EECE 5550 Mobile Robotics - Fall 2019
Homework Assignment 1
Due: Wednesday, September 25, 10:00pm

The goal of this homework is to make you familiar with ROS programming. This is pursued through implementation of the trajectory tracker presented in the class.

Problem definition

We have a tri-cycle with differential rear wheels and a front castor wheel. The goal is to implement a trajectory tracker to follow a given trajectory. The inertial frame is defined as $o_0x_0y_0z_0$ and the robot frame is defined as $o_1x_1y_1z_1$. The robot configuration vector is defined as $q(t) = [x \quad y \quad \theta]^T$ where x, y, and θ are elements of robot position in the inertial frame. At time $t = 0$ the robot is located at $q(0) = [x_0 \quad y_0 \quad \theta_0]^T$. The goal is to asymptotically follow the desired trajectory which is given as $r = \{(x_d(t), y_d(t))|t = 0,1,2, \dots\}$. The desired robot configuration is formed based on the trajectory as $q_d(t) = [x_d(t) \quad y_d(t) \quad \theta_d(t)]^T$. Design and implement a controller in ROS that gets the trajectory r as an input parameter and controls the robot so it converges to the desired trajectory and follow it. The results should be demonstrated using the `turtlesim`[1] simulator in ROS.

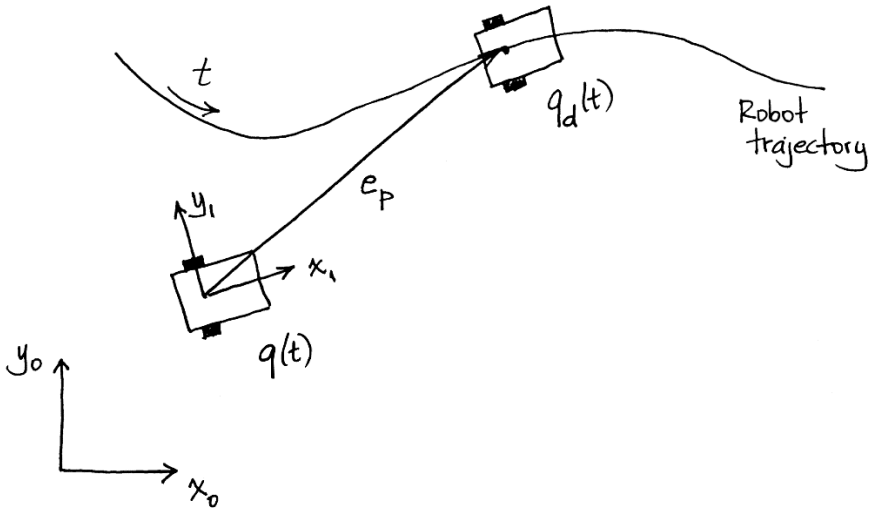

Figure 1 – Illustration of the mobile robot on a 2D plane. e_p indicates the error vector between current robot state and desired robot state.

Follow these steps to form a solution:

1. Create a catkin workspace and initialize it.
2. Create a package with `rospy` dependency and name it "`traj_tracker`".
3. Create a python node named "`scripts/tracker.py`" and implement the basics of a python node structure.
4. Run `catkin_make` for your catkin workspace and source the relevant `setup` file. Make sure that your catkin workspace is visible to ROS by checking `rospack profile` command output. Try running your node and make sure there are no issues.

[1] http://wiki.ros.org/turtlesim

5. Create a YAML file which includes the information for the input trajectory. Name the trajectory "`config/traj_1.yaml`" in your project. The format of the YAML file[2] should look like:

```
list_of_x: [x_d0, x_d1, ..., x_dn]
list_of_y: [y_d0, y_d1, ..., y_dn]
timestep: dt
```

where x_{di} and y_{di} with $i = 1, \dots, N$ are the desired trajectory points. Also, dt is the timestep for the desired trajectory. Populate the `list_of_x` and `list_of_y` for a square trajectory. The global coordinates of the square vertices are $\{(3,3), (8,3), (8,8), (3,8)\}$. The trajectory should be 10 times a single square trajectory. The overall duration of the trajectory (for all the 10 times of tracking) should be 60 seconds.

6. Create a launch file[3] named "`all.launch`" which reads "`config/traj_1.yaml`" into a parameter named "`trajectory_description`" and runs the "`scripts/tracker.py`" node.
7. Modify your node to read the "`trajectory_description`" parameter and print it on the screen, then exit. Run the launch file and make sure there no issues up to this point.
8. Modify the launch file to run the `turtlesim` simulator as well. This simulator is in `turtlesim` package and is named `turtlesim_node`. Run the launch file and make sure it works.
9. Use `rostopic list` to find the topic that is used for publishing input velocity messages for the turtle simulator.
10. Make your python node populate *random* messages into the topic you just found. Run the launch file, and everything should work fine together.
11. Implement the trajectory tracker and publish control inputs to control robot trajectory. To get robot pose as feedback you should subscribe to the corresponding topic.
12. Use the error norm to set the color of the turtle track using the `set_pen` service of the `turtlesim`. The color should range from pure red when turtle is far from desired trajectory to pure white when exactly following the desired trajectory.
13. Play with the controller parameters and see the results. Choose the best controller parameters.

Deliverables:

Create a zip file of all of the deliverables and upload them on Blackboard. The name should be:

"`<last_name>_<first_name>_<student_id>_hw1.zip`"

1. A two-page report which describes the project, how the source is run, what it does exactly, etc.
2. The project source files. Compress it and name it "`source.tar.gz`".
3. Simulation results for 5 different starting points (center, top-left corner, top-right corner, bottom-left corner, bottom-right corner). For the top-left corner as the starting point, also do the simulations for 3 different controller parameters (5+3=8 simulations overall).
4. A video of the robot doing all the above simulations[4]. Names should be set as "`<start_pos>.mp4`" for the first 5 simulations where "`<start_pos>`" is a placeholder for the starting position and "`param_set_<N>.mp4`" for the last three simulations. Define the parameter sets you have used in your report.
5. Plots of the robot configuration (x, y, θ) vs time for all the above simulations. Use the naming conventions of video files.
6. **(Bonus: +10%)** Create a trajectory of the course initials (MR for Mobile Robotic). There should be space between M and R letters and the robot should set the color off during the transition to R after finishing M (using `set_pen` service). The size of the M & R letters should be so that they fill the whole scene with a margin of 1 from all edges.

---

[2] Visit the following for a reference for YAML file specifications: https://roboticsbackend.com/ros-param-yaml-format/

[3] As an example of a launch file you can see: https://riptutorial.com/ros/example/24423/launch-ros-nodes-and-load-parameters-from-yaml-file

[4] You may use Kazam to record your screen in Ubuntu: https://www.linuxbabe.com/multimedia/install-kazam-screencaster

The overall time for the trajectory should be no more than 20 seconds. The robot's initial position should be set to the beginning of the letter M. Provide the video recording of the tracking.

References:

1. Several online code bases and tutorials that might be useful:
    a. http://wiki.ros.org/turtlesim/Tutorials/Go%20to%20Goal
    b. https://github.com/BlakeStrebel/Figure-Eight
    c. https://sceweb.uhcl.edu/harman/A_CRS_ROS_I_SEMINAR/3_1ROS_I_Seminar_TurtlesimControl_1_23_2018a.pdf
2. Cheat-sheets:
    a. [ROS] https://w3.cs.jmu.edu/spragunr/CS354/handouts/ROSCheatsheet.pdf
    b. [TMUX] https://tmuxcheatsheet.com/

Appendix

A docker container will be used to evaluate your results. The docker simply has ROS Kinetic installed together with some basic packages. Your package may be tested using this docker container for assessment. It is advised to test your packages in the provided docker container to make sure it works. It is simple to bring-up the docker container; just follow the instructions at https://github.com/sharif1093/eece5550_master.