Day 3: Encoders, Mapping, and Intro to the Bridge of Death

Quantitative Engineering Analysis

Spring 2019

1 Schedule

- 0900-0915: Quiz
- 0915-0945: Debrief and Synthesis
- 0945-1030: Measured Paths
- 1030-1045: Coffee
- 1045-1215: Encoders in Action
- 1215-1230: Preview of the Overnight

2 Quiz [15 minutes]

For this quiz, consider the following parametric curve,

$$\mathbf{r}(u) = a\cos u\hat{\mathbf{i}} + a\sin u\hat{\mathbf{j}},$$

where u is a function of time t,

$$u = bt^2,$$

and a and b are positive constants.

1. If your robot drives this curve, it will:
   (a) Drive an ellipse at constant speed.
   (b) Drive upward in a spiral.
   (c) Drive in a circle at increasing speed.
   (d) Spin in place.
   (e) Do something not listed above.
2. The linear velocity $\frac{d\mathbf{r}}{dt}$ of the robot can be expressed using the chain rule as:
   (a) $\frac{\partial \mathbf{r}}{\partial a}\frac{\partial a}{\partial t}+\frac{\partial \mathbf{r}}{\partial b}\frac{\partial b}{\partial t}$
   (b) $\frac{\partial \mathbf{r}}{\partial u}\frac{\partial u}{\partial t}$
   (c) $\frac{\partial \mathbf{r}}{\partial u}\frac{\partial t}{\partial u}$
   (d) None of the above.

3. What is the linear speed?
   (a) $2bt$
   (b) $2abt$
   (c) ab
   (d) None of the above.
4. What is the angular speed?
   (a) $2bt$
   (b) $2ab$
   (c) $2at$
   (d) $\hat{\mathbf{k}}$
   (e) None of the above.

3 *Debrief and Synthesis [30 minutes]*

Exercise (1) Briefly discuss the overnight with your table-mates, and make a list of concepts you feel solid on, and concepts you feel shaky on. Make a list of important definitions that you encountered in the overnight.

Exercise (2) During class and in the overnight exercises we have been building capacity towards having the NEATO robot drive along a curve. Of course there are lots of different ways to parameterize a curve, each of which would correspond to different motion of the NEATO. Let's take a few minutes to review those concepts here. With your table, please answer the following questions on the board given a circle of radius R,

$$\mathbf{r}(u) = R\cos u\hat{\mathbf{i}} + R\sin u\hat{\mathbf{j}},\ u \in [0, 2\pi]$$

and the following parameterizations:

$$\begin{aligned} u(t) &= \alpha t \\ u(t) &= \alpha t^2 \\ u(t) &= \alpha(2 + sin(t))t \end{aligned}$$

(a) Qualitatively describe the motion of the NEATO in each case.

(b) How long does it take to traverse the curve once? (Your answer should depend on α)

(c) How would you find the linear velocity of a robot traveling along the curve? What is the direction of this velocity? (It would be really great if you employed the chain rule here and kept your work as general as possible.)

(d) How would you find the angular velocity? What is the direction of the angular velocity? (It would be really great if you employed the chain rule here and kept your work as general as possible.)

(e) Having found the linear and angular velocity, how would you find the left and right wheel speeds for the differential drive?

(f) The robot has a maximum wheel speed of 0.3 m/s. How would you choose α to ensure your robot never exceeds this speed limit?

4 *Measured Paths [45 minutes]*

One potential source of error that you may have identified in the in class and overnight exercises is that your robot is not able to instantaneously achieve a desired V_L and V_R when you send it a particular motor command. Given the pesky laws of physics, instead, the robot needs to accelerate to the desired velocity. In order to get a more accurate picture of what the robot actually did, we can use *measurements* of the wheel velocities to give us a more accurate estimate of the robot's actual path in the world. Our Neato is outfitted with sensors called *wheel encoders*, which provide accurate estimates of the linear travel of each wheel over time. Knowing the linear travel and the time between measurements, the velocity of each wheel can be calculated. Next, you'll be determining formulas to update the robot's position and heading given measured values of V_L and V_R.

Exercise (3) Suppose that at time t your robot is at position $\mathbf{r}(t)$, with a heading of $\theta(t)$. Let's further assume that at $t = 0$ the robot is stationary and pointing along $\theta = 0$, which corresponds to the robot facing along the positive x-axis of the room.

(a) Draw a picture on the board to make sure that you are clear as to the definition of the coordinate system.

(b) Sketch a (fairly smooth) potential trajectory that your robot will be driving. Choose several points along the curve seperated by Δt. Sketch in the unit tangent and unit normal vector at each point.

(c) Sketch an estimated plot of your robot's linear and angular speeds as a function of time as it traverses your curve.

(d) Sketch an estimated plot of your robot's tangential and normal components of acceleration as a function of time as it traverses your curve.

(e) On the board, sketch a qualitative version of your estimate of right and left wheel velocity as a function of time.

For a discretized path (expressed in terms of short time increments rather than continuously) you can, assuming that the time-step Δt is small, approximate a path by a series of movements in $\mathbf{r}$ and movements about the center of the robot in θ. The velocity of the robot is

$$\begin{aligned} \frac{d\mathbf{r}}{dt} &= v\hat{\mathbf{T}} \\ \frac{d\theta}{dt} &= \omega \end{aligned}$$

where v is the linear speed, ω is the angular velocity in the $\hat{\mathbf{k}}$ direction, and since the robot is always oriented along the path we can define $\hat{\mathbf{T}}$ in relation to the global (classroom fixed) coordinate system as

$$\hat{\mathbf{T}} = \cos\theta\hat{\mathbf{i}} + \sin\theta\hat{\mathbf{j}}$$

(f) Given measured values for V_L and V_R determine the values of $\mathbf{r}(t + \Delta t)$, and $\theta(t + \Delta t)$ which represent the position and heading of your robot at time $t + \Delta t$.

5 *Coffee Break [15 minutes]*

6 *Encoders in Action [90 minutes]*

Next, you will complete a series of simple experiments with the NEATO, similar to those conducted at the end of Monday's class, while simultaneously collecting motion data. In order to do this, we have provided you a nice little script written by Paul Ruvolo (edited by Jeff Dusek) which you can use to collect the wheel position encoder data while you are running your experiment (which you can easily convert to velocities by taking the difference between two adjacent positions and dividing it by the timestep). The Matlab function *diff* will likely be useful in this process. Note that the robot's initial position is arbitrary.

The script for collecting encoder data is called collectDataset.m and is linked here and to the Canvas assignment. It is also available from the sample code page. To collect encoder data, run the function *collectDataset('filename.mat')* from your command window. This will bring up a new figure window with the title "Dataset Collection Window". To start data collection , hit the space bar while focusing on the figure window. You will see the message "Starting Dataset Collection" in your command window if everything is working as intended. You can then run your personal script to control the robot, and encoder data will be collected in the background. When your robot motion has concluded, re-focus on the "Dataset Collection Window", and hit the space bar to stop data collection. You will see the message "Stopping Dataset Collection" in your command window.

After you stop the data collection, you will have a file *filename.mat* in your current directory. If you load this file, you will find a matrix "dataset" that contains the encoder and accelerometer data recorded from the robot. For this exercise and the upcoming challenge you only care about the encoder data in columns 2 and 3, and the time stamps in column 1 (recall this data is linear travel of the wheel). The form of the data is:

$$dataset = [time, Pos_{left}, Pos_{right}, AccelX, AccelY, AccelZ] \qquad (1)$$

Important: If you include a loop in your personal robot control script, make sure to include a pause of the form *pause(0.1)* within the loop. Otherwise, Matlab will try to execute that loop as fast as possible and will prevent the data collection script from recording encoder data.

Complete the following simple experiments. For each experiment, record the encoder data for analysis using the *collectDataset* function.

Exercise (4) Using the basic robot motion code from Monday's class (copied below), have the robot complete three complete counterclockwise and three complete clockwise circles around the robot's center (i.e V_R=-V_L or the opposite). Collect and plot the left and right wheel encoder data.

(a) Does the plot of the wheel linear travel look as you would expect?

(b) Find the linear and angular velocity at for each time step and plot them. Do they match your expectations?

```
pub = rospublisher('/raw_vel');
msg = rosmessage(pub);
msg.Data = [V_R, V_L];
send(pub, msg);

prompt = 'Press Enter to Stop Robot';
str = input(prompt,'s');
if isempty(str)
    msg.Data=[0,0];
    send(pub,msg)
end
```

Exercise (5) Using the driveforward.m function introduced in the overnight, conduct three experiments where the robot drives a specified distance at increasing speeds. Collect and plot the encoder data.

(a) Does the linear distance traveled collected by the encoders match the distance input to the function?

(b) If the values do not match, why not?

(c) Plot the linear and angular velocity as a function of time. Do they match your expectations?

Exercise (6) Calculate the left and right wheel velocities needed to drive a circle of radius 0.5m in 20s. Use the code from Monday's class (above) to drive the circle while simultaneously collecting the wheel encoder data.

(a) Plot the anticipated trajectory of your robot in Matlab. You may want to use the Matlab command *axis EQUAL* to make sure your circle looks nice and circular.

(b) In the same figure, plot the actual trajectory of your robot. How closely do they match?

(c) Quantifying error is an important part of any experiment. For the circular path, how would you calculate error between the anticipated trajectory and the actual path of the robot? Are you interested in total accumulated error? Average error and each time step? Distance away from the "ideal" curve at each time stop (e.g. are you stying within a lane)? Does something else make sense? On the whiteboard draw sketches of what each of these types of error would represent visually, and come up with mathematical expressions.

6.1 Extension: Encoders in Real-Time

You can also read the robot's wheel positions real-time from the */encoders* ROS topic and use this to plot the robot's motion real-time as it moves through the exercise...or even to correct for motion errors to bring it back closer to the desired path!

Exercise (7) Modify the driveforward.m function to use encoder feedback instead of time to determine when to stop the robot.

Exercise (8) Try writing a function that commands the NEATO to drive a square with the side length as an input variable. Think about using the encoder feedback to determine when to turn, and whether the robot has turned 90 degrees. The *'/cmd_vel'* Ros topic might be a good option here because it takes the linear and angular velocity as inputs. If the square is too easy, how about driving a star?

Exercise (9) Feedback from the encoders can be used to correct the robot's position if it strays from the planned trajectory. Think about how you could calculate error between the anticipated and actual path, and what actions would need to be takent to reduce that error.

7 *Preview of the Overnight [15 minutes]*

Introduction to the Bridge of Death challenge.