CS148
Building Intelligent Robots

Week 3 : Odometry with Rotation Sensors *Out: 27 Sept 2005*

Preliminary Tasks *before 29 Sept 2005, 9am*

The purpose of this week's lab is to familiarize you with the theory and practice of odometry, in preparation for your next project. You should should be able to use your code from this lab in the project.

For this week's lab, you will need to bring your robot with a rotation sensor attached or geared to each drive wheel (or tread). We recommend that you configure your robot to have two drive wheels and a third unpowered swivelling wheel. This will allow the robot to move more accurately and therefore provide better odometry data.

Rotation Sensors

You will be using rotation sensors (aka angle sensors) to measure the distance travled by your robot. To activate a rotation sensor, you use use `ds_active(&SENSOR_X)` and `ds_rotation_on(&SENSOR_X)`. These commands turn on the power to the sensor, and the brickOS processing of the input respectively. Once activated, the value of the light sensor can be read using `ROTATION_X`. This value will change every time the sensor turns 1/16 of a revolution.

0.1

The first step in odometry is to get your robot moving in a predictable manner. When driving in a straight line, your robot's rotation sensor readings should increase or decrease by the same amount on each wheel. Similarly, when turning in place, the readings should change by the same magnitude, but in the opposite direction. Your program will have to monitor the changing rotation sensor values, and adjust the power to the corresponding motors accordingly.

Task:

- Program your robot to move forward and backward in a straight line, using the rotation sensors to make sure the wheels turn the same distance. Make your robot drive forward and back the same distance so it ends up where it started.
- Program your robot to turn left and right in place. Again, use your rotation sensors to ensure that the robots wheels turn the same distance (in opposite directions). Program your robot to move forward, turn left, turn right, and then move backward, ending where it started. Experiment with the speed the robot moves to see its relationship to the robots consistency.

0.2

By now your robot should move consistently. You should be able to control howi many rotations its wheels take as it moves, but you don't know how far that is. To get meaningful odometry data, you will have to convert angle rotations into distance units such as centimeters. Your robot will probably have to do some rounding of its distance commmands, because it can't measure rotation's smaller than 1/16.

Task:

- Program your robot to move forward and backward specific distances. You may want to use a calibration phase, as you did in the line following lab, or you may use trial and error to determine how much distance your robot covers with 1/16 of a turn of a rotation sensor.
- Program your robot to turn left and right specific angles. Again, you may use a calibration phase, or trial and error.
- Program your robot drive a square path, going forward, and turning left. The square should be 50 cm on a side. The robot should drive around the square at least twice.

0.3

You may notice that after your robot does not end up where it started after a a few laps around its square path. This is in part due to the rounding meantioned in the last section.

Task:

- Change your odometry code to keep track of how far the robot has actually traveled, taking the rouding into account. Have your robot display its relative position after it goes around the square a few times. Assume your robot starts out facing the positive x position, and that left is the positive y position. The angle of your robot's orientation start as 0, and increase in the counter-clockwise direction.