Lab1 - Introduction to Turtlebot

Probabilistic Robotics

Songyou Peng
psy920710@gmail.com

March 7th, 2016

1 Implementation

In order to control turtlebot to move, the most important thing is to find the topics to publish velocity and to subscribe the turtlebot's pose information. It was not easy to find this two topic actually. First, we tried to read the rqt_graph, which is shown in the Fig. 1.

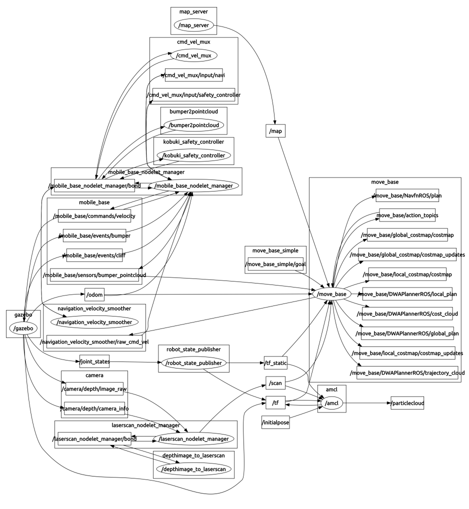
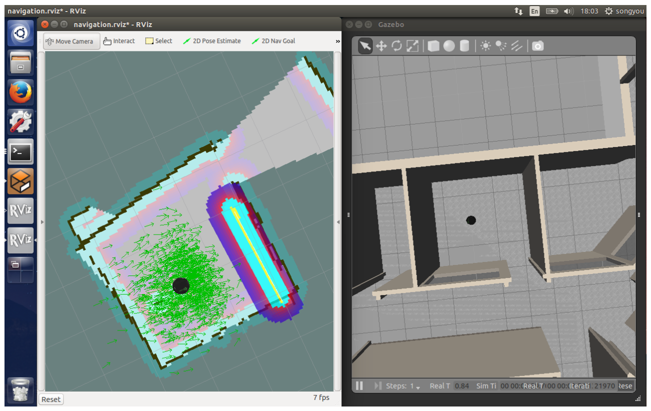

map_server
/map_server
cmd_vel_mux
/cmd_vel_mux
/cmd_vel_mux/input/navi
/cmd_vel_mux/input/safety_controller
bumper2pointcloud
/bumper2pointcloud
kobuki_safety_controller
/kobuki_safety_controller
mobile_base_nodelet_manager
/mobile_base_nodelet_manager/bond
/mobile_base_nodelet_manager
mobile_base
/mobile_base/commands/velocity
/mobile_base/events/bumper
/mobile_base/events/cliff
/mobile_base/sensors/bumper_pointcloud
/odom
gazebo
/gazebo
navigation_velocity_smoother
/navigation_velocity_smoother
/navigation_velocity_smoother/raw_cmd_vel
/joint_states
robot_state_publisher
/robot_state_publisher
camera
/camera/depth/image_raw
/camera/depth/camera_info
laserscan_nodelet_manager
/laserscan_nodelet_manager/bond
/laserscan_nodelet_manager
depthimage_to_laserscan
/depthimage_to_laserscan
/map
move_base_simple
/move_base_simple/goal
move_base
/move_base/NavfnROS/plan
move_base/action_topics
/move_base/global_costmap/costmap
/move_base/global_costmap/costmap_updates
/move_base/local_costmap/costmap
/move_base
/move_base/DWAPlannerROS/local_plan
/move_base/DWAPlannerROS/cost_cloud
/move_base/DWAPlannerROS/global_plan
/move_base/local_costmap/costmap_updates
/move_base/DWAPlannerROS/trajectory_cloud
/tf_static
/scan
/tf
/initialpose
amcl
/amcl
/particlecloud

Figure 1: rqt_graph

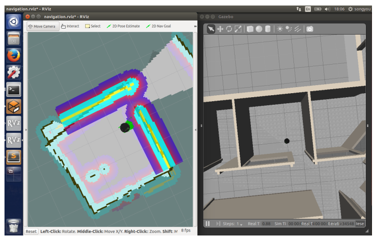
Fig. 1 is a complicated graph but we can still guess that topic `/mobile_base` and `/cmd_vel_mux` are about velocity. Because we used `/turtle1/cmd_vel` in lab0, so we tried topic within `/cmd_vel_mux`. It turns out both `/cmd_vel_mux/input/navi` and `/cmd_vel_mux/input/teleop` can be used as the topic to publish velocity. The message of them are `Twist`. As for subscriber, we find out that the message of topic `/odom` is `Odometry`, which contains not only the position but also the orientation of the turtlebot. Therefore, this topic is our subscriber.

It should be noted that the orientation infomation obtained from topic `/odom` is in $4D$ quaternion system, so we import `euler_from_quaternion function` from `tf.transformations` for the sake of the angle computation.

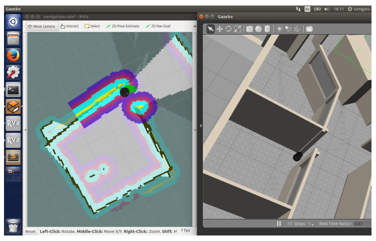
When all the parameters are obtained, we can implement something similar to the last lab and move the turtlebot from one place to the other. In the Fig. 2(a), we can see the Gazebo world on the right, and one the left, the map we saved before is loaded in rviz. And then we specify the goal point $(x, y, theta)$, where the turtlebot goes to . The reuslts are shown in Fig. 2(b) and Fig. 2(c).



(a) The Gazebo world and map before moving turtlebot



(b) $x = 1, y = 1, theta = 0$



(c) $x = 2, y = 2, theta = 0$

Figure 2: Results

2 Discussions

Like in the previous lab, instead of giving priority to the rotation, we let turtlebot rotate and go straight simultaneously. The linear and angular velocities are proportional to the distance to the goal and the angle difference between goal and robot. However, not like the turtlesim of lab0, which is an ideal environment, the gazebo is more like a real world, so at first, we tried to use the same speed strategy as the lab0, but the turtlebot just ran away crazily. What we change is to tune both linear and angular velocity parameters in order to get the trade-off between speed and accuracy.

However, we still notice that the turtlebot works well in most cases but sometimes may go to other places instead of goal point when the goal has minus position coordinate, for example, $x = -2, y = -2$. We find out that the main reason of the problem should owe to centrifugal force. However, we still notice that the turtlebot works well in most cases but sometimes may go to other places instead of goal point especially when the goal has minus position coordinate, for example, $x = -2, y = -2$. We find out that the main reason of the problem is to owe to centrifugal force. Usually, the angular and linear velocity are relatively high at the beginning, so the turtlebot will easily spin around because of the centrifugal force and then miss the correct routes. Our solution is to reduce the proportional parameters of both velocities. It turns out the turtlebot is able to go to all the points.

Due to the fact that both velocities are relatively slow, and the closer the turtlebot to the goal, the slower the speed is, it takes a long time to reach the goal. We notice when the distance to the goal is smaller than a certain number (1 in our case), the angle between turtlebot and goal is really small, so we come up with a solution that we give a static linear speed after that distance. Therefore, when the turtlebot is already near to the goal, we give an a bit high speed in order to smoothly arrive the goal.

3 Conclusions

In this lab, we understand how to publish and subscribe the topics of the pose and velocity of the turtlebot, and also, the algorithm of moving the turtlebot from one point to another has been implemented. Although there are some problems of ROS (Gazebo, rviz, virtual machine, etc.), the lab task has been achieved.