CSE461 Lab Report 03
Fall 23

Title

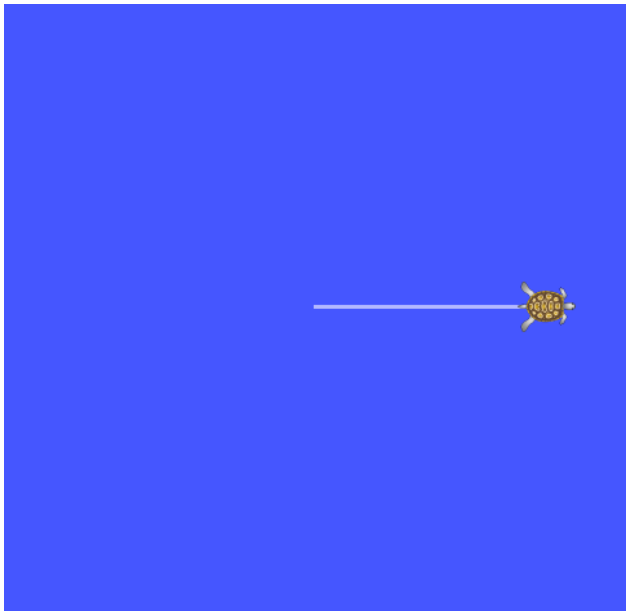
Turtle Movement Simulation Using ROS, Turtlebot & Python

Name: ANIKA ISLAM
ID: 21101298
Section: 09

Procedure

(1) ROS and Turtlebot packages are installed in a virtual machine.
(2) Roscore is run in a terminal to boot the master node.
(3) Rosrun turtlesim turtlesim_node is run in a terminal to pop up a window with a turtle at rest.
(4) In a different terminal, bash is run together with the move.py in the src folder of turtlesim_cleaner (created during turtle installation).
(5) User input is asked in the terminal using the code in the move.py file.
(6) Speed, distance moved (at max 5) and direction (1 for forward, 0 for backward) can be controlled.
(7) Turtle moves backward and forward using the given input.
(8) If input of distance moved is more than 5, then the turtle stops and a warning pops up on the terminal where the turtlesim_node is running.
(9) Turtle color changes every time rosrun is run. But, the behavior of the turtle remains unchanged.

Simulation Image



Code

```
#!/usr/bin/python3

import rospy # Communication

from geometry_msgs.msg import Twist # Message: position, angle etc



def move():

  # Starts a new node

  rospy.init_node('robot_cleaner', anonymous=True)

  velocity_publisher = rospy.Publisher('/turtle1/cmd_vel', Twist,
queue_size=10)

  vel_msg = Twist()



  #Receiveing the user's input

  print("Let's move your robot")

  speed = input("Input your speed:") # Say 1

  distance = input("Type your distance:") # Say 1

  isForward = input("Foward?: ")#True or False (0/1)

  speed = float(speed)

  distance = float(distance)



  isForward = int(isForward)



  #Checking if the movement is forward or backwards

  if(isForward):

      vel_msg.linear.x = abs(speed)
```

```
else:
    vel_msg.linear.x = -abs(speed)
#Since we are moving just in x-axis
vel_msg.linear.y = 0
vel_msg.linear.z = 0
vel_msg.angular.x = 0
vel_msg.angular.y = 0
vel_msg.angular.z = 0


while not rospy.is_shutdown():


    #Setting the current time for distance calculus
    t0 = rospy.Time.now().to_sec()
    current_distance = 0


    #Loop to move the turtle in an specified distance
    while(current_distance < distance):
        #Publish the velocity
        velocity_publisher.publish(vel_msg)
        #Takes actual time to velocity calculus
        t1=rospy.Time.now().to_sec()
        #Calculates distancePoseStamped
        current_distance= speed*(t1-t0)
    #After the loop, stops the robot
    vel_msg.linear.x = 0
```

```
      #Force the robot to stop
      velocity_publisher.publish(vel_msg)


if __name__ == '__main__':
  try:
      #Testing our function
      move()
  except rospy.ROSInterruptException: pass
```

Discussion

Through this lab task, we have learned how to simulate the movement of turtle using ROS, Turtlebot and python code. It is observed that the movement of the turtle is controlled by limiting the distance traveled using the user input and executed by the python code. Some difficulties such as ubuntu installation from the given zip file and booting it in the virtual machine were faced. By closing all the background windows, solved the ubuntu booting problem. However, no difficulties were encountered during the simulation process.

Video Link

How to move TurtleBot forward and backward ? | ROS | Python | Turtlesim