COSC81/181 – Robotics – Fall 2019

Programming assignment 1 (7% over the final grade)

The purpose of this assignment is to introduce you to the Robot Operating System (ROS) software platform, and to get you familiar with writing software that interacts with ROS, as well as applying the forward kinematics model and transformations we discussed in class.

Instructions

Please read carefully the following tasks and write the program(s) accordingly. *You should do this assignment individually.*

The instructions on how to submit the project assignment is on Canvas, at the following link

https://canvas.dartmouth.edu/files/5616794/download?download_frd=1
Please look also at the general guidelines for writing code:

https://canvas.dartmouth.edu/files/5585034/download?download_frd=1

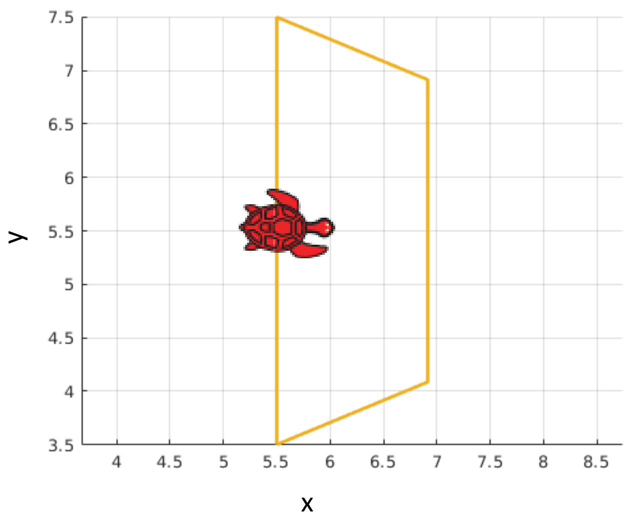
Getting ready

Please revise the material we went over during the first and second X-hours on ROS, how to write a program, how to create a ROS package. You can find the files used as well as additional readings on the description of each day in the calendar.

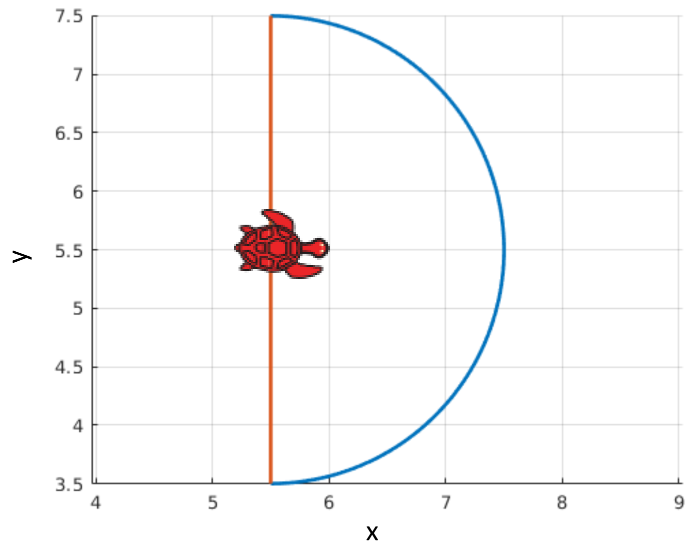
The task

Create a ROS package and write a simple ROS node that:

1. Will drive the robot to draw a simplified "D" represented as an isosceles trapezoid. The robot starts at the center of the longer base, with the robot local frame x-axis perpendicular to it and the vertices of the shorter base vertices are determined by the vectors of radius r and angle +/- 45 degrees. For example, in the figure below, the robot is at (5.5,5.5) m, with theta=0, and the radius is 2 m.



2. Instead of the trapezoid, will drive the robot to follow a D where the right stroke is represented with a half circle centered at where robot started with a radius r (e.g., 2 m in the figure).



3. Generalize the code to drive the robot to draw general arbitrary polygonal shapes. The coordinates of the vertices of such a polygon are given as input (represented as a polyline) in the world reference frame (hint: get the information about the robot pose from the topic `/turtle1/pose`). Print out in the terminal the points of that polyline in a local reference frame, where the origin is on the first point of the list, and the x-axis is aligned with the line segment determined by the first two points. Print also the 2D homogeneous transformation matrix in 2D between the two frames (from local to global).

To test your ROS node use `rosrun turtlesim turtlesim_node` to run the simulator.

Comments

Please see the course website for details on how to submit your project; here some additional reminders:

1. Show the math in the report.
2. Attach figures of your results.

Evaluation

Your programs will be evaluated based on both their functionality and their coding style. In the notes for writing programs, you can find an informal style guide to help give you an idea of what is expected together with the coding style that you should follow. In particular, the following aspects are considered:

ROS usage (20):

- Submitted file contains a well-formed ROS package.
- Package is named correctly.
- Package dependencies are correct.
- Package is configured correctly to build executable.

ROS Correctness (20):

- Becomes a ROS node correctly.
- Subscribes to correct topic and processes callbacks appropriately.
- Publishes to correct topic.
- Publishes messages of the correct type.

Performance (40):

- Correct behavior for each shape.
- Correct math.

Style (10):

- No duplication of executable code?
- No magic numbers?
- Names match functionality?
- Adequate comments?
- Comments match code?
- Consistent formatting?

Documentation (10):

- Report is complete and clear.
- Required sections exist under readily identifiable headings.
- Free of typos and grammatical errors.