User Manual
for *placebot-sim* Package

Author: Ali Mantash

1 INTRODUCTION

Placebot is a robot base that features:

- 4 omni-directional wheels each capable of free 360 degrees steering angle.

- 2 sick-s300 lidar sensors mounted on opposite corners allowing a 360 degrees coverage and 30 meters range scans.

Placebot-sim package is built to simulate a placebot model in gazebo environments. It contains the robot model description including sensors and actuators, base controller, gazebo plugins and supplementary scripts.

Published topics:

- /odom (nav_msgs/Odometry

- /scan (sensor_msgs/LaserScan)

Subscribed topics:

- /cmd_vel (geometry_msgs/Twist)

Implemented packages:

- g-mapping for SLAM

- AMCL for localization

- Navigation stack for autonomous navigation

2 PREREQUISITES:

- ROS Kinetic and Gazebo 7 installed and configured properly on Ubuntu 16.04

Refer to the guide included in this package on how to install ROS and Gazebo for detailed instructions.

3 PACKAGE INSTALLATION:

Create a catkin workspace; you can use any name; in this tutorial we will call it “catkin_ws”.

$ source /opt/ros/kinetic/setup.bash

$ mkdir -p ~/catkin_ws/src

$ cd ~/catkin_ws/

$ catkin_make

$ source devel/setup.bash

Copy the placebot-sim package folder into ~/catkin_ws/src, and build workspace again:

$ cd ~/catkin_ws/

$ catkin_make

$ source devel/setup.bash

You might get errors related to missing package dependencies; install those packages and move to next step.

(NB:

You can also use rosdep tool to update and install packages dependencies:

$ cd ~/catkin_ws/

$ rosdep update

$ rosdep install --from-paths src -i

$ catkin_make

$ source devel/setup.bash

)

At this point, the placebot sim package is installed and ready to use and develop on.

You can now start simulating placebot in gazebo, run packages already implemented and listed above, and implement new packages.

4 HOW TO RUN THE SIMULATION AND THE ASSOCIATED PACKAGES:

Placebot-sim contains a set of launch files that make it easy for users to run its packages.

All you have to do is roslaunch those files from a terminal with the ability to set parameter values related to each package.

4.1 LAUNCH PLACEBOT IN GAZEBO:

This will launch gazebo and rviz, load placebot model including actuators and sensors, and run its controller and odometery publisher nodes.

Subscribed topics:

- /cmd_vel

Published topics:

- /odom
- /scan

The controller listens to velocity commands on /cmd_vel topic and commands the wheels axles and steering actuators accordingly.

Odometry data is published on /odom topic, and lidar sensor data is published on /scan topic.

$ roslaunch placebot_gazebo placebot_gazebo.launch

4.2 LAUNCH TELEOP NODE:

Teleop node is used to control the placebot base with a keyboard or joystick; it maps the keyboard or joystick strokes to Twist messages and published on /cmd_vel topic.

In a new terminal type:

$ roslaunch placebot_gazebo teleop.launch

You will have to click on this terminal when driving placebot in order for the node to register the key strokes.

4.3 LAUNCH G-MAPPING NODE:

ROS g-mapping node from the gmapping package implements SLAM to build a map based on odometry and laser sensor data. The node has a lot of parameters that can be tuned to better fit different environments and applications.

In a new terminal:

$ roslaunch placebot_gazebo g-mapper.launch

In rviz add the following topics:

- sensor laser
- robot model
- map

You should see obstacles detected by the laser sensors marked with white dots; you can adjust the shape and color of marks.

You should also see a map being built and updated as you navigate the robot around.

In the terminal where you launched teleop, use the keyboard keys to drive the robot around until you are satisfied with the map visualized in rviz.

Now open a new terminal and save the map:

$ rosrun map_server map_saver -f <filename>

Filename will be the name and directory for the map files, ex:

$ rosrun map_server map_saver -f map1

This will create two files: map1.pgm and map1.yaml

4.4 LAUNCH AMCL NODE:

This node implements the adaptive (or KLD-sampling) Monte Carlo localization approach, which uses a particle filter to track the pose of a robot against a known map; it publishes the robot position in the map on /amcl_pose topic, and publishes transform from odom to map frames.

In a new terminal:

$ roslaunch placebot_gazebo amcl.launch

4.5 LAUNCH MOVE-BASE NODE:

Move-base along with other packages in the navigation stack is used for the autonomous navigation of robots.

Move-base requires a set of configuration files containing parameters for the following packages:

- Move-base
- Global path planners
- Local path planners (we use DWA local path planner)
- Global and local cost maps

Refer to the ros wiki pages for detailed description of navigation stack and all its sub packages.

For a more comprehensive guide on how to tune and troubleshoot navigation stack parameters, refer to the following links:

- http://wiki.ros.org/navigation/Tutorials/Navigation%20Tuning%20Guide
- http://kaiyuzheng.me/documents/navguide.pdf

To launch move-base, type in a new terminal:

$ roslaunch placebot_gazebo move_base_placebot.launch

You can now send “goal” commands to the move-base server, and robot will attempt to navigate to the specified location while avoiding obstacles.

The movebasegoal type msg contains frame_id, timestamp, and the goal linear and angular position fields. For a simplified guide, refer to the guide on move-base included in this package.