Lesson 5 Lab sheet- Intelligent mobile robotics
LaserScan data and obstacle avoidance in tutulebot3

Aims

- Understand the LaserScan data
- Write a python code to read LaserScan data
- Create a package to subscribe to the topic related to laser scan
- Python Script for moving TurtleBot3
- Write a python code for obstacle avoidance

1. Understand LaserScan data

- Open the Tourtlebote3 gazebo simulator project that we practiced in the previous week, and Launch the Turtlebot in a non-empty environment:

  $ export TURTLEBOT3_MODEL=burger

  $ roslaunch turtlebot3_gazebo turtlebot3_world.launch

- Use the following command to get the list of active topics:$rostopic list
- Find a topic that is related to the laser scan, for instance it contains laser/scan,
- See the content of the topic using the following commend:

  $rostopic echo [topic name]

- You could ad '-n1' to only see one message from the topic:

  $rostopic echo [topic name] -n1

- Find the type of the message in this topic(use ' $rostopic info [topic name]'
- Use '$rossmg show [message type]' to see the structure of the message

angle_min and **angle_max** show the limits of field of view in radian which is started from -**angle_min** to **angle_max** degree. The array of 'ranges' shows the distance from obstacle which is calculated in different direction in the field of the view. The ranges might continue 360 readings that indicate the distances of the closest obstacle in 360 direction in the 360 degree of the field of the view of the robot. See sppendix A for more information. Let have a close look at the range array.

2. Create a package to subscribe to the topic related to laser scan:

- Create an empty package called 'laser_values' with rospy as dependency in 'catkin_ws/src' folder, using 'catkin_create_pkg' as we learn in the previous lab work.
- Go to 'laser_values' folder and create a launch folder by '$mkdir launch'
- Create a python script called 'scan.py' in the src folder in the package folder with the following code:

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

import rospy
from sensor_msgs.msg import LaserScan

def callback(msg):
    print (len(msg.ranges))
```

```
rospy.init_node('scan_values')
sub = rospy.Subscriber('/scan', LaserScan, callback)
rospy.spin()
```

- Use the following command to convert the mode of the python file, i.e. scan.py, to an executable file:

  $ chmod +x src/scan.py

- Create a launch file called ‘laser.launch’ in the launch folder for your package that contains the subscriber node described in the ‘scan.py’ file.

  <launch>

  <node pkg="laser_values" type="scan.py" name="scan_values" output="screen">

  </node>

  </launch>

- Launch ‘laser.launch’ using: $roslaunch laser_values laser.launch
- The output should be 360, that is the number of reading for 360-degree of field of view of the bot.
- Change the ‘callback (msg)’ function in the python code to print the range for the reading in degrees of 0, 90 and 180. Use msg.ranges[0] to reading the range (distance) for barrier which is in 0 degree, i.e. print (msg.ranges[0]). [1]
- Use run the teleoperate for the bot to move the bot using keyboard:
  $export TURTLEBOT3_MODEL=burger
  $ roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch

3. Python Script for moving TurtleBot3

- Create a Package called move_bot
- Create a python file called ‘trajectory.py’ (don’t forget to make it executable using $chmod +x name_of_the_file.py)

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

import rospy                                  # Import the Python library for ROS
from geometry_msgs.msg import Twist           # Import the Twist message from the std_msgs package

def talker():
        rospy.init_node('vel_publisher')              # Initiate a Node named 'vel_publisher'
        pub = rospy.Publisher('cmd_vel', Twist, queue_size=10)  # Create a Publisher object
        move = Twist()                       # Create a var named move of type Twist
        rate = rospy.Rate(1)                 # Set a publish rate of 0.5 Hz
        while not rospy.is_shutdown():
                move.linear.x = 1
                move.angular.z = 1
                pub.publish(move)
                rate.sleep()

if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

- Run the file using rosrun
  $rosrun <package> <executable>

- Create a launch file for your package contains the python code and run the code using ‘roslaunch’ commands [8].
  $roslaunch <package> <.launch file>

4. Write a python code for obstacle avoidance

- Write a new python code called ‘avoid_obstacle.py’ to move the robot, and it stops if it finds an obstacle with less than 0.5 meter distance.

```
import rospy
from sensor_msgs.msg import LaserScan
from geometry_msgs.msg import Twist

def callback(msg):          # Define a function called 'callback' that receives a parameter named 'msg'
    print('===================================================')
    print('s1 [270]')       #value right-direction laser beam
    print msg.ranges[270]
    print('s2 [0]')         #value front-direction laser beam
    print msg.ranges[0]     # print the distance to an obstacle in front of the robot. the sensor returns a vector
                            # of 359 values, being the initial value the corresponding to the front of the robot
    print('s3 [90]')        #value left-direction laser beam
    print msg.ranges[90]


    #If the distance to an obstacle in front of the robot is bigger than 1 meter, the robot will move forward
    if msg.ranges[0] > 0.5:
      move.linear.x = 0.5
      move.angular.z = 0.0
    else:
      move.linear.x = 0.0
      move.angular.z = 0.0

    pub.publish(move)

rospy.init_node('obstacle_avoidance')                  # Initiate a Node called 'obstacle_avoidance'
sub = rospy.Subscriber('/scan', LaserScan, callback)   # Create a Subscriber to the /scan topic
pub = rospy.Publisher('/cmd_vel', Twist)               #Create a publisher on the /cmd_vel topic
move = Twist()

rospy.spin()
```

- Change the previous code when the robot find a close obstacle to change it direction randomly. Use reference [3] to get help.

Appendix A

need explanation on sensor_msgs/LaserScan.msg [4]

```
Header header              # timestamp in the header is the acquisition time of
                           # the first ray in the scan.
                           #
                           # in frame frame_id, angles are measured around
                           # the positive Z axis (counterclockwise, if Z is up)
                           # with zero angle being forward along the x axis

float32 angle_min          # start angle of the scan [rad]
float32 angle_max          # end angle of the scan [rad]
float32 angle_increment    # angular distance between measurements [rad]

float32 time_increment     # time between measurements [seconds] - if your scanner
                           # is moving, this will be used in interpolating position
                           # of 3d points
float32 scan_time          # time between scans [seconds]

float32 range_min          # minimum range value [m]
float32 range_max          # maximum range value [m]

float32[] ranges           # range data [m] (Note: values < range_min or > range_max
should be discarded)
float32[] intensities      # intensity data [device-specific units].  If your
                           # device does not provide intensities, please leave
                           # the array empty.
-   To understand Intensities
```

```
-   if a laser beam hits reflective surface like glass it will have intensity 1. if
    beam hit some surface which absorbs laser , then intensity is zero. Middle value
    s are different surfaces in between
```

Appendix B) code for trajectory.py

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

import rospy
from geometry_msgs.msg import Twist

def talker():
    rospy.init_node('vel_publisher')
    pub = rospy.Publisher('cmd_vel', Twist, queue_size=10)
    move = Twist()
    rate = rospy.Rate(1)
    while not rospy.is_shutdown():
        print('------------')
        move.linear.x = 1
        move.angular.z = 1
        pub.publish(move)
        rate.sleep()

if __name__ == '__main__':
    try:
        talker()
        print('-------')
    except rospy.ROSInterruptException:
        pass
```

References:

[1] How to read LaserScan data (ROS python)
https://www.theconstructsim.com/read-laserscan-data/
[2] Getting laser data (python script)
http://www2.ece.ohio-state.edu/~zhang/RoboticsClass/docs/ECE5463_ROSTutorialLecture3.pdf
[3] Exploring ROS with a 2 Wheeled Robot #5 – Obstacle Avoidance
https://www.theconstructsim.com/exploring-ros-2-wheeled-robot-part-5/
[4] need explanation on sensor_msgs/LaserScan.msg
https://answers.ros.org/question/198843/need-explanation-on-sensor_msgslaserscanmsg/
[5] [RDS] 007 - ROS Development Studio #Howto use RViz and other ROS Graphical Tools in RDS
https://www.youtube.com/watch?v=xkS_OrMN2ag
[6] ROS Navigation
https://risc.readthedocs.io/1-ros-navigation.html#mapping
[7] tf tutorial
http://wiki.ros.org/tf/Tutorials
[8] Random moving
http://www2.ece.ohio-state.edu/~zhang/RoboticsClass/docs/ECE5463_ROSTutorialLecture3.pdf