Experiment 9

Familiarization of 2D navigation stack, Basic ROS navigation

Objective:

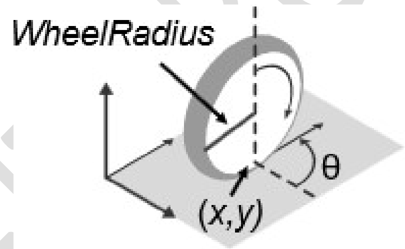
1. Launch a mobile robot in the Gazebo for 2D navigation
2. Control the linear and angular velocities for the robot

Theory

In order to know how a mobile robot is working, one should know the concept of unicycle model. It is explained as follows

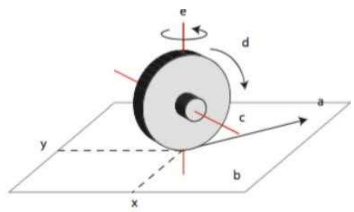
Kinematic Modelling of a differential drive robot- Unicycle Model

A unicycle model of control a mobile robot is a simplified modelling approach modified from the differential drive mobile robots. Instead of controlling the right speed, and the left speed of the drive systems, the unicycle model is using uu and $\omega\omega$ as the controller parameters. Tracking is much easier in this model.



Unicycle Kinematic Model

$$\dot{q} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$



v: The linear velocity of the contact point between the wheel and the ground and is equal to the product between angular velocity of the wheel around the horizontal axis and the radius of the wheel

ω: The angular velocity of the robot.

So by controlling the two values the navigation can be controlled

Procedure

1. **Install Simulation Package**: The TurtleBot3 Simulation Package requires turtlebot3 and turtlebot3_msgs packages as prerequisite.
   $ cd ~/catkin_ws/src/
   $ git clone -b kinetic-devel https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git
   $ cd ~/catkin_ws && catkin_make
2. **Launch Simulation World:** Three simulation environments are prepared for TurtleBot3. Please select one of these environments to launch Gazebo. Please make sure to completely terminate other Simulation world before launching a new world.
   a. **For empty world**
      $ export TURTLEBOT3_MODEL=burger
      $ roslaunch turtlebot3_gazebo turtlebot3_empty_world.launch
   b. **TurtleBot3 World**
      $ export TURTLEBOT3_MODEL=waffle
      $ roslaunch turtlebot3_gazebo turtlebot3_world.launch
   c. **TurtleBot3 House**
      $ export TURTLEBOT3_MODEL=waffle_pi
      $ roslaunch turtlebot3_gazebo turtlebot3_house.launch
3. Teleoperation of robot: To remote control the robot, follow the code
   roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch
   Control the robot using the keyboard

```
---------------------------
 Moving around:
        w
   a    s    d
        x

 w/x : increase/decrease linear velocity
 a/d : increase/decrease angular velocity
 space key, s : force stop

 CTRL-C to quit
```

4. Autonomous Navigation: Create package or select an existing package and create a python file titled move.py to control linear and angular velocities
   Copy the following code

```
#!/usr/bin/env python

import rospy
from geometry_msgs.msg import Twist

rospy.init_node('topic_publisher')
pub = rospy.Publisher('/cmd_vel', Twist, queue_size=1)
rate = rospy.Rate(2)
move = Twist() # defining the way we can allocate the values
move.linear.x = 1 # allocating the values in x direction - linear
move.angular.z = 1 # allocating the values in z direction - angular

while not rospy.is_shutdown():
  pub.publish(move)
  rate.sleep()
```

5. The roslaunch command will open a turtlebot and a masternode to execute the code. Execute using rosrun package_name move.py

6. Try different linear and angular velocities and observe the change in robot navigation.
7.