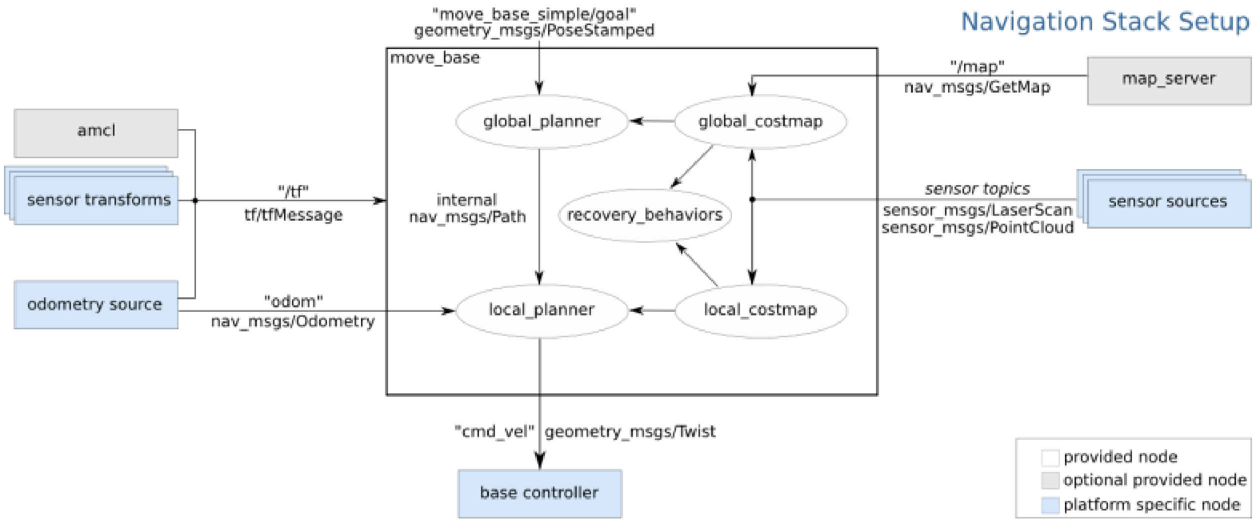


Fig.3 Move_base node and its connections

3. AMCL package

The functionality of AMCL package is to compensate for the accumulated. Amcl node is subscribing TF coordination of the robot base_footprint and lidar information and calculating a deviation of position information on the map, then publishing map to TF of the odom_combined, which is to compensate for the deviation of base_print on the map.

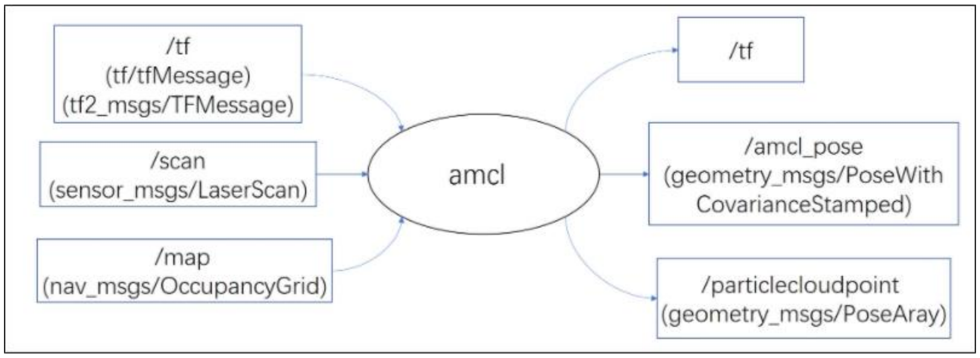


Fig.4 Amcl node

IV. Implementation Details

1. Firstly, this study uses gmapping to get a map of my home using:

```
# launch mapping, mapping method is set in gmapping.

$ roslaunch turn_on_wheeltec_robot mapping.launch

# When gmapping is on, turn on the teleoperation to move the robot with my
keyboard.

$ roslaunch wheeltec_robot_rc keyboard_teleop.launch

# Save the map

$ rosrun map_server map_saver -f my_home
```

2. Secondly, this study launches the navigation stack:

```
# launch the navigation stack

$ roslaunch turn_on_wheeltec_robot navigation.launch
map_file:=$HOME/my_home.yaml

# open rviz

$ rviz
```

3. Run patrol python file, the patrol waypoints are listed in the patrol file, the details can be obtained from code zip file. The following image is an example of patrol codes.

The code is to send navigation waypoints to the MoveBaseAction to move the robot.

```
$ rosrun navigation navi.py
```

In summary, the procedure of this project is that

- Launch the gmapping, using keyboard teleoperation to navigate the robot through the whole environment that you are interested in, then save the map.
- Launch the navigation stack specifying the saved map. Then open rviz.
- Run the python file to implement the patrol function.

V. Demonstration

This demonstration is based on my robot which has a mecanum-wheel platform shown as below; The mapping and navigation is implemented with the RPlidar on the top of it; The base of it is controlled by a STM32 board; The processor is Nvidia TX2 that connects to the STM32.

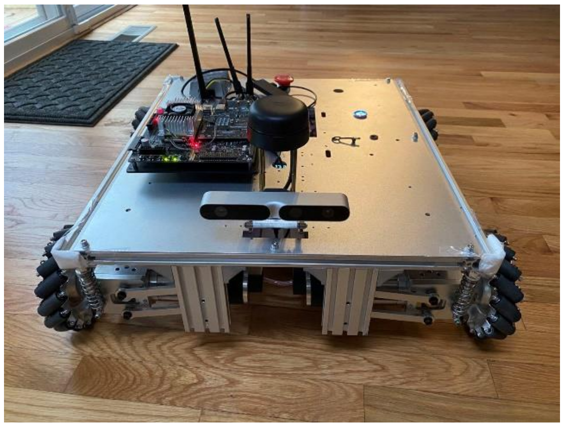
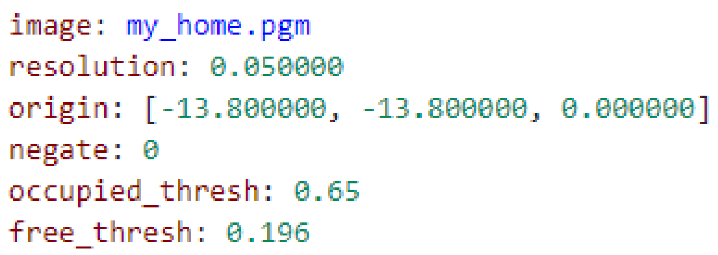


Fig.5 Robot in this project

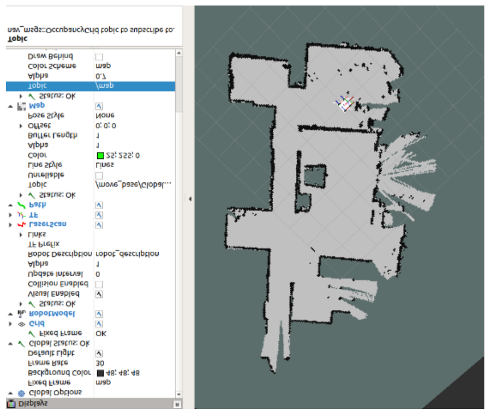
1. Mapping

Launch the mapping package and open the keyboard teleoperation to move the robot navigating through the home, exploring unknown spaces. The map is saved in the home file named my_home.yaml, the yaml file and visualization in Rviz are shown below. Note that the resolution is set at 0.05 which could be modified in the Gmapping settings.



```
image: my_home.pgm
resolution: 0.050000
origin: [-13.800000, -13.800000, 0.000000]
negate: 0
occupied_thresh: 0.65
free_thresh: 0.196
```

(a). Map yaml file



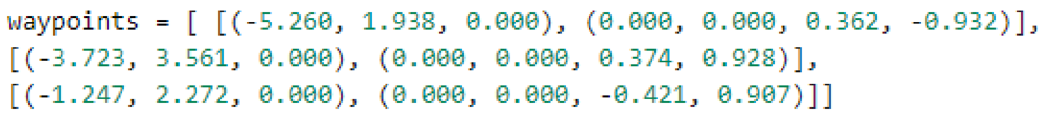
(b). Map visualized in Rviz

Fig.6 Map information

2. Navigation & Patrol

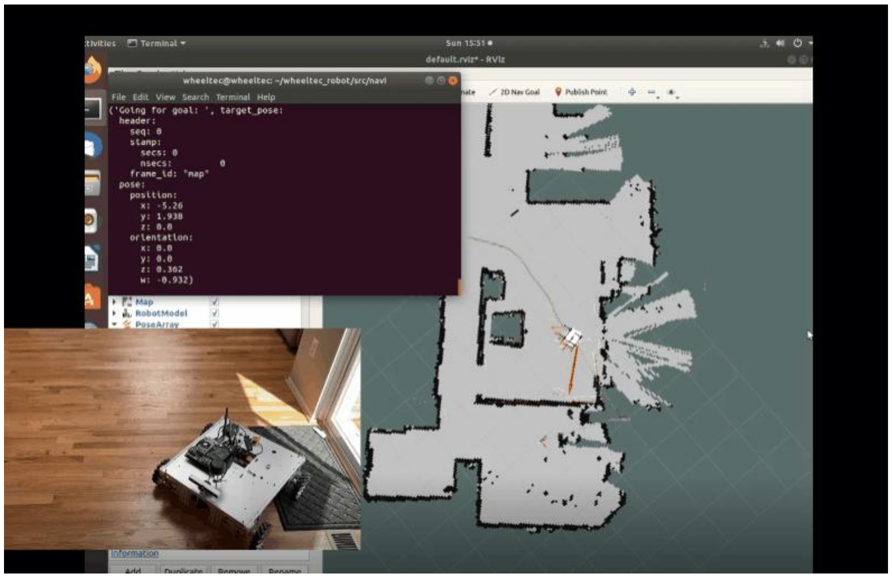
Launch the navigation stack, then estimate the initial pose of the robot in the rviz. Once it is done, you can either navigate with manually sent goal generated by clicking on the rviz map or with a python file that sending preset goals to the move.

To implement the patrol on the robot, we will use the second method, to use a python file to automatically send the navigation goals. I choose three waypoints to patrol on. After running the python file, the result can be obtained, which is also shown below.



```
waypoints = [ [(-5.260, 1.938, 0.000), (0.000, 0.000, 0.362, -0.932)],
[(-3.723, 3.561, 0.000), (0.000, 0.000, 0.374, 0.928)],
[(-1.247, 2.272, 0.000), (0.000, 0.000, -0.421, 0.907)]]
```

Fig.7 Three preset waypoints in the python file



```
('Going for goal: ', target_pose:
  header:
    seq: 0
    stamp:
      secs: 0
      nsecs:         0
    frame_id: "map"
  pose:
    position:
      x: -5.26
      y: 1.938
      z: 0.0
    orientation:
      x: 0.0
      y: 0.0
      z: 0.362
      w: -0.932)
```

Fig.8 Patrolling demonstration

3. Demo Resource

To get the full video of this demonstration, please go to:
https://www.linkedin.com/posts/zhaofeng-tian-104132191_recently-tried-a-robot-patrol-using-ros-and-ugcPost-6793572047478951936-KE9J

To get related packages or codes, please go to:

https://github.com/Zhaofeng-Tian/ROS_Robot_Patrol

VI. Conclusion

This project successfully implemented an indoor-patrolling function using a mecanum-wheel-based robot. The map was built with gmapping algorithm, and by using this indoor-map, three waypoints are set in the python file which could are sending the waypoints to movebase action to navigate the robot. Finally, the robot can navigate among these three points in a preset sequence without big deviations.

VII. Reference

[1] G. Grisetti, C. Stachniss, and W. Burgard, “Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling”, Proceedings of 2005 IEEE International Conference on Robotics and Automation, Barcelona, pp. 2432-2437, April 2005.

[2] G. Grisetti, C. Stachniss, and W. Burgard, “Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters”, IEEE Trans. on Robotics, Vol. 23, No. 1, pp. 34-46, Feb. 2007.

[3] H. I. M. A. Omara and K. S. M. Sahari, “Indoor Mapping using Kinect and ROS”, Proceedings of 2015 International Symposium on Agents, Multi-Agent Systems and Robotics (ISAMSR), pp. 110-116, Aug. 2015.

[4] W. Zeng and R. L. Church, “Finding Shortest Paths on Real Road Networks: The Case for A*”, International Journal of Geographical Information Science, Vol. 23, No. 4, pp. 531-543, 2009.

[5] J. Yao, C. Lin, X. Xie, A. J. Wang, and C.-C. Hung, “Path Planning for Virtual Human Motion Using Improved A* Algorithm”, Proceedings of 2010 Seventh International Conference on Information Technology: New Generations, pp. 154-1158, 2010.

[6] N. Jasika, N. Alispahic, A. Elma, K. Ilvana, L. Elma, and N. Nosovic, “Dijkstra's Shortest Path Algorithm Serial and Parallel Execution Performance Analysis”, Proceedings of the 35th International Convention MIPRO, pp. 1811-1815, May 2012.

[7] E. W. Dijkstra, “A Note on Two Problems in Connexion with Graphs”, Numerische Mathematlk, Vol. 23, No. 1, pp. 269-271, 1959.

[8] K.-Y. Wei, I-H. Lee, C.-C. Hsu, and W.-Y. Wang, “Mobile Robot Navigation System Using Distributed Computing System Based on ROS Architecture”, Proceedings of Joint 17th World Congress of International Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems, June 2017.

[9] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, “Monte Carlo Localization for Mobile Robots”, Proceedings of 1999 IEEE International Conference on Robotics and Automation, May 1999.

[10] S. Thrun, W. Burgard, and D. Fox, Probabilistic Robotics, MIT Press, 2005.

[11] Fox D, Burgard W, Dellaert F, Thrun S. Monte carlo localization: Efficient position estimation for mobile robots. AAAI/IAAI. 1999 Jul 18;1999(343-349):2-.

[12] Pfaff P, Burgard W, Fox D. Robust monte-carlo localization using adaptive likelihood models. InEuropean robotics symposium 2006 2006 (pp. 181-194). Springer Berlin/Heidelberg.

[13] Doucet, Arnaud. "On sequential simulation-based methods for Bayesian filtering.", 1998.