Lab 5: Odometry for a Roving Bot

Lab objectives

1. In this lab you will design an odometry system for the Tribot. In particular, you will explore the accuracy of measuring position through dead reckoning. You will also implement simple control commands such as drive forward, turn, etc, which will be useful later in the project.

Step By Step

1. The Tribot has no way of absolutely determining its position or heading. Therefore we must design a way to estimate its position and heading by observing the number of degrees rotated by each wheel. Measure the wheels of the Tribot, and the distance between them, and develop a method for computing the robots heading, and the position of its center from the movement of the wheels. You can assume that the robot starts out pointing in direction 0, at x, y position 0,0.
2. Implement your odometry system using NXC. You may use the program odometryshell.nxc as a starting point.
3. Create macros or functions to
    a. Rotate the robot to a specified orientation
    b. Drive the robot a specified distance forward or backward

    These functions should use the odometer to control the robot. They should not reset the odometer – that is, if you say rotate to 20 degrees, followed by rotate to 30 degrees the robot will NOT rotate 50 degrees, but will rotate to 30 degrees from its starting position.
4. We will do an experiment to determine the accuracy of the Tribot's driving ability and your odometry algorithm. Using the functions created in the last step, make the Tribot drive straight ahead for 70cm, turn to a heading of 325 degrees, and drive straight ahead another 70 cm. Measure its X, Y position from its starting point, and compare to where it should be, and where the odometer says it is. Perform this experiment 5 times for each of 30%, 50% and 100% power. (The power does not have to be used for the turning, just for the driving forward)
   You should measure the position of the robot at its center point – i.e. Between the wheels, and the heading along the centerline. To make it easier to measure the position and heading of the robot, you may wish to attach pieces to make the center easy to see when the robot is on the floor, and extend something to make the centerline easy to line up with a protractor. You may also find the printable protractor available here http://www.ossmann.com/protractor/ useful.
5. Analyze your results looking for accuracy in two ways - how accurately did the robot drive, and how accurately does it know where it is. That is – the robot may not drive very straight, but your odometry may detect that very successfully, or vice versa.
6. Demonstrate your program to a TA. To demonstrate your macros/functions, the TA will specify a path for the robot to follow, you will implement it and show it. For example, the TA may say "Turn to a heading of 30 degrees, drive 50 cm, turn to a heading of 25 degrees, stop." There may be more than three steps in the request. This should be a few lines of code using your macros or functions.

To Hand In (60 marks)

1. Your analysis of how to compute the robots position. 5 marks
2. A proposed algorithm for computing the robots position 5 marks
3. Your code from step 3 10 marks for odometry, 2.5 marks each heading and drive 15 marks
4. The results of the experiment in step 4 5 marks
5. Discussion of your results (step 5). 10 marks
6. Demonstration is worth 20 marks