Universitat De Girona

Medical Robotics

Introduction to ROS

Authors:
Ali Berrada
Ama Katseena Yawson

Supervisor:
Pere Ridao

January 27, 2019

1 Introduction

ROS (Robot Operating System) provides libraries and tools to help software developers create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more. ROS is licensed under an open source, BSD license (`http://wiki.ros.org/`). In this exercise, our goal is to get familiar with the ROS software tool.

2 Identifying the published and subscribed topics

To find the name of topics, we used the command "rostopic list -v". The obtained output is given as:

1. Published topics:
    - /turtle1/color_sensor [turtlesim/Color] 1 publisher
    - /rosout [rosgraph_msgs/Log] 1 publisher
    - /rosout_agg [rosgraph_msgs/Log] 1 publisher
    - /turtle1/pose [turtlesim/Pose] 1 publisher
2. Subscribed topics:
    - /turtle1/cmd_vel [geometry_msgs/Twist] 1 subscriber
    - /rosout [rosgraph_msgs/Log] 1 subscriber

The name of the topic used to send velocity commands to the turtle is "/turtle1/cmd_vel" while the name of the topic where the turtle position is published at is "/turtle1/pose". The messages type of the first topic is "geometry_msgs/Twist" while of the second topic is "turtlesim/Pose". We obtained these using the command "rostopic type *topic_name*".

3 Code

As ROS is a new platform for us and that the code is in Python which requires a level of familiarity with the appropriate APIs (rospy, common_msgs, etc.), we searched for materials online and one was particularly very helpful: `http://wiki.ros.org/turtlesim/Tutorials/Go%20to%20Goal`. The additions to the code are as follow:

- Import the needed libraries to handle the message data types "Twist" and "Pose".
- Set the publisher on the topic "/turtle1/cmd_vel".
- Set the subscriber on the topic "/turtle1/pose" with a callback method.
- Check if there are default waypoint coordinates using "rospy.has_param()" and if so, read them using "rospy.get_param()".
- Update the x, y and angle data of the turtle through the callback method.
- Compute the Euclidean distance between the current turtle position and the target position (waypoint).
- If the distance is smaller than a tolerance value, the turtle is declared to have reached the target. Otherwise:
  - Compute the linear velocity in the x-axis and the angular velocity in the z-axis that moves the turtle closer to and in the direction of the target position (see Equation 1 and 2).
  - Publish the new linear and angular velocities as a message of type "Twist".

$$v^* = K_v\sqrt{(x^* - x)^2 + (y^* - y)^2} \tag{1}$$

where v^* is the linear velocity, K_v is a constant, x^* and y^* represents the target point coordinates and x and y represents current point coordinates

$$\theta^* = tan^{-1}\frac{y^* - y}{x^* - x} \tag{2}$$

where θ^* is the steering angle

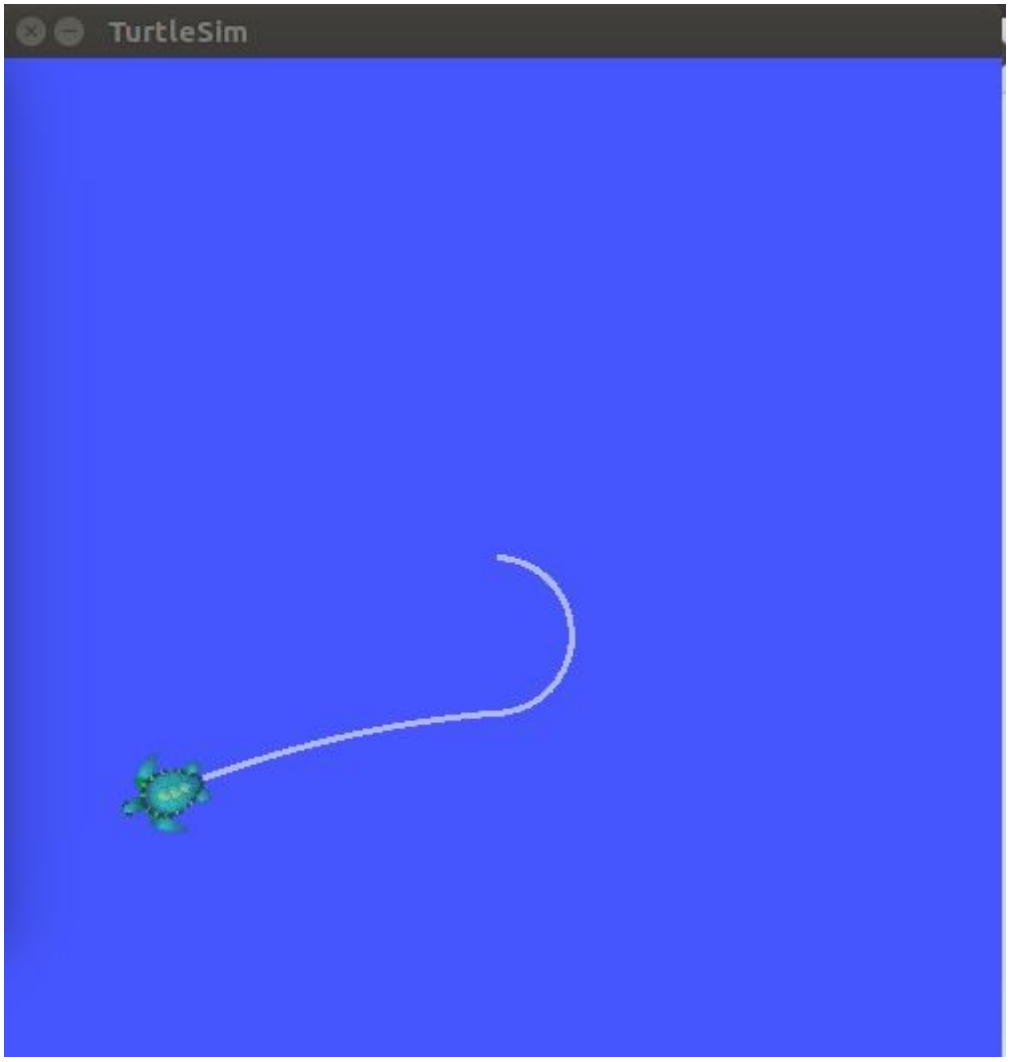
4 Examples of Program Execution

4.1 Waypoint Specified

Command:

```
hp4540@ubuntu:~$ rosrun lab0_ros turtle_waypoint.py 2 3
Heading to: 2.00, 3.00
Waypoint reached
```

Output:



4.2 Default Waypoint

Command:

```
hp4540@ubuntu:~$ rosrun lab0_ros turtle_waypoint.py
No waypoint specified in commandline
Waypoint found in param server
Heading to: 5.00, 10.00
Waypoint reached
```

Output: