ELEE 4200/5200: Autonomous Mobility Robotics
Term I, 2018
Homework 6: Position Estimation with Odometry Model

Guidelines:

- Due date: Thursday, October 25, 2018 by 12 Noon.
- Each group of no more than two students must work on its own in completing this assignment! Feel free to consult with others (and with me and the TAs) in developing solution ideas, but the final implemented code must be your work product alone. Refer to the Syllabus, where the policy on academic integrity is clearly outlined, our classroom discussion on this topic, and consult with me if you have any questions!
- State the full names and T# of the students in the group on the cover page of every document that you submit.
- Submit the report by responding to this assignment posting in Blackboard.
- The submission should at least include the following documents, bundled together into a single zip file with the name *YourNameHW6* (use one of the group member names).
  - The main report (following the template provided).
  - The main report in 'pdf' form.
  - The MATLAB program code.
- A hard copy (printout) of the 'pdf' report with MATLAB code; staple all pages together and follow the TA's instructions on how to submit.

Goals:

- To investigate the position model based on odometry from wheel encoder data for robot localization (localization = estimating the pose of the robot).
- To do the above in two ways:
  - Using the "exact" circle equations for robot pose;
  - Using the Borenstein approximation equations to estimate robot pose.
- To compare the two sets of results.
- To understand the limitations (errors) associated with this approach.

Theoretical Background:

A differentially-steered robot with L= 0.2 m is given the following drive command: v= 0.5 m/s & $\omega = 0.25\ r/s$. However, the drive command is an input and the actual position of the robot is measured using wheel encoder readings every 0.05 m.

The nominal (baseline) left and right wheel encoder readings can be calculated by solving the equations from Slide 12 in Reference [1], repeated below for convenience:

$$s_M = \frac{s_R + s_L}{2};\ \theta = \frac{s_R - s_L}{2L}$$

The values of s_R & s_L that you solve for from the above equations, which provide *relative* motion with respect to the previous position, represent the ideal case. We will model the real world by adding Gaussian noise to s_R & s_L, with the following characteristics – mean of zero and variance of 0.005 m, with any individual encoder noise value capped at +/- 0.01 m. Remember that each successive encoder reading is based on the previous noisy encoder reading and not on the base reading.

With the resultant encoder readings obtained above, you can estimate the robot location using the ideal equation as well as the Borenstein approximation (as on Slide 9 [1] with 's' replacing 'v' & Slide 14 [1]), repeated below for convenience:

Ideal:

$$x(t) = x_0 + L\frac{s_R + s_L}{s_R - s_L}\left[sin\left(\frac{(s_R - s_L)}{2L} + \theta_0\right) - sin(\theta_0)\right]$$

$$y(t) = y_0 - L\frac{s_R + s_L}{s_R - s_L}\left[cos\left(\frac{(s_R - s_L)}{2L} + \theta_0\right) - cos(\theta_0)\right]$$

Borenstein approximation:

$$x_i = s_M cos(\theta) + x_{i-1};\ \theta: pose\ angle$$

$$y_i = s_M sin(\theta) + y_{i-1};\ \theta: pose\ angle$$

Specific Tasks:

a) Using MATLAB, plot the ideal pose of the robot, assuming no noise, over one full circle.
b) Plot the pose of the robot, based on the noisy wheel encoder readings [2] and the "exact" position model.
c) Use the same noisy encoder readings as in part (b) and repeat, this time using the Borenstein approximation. Note: Using the *same* encoder readings ensures that the same noisy values are being used for both methods.
d) Repeat parts (b) and (c), starting the robot at the same initial location, but re-seeding the random number generator.
e) Compare and comment.

Other information:

- Be sure to apply the initial conditions appropriately in determining new pose!

References

1. Power point class notes: "From Velocity to Position"
2. Generating random numbers in MATLAB: https://www.mathworks.com/help/matlab/random-number-generation.html