Assignment1:
Differential Driver and UMBmark Calibration
Due: Thurs, Feb 16, class time

10 points

Summary

You and your partner are to first build the standard Lego DifferentalDrive robot (TriBot) if you have not already done so. You will need to install Lejos on your laptop and on the SDcard for the EV3. Next you will write code to drive your robot over precise distances and turns. Lastly you will calibrate your driving code to adjust for systematic errors.

Differential Driver

You are to create a Lejos class called DiffDriver that can drive the TriBot forward a specified distance and allow it to make a sweeping turn around a given point. Note that Lejos comes with a built-in pilot class, but we will not be using it for this assignment. We will be writing our own differential driver class using the EV3LargeRegulatedMotor class to do the low-level control of the wheels. Specifically you will be using the rotate, waitComplete, setSpeed and perhaps setAcceleration methods.

You should have a constructor that creates a driver for your robot layout:

DiffDriver(int wheelBase, int wheelDiam)

Where both the wheelBase and the wheelDiam are in mm. These will never change so only passing them once into the constructor is the best way.

There should be a method that allows the robot to drive forward (or backwards) in a straight line:

Public void forward(int distance, int speed)

Where distance is the distance to cover in mm. A negative distance implies the robot has to drive backwards. The speed is given in degrees/second in terms of how fast the wheels should rotate.

And lastly there should be a method that allows the robot to do a sweeping turn around a pivot point:

Public void turn(int radius, int angle, Boolean leftTurn, int speed)

Radius is the distance in mm to the pivot point from the center of the robot. Angle is the amount of rotation to perform around that pivot point, given in degrees. leftTurn is a Boolean that indicates if the turn is to the left or the right. And speed is given in degrees/second in terms of how fast the robot should move. You may assume the radius and angle are always be positive. However, note that when the radius is smaller than half the wheelbase then the inner wheel will end up running backwards so you

might have to account for this in a special case. Similarly, if the turn is really a pivot where the radius is 0, you have to watch out for divide by 0 errors.

Calibration

Lastly, you need to run the UMBmark calibration. This involves writing a program that will run the robot around a square, either clockwise or counterclockwise. Ideally this will be a 1meter square. You will do this 5 times each direction and record how much the robot is off in the final position. You are to record these measurements (in mm) in an Excel spreadsheet. You are then to code the UMBmark calibration equations in this same spreadsheet so that it can automatically calculate the errors and produce the corrected values for both wheel base and wheel diameter. Your fields in your spreadsheet should be labeled so I can tell what values are what. I should be able to put in new error measurements and everything should automatically calculate from those values. You are also to produce a scatter plot of your values and their centers of gravity automatically from the data. You will need a starting wheel base and wheel diameter when you do the calibration. Use 56mm for the diameter (that is the value given on the wheels) but use a value for the wheel base that is 5 to 10mm larger than you think the wheel base is. This last part is to magnify the errors to better illustrate the calibration process.

Useful links for UMBmark are:

The slides I used in class to describe it: http://mrl.engin.umich.edu/31UMBm.html

The 25 page paper published in IEEE Transactions on Robotics and Automation: http://www-personal.umich.edu/~johannb/Papers/paper58.pdf

The full 77 page technical report: http://www-personal.umich.edu/~johannb/Papers/umbmark.pdf