ELEE 4200/5200: Autonomous Mobility Robotics
Term I, 2019
Homework 4: Position Estimation with Odometry

Guidelines:

- Due date: Thursday, October 10, 2019 by 12 Noon.
- You are permitted to work in groups of no more than two students. State the full names and T# of the students in the group on the cover page of every document that you submit. Only one report needs to be submitted per group!
- 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 *YourNameHW4* (use one of the group member's names).
    - The main report in 'pdf' form (using the LaTeX template provided).
    - The MATLAB program code (as an m-file, so that it can be executed!).
    - Support documents like videos, etc. No videos are required for this assignment!
- A separate hard copy (printout) of the 'pdf' report with MATLAB code listing; staple all pages together and follow the TA's instructions on how to submit the hardcopy.
- Each group must work on its own in completing this assignment! Feel free to consult with others in developing solution ideas, but the final code implemented must be your work product alone. Refer to the Syllabus, where the policy on academic integrity is clearly outlined, our classroom discussions on this topic, and consult with me and/or the TAs if you have any questions!

Goals:

- To investigate position models for robot localization based on odometry using wheel encoder data. Localization = Estimation of (x, y, θ).
- To do the above in two ways:
    - Using the *exact* circle equations;
    - Using the Borenstein approximation.
- To compare the two sets of results.
- To understand the limitations (errors) associated with the odometry approach from the above experiments.

Theoretical Background:

A differentially-steered robot with L= 0.25 m is given the following drive commands: v= 0.4 m/s & $\omega = -0.2\ r/s$. However, the drive command is an input, and so the actual position of the robot is measured using wheel encoder readings every 0.05 m, after the motion has occurred.

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 0 m and variance of 25 X 10^{-6} m^2, 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 (circle) equation as well as the Borenstein approximation (as on Slide 9 [1] with 's' replacing 'v' & Slide 14 [1]), repeated below for convenience:

Ideal (circle) equations:

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

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

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) First, understand the equations thoroughly!
b) Using MATLAB, plot the ideal pose of the robot, assuming no noise, over one full circle.
c) Plot the pose of the robot, based on the noisy wheel encoder readings [2] and the "exact" position model.
d) Use the same noisy encoder readings as in part (c) 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.
e) Repeat parts (c) and (d), starting the robot at the same initial location, but re-seeding the random number generator.
f) Compare and comment on the above experiments.

Note:

- When you do odometry, each point is constructed from the previous point and not from the starting point!

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

In the next iteration:

- Consider introducing wheel slip; first left wheel, then right wheel, by the same amounts, to show that they don't compensate for each other. Emphasizes the non-linearity of the odometry model.
- Also, very important – check the numbers of the problem! Student claims that the noise limits are so restrictive, that it negates the Gaussian distribution assumption for the odometry error.