CMPUT 499 Assignment 1 Report

CMPUT 499 Winter 2014

Doug Schneider - 1242106
Aaron Krebs - 1242107

2.

k. The drawn shapes were fairly accurate (IE the circle was quite round, the rectangle was rectangular, etc). However, reproducing the shapes consistently was difficult, as the shapes would be slightly different each time, or in slightly different positions even though the robot started in nearly the same spot. From this, we can conclude that the motors are not necessarily consistent. We can also conclude that friction differs across different surfaces. We can conclude these two facts because the motors require different power inputs in order to travel in a straight line. In addition, the differences in the shapes between runs can be explained by friction and imperfections in the surface. These result in the wheels spinning slightly different each time and resulting in a slightly different/imperfect shape on each iteration.

3. The dead reckoning position controller is implemented using formulae presented in the lab. Every 10ms we measure the number of rotations of each wheel, and using the wheel radius we can compute the distance travelled by each wheel, from which we can compute the change in heading of the robot as well as its change in X and Y position (starting from 0,0). The accumulated position and heading changes are displayed on the screen after each command.

4.

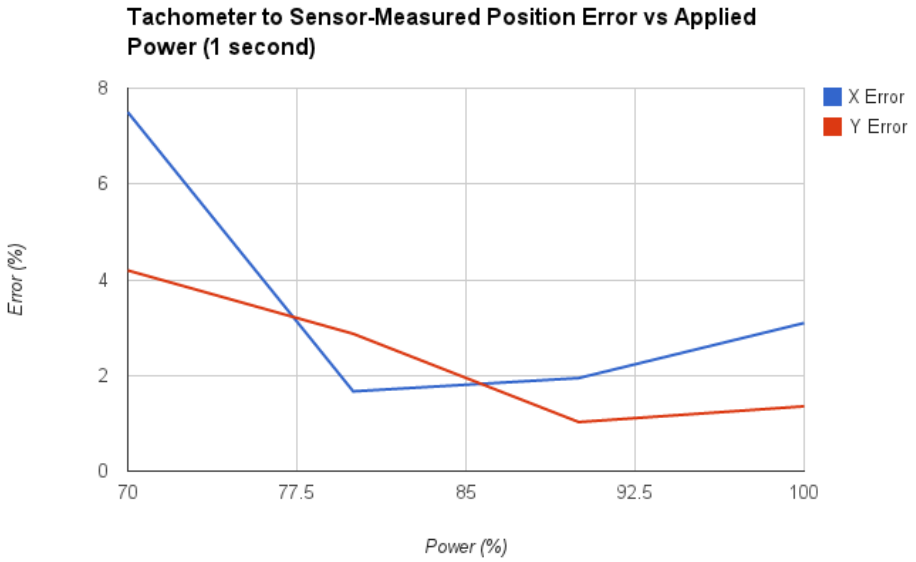
a. Our first method to measure straight-line distance is to measure the actual distance travelled using a ruler and compare this to the tachometer calculations. We have the robot travel in a straight line (applying power for 2 seconds). We ruler-measure the X and Y distance travelled and compare this with the tachometer-computed X and Y distance. The difference between the two measurements is the error. The error is normalized to the percentage of the ruler-measured distance (IE we are treating the ruler-measured distance as the actual, and seeing how much the tachometer differed). For example, if the robot is ruler-measured to have travelled 120mm, and the tachometer calculates 122mm, the error would be (|120-122|)/120 = 2/120 = 1.7%. This normalization allows us to compare errors for different travelled distances (since the robot travels further under 100% power than 70% power) and for applications of power for different amounts of time (this is more important for the second method, in

which we only applied power for 1 second to reduce the distance travelled, which made measurements easier).

The error in X and Y position measurement at various power levels for the first method is shown in this figure:

Our second method involves using two distance sensors. One distance sensor faces away from the robot, to the side. The second sensor faces forward. These two sensors are used to calculate the X and Y distance travelled, as a delta from the starting distances measured, and the final distance measured. The error is then calculated from the difference between the sensor-measured distances and the tachometer-computed differences. Similar to the first method, the errors are normalized against the actual distance travelled, which in this case is assumed to be the sensor measurements.

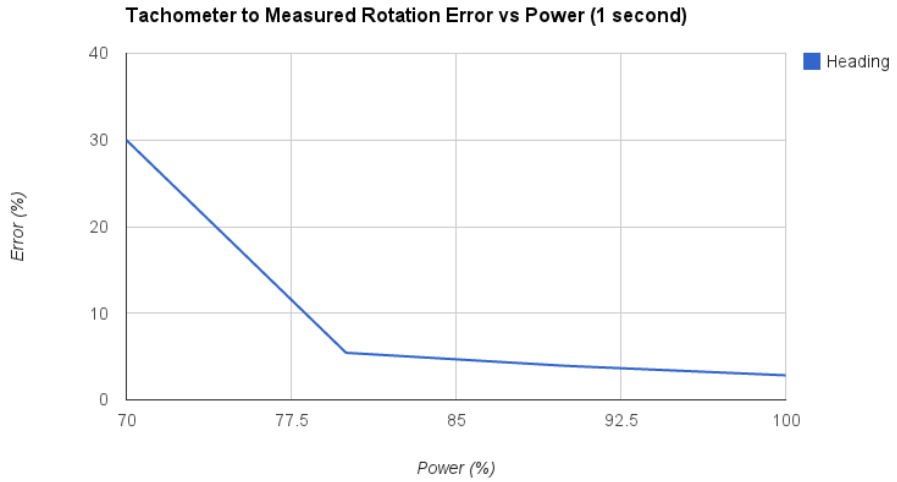
The error in sensor measurement at various power levels is shown in this figure:



When comparing the measured error between the two methods we found that method one had a higher error than method two for all motor powers. However since method one was a measure of the distance travelled using a ruler its measured error should be more accurate than the error measured using the sensors. There are three sources we can think of for the different error. The first is that one of the sensors we used for measuring the distance is only accurate down to the nearest centimetre. As a result the measured error using sensors isn't nearly as accurate as using a ruler, which is down to the nearest millimetre, as the second method has far fewer significant digits in the measurement. The second reason we can think of for error is imperfections in the sensors resulting in occasional noise values that off the measured error. The final source of error is that when the robot was moving in a line it tended to curve to the right. Since the distance sensor for the X axis was pointing to the right, when the robot curved it was no longer perpendicular to the wall. As a result the distance from the wall is no longer accurate. This is able to introduce substantial error in the sensors measurements. Which combined with the error of the tachometer, changes the overall error. In conclusion, method one is much more accurate, but ultimately not something that can be automated.

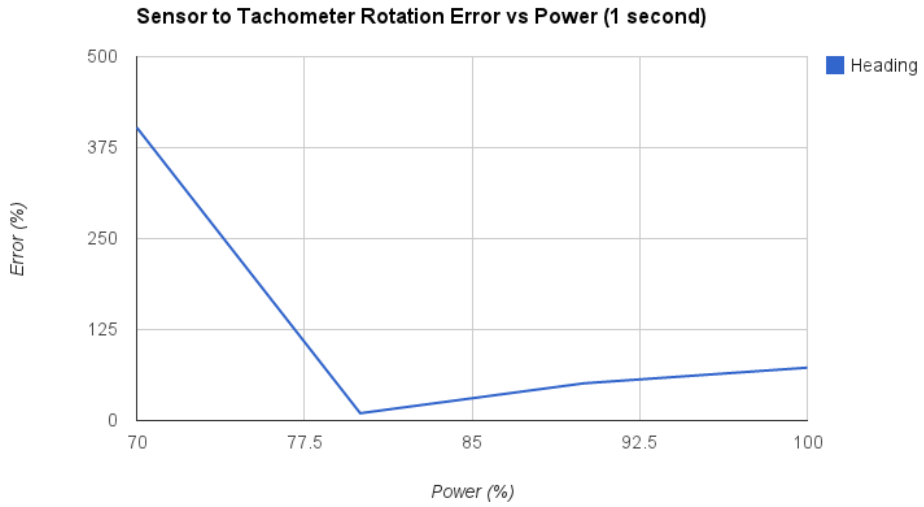
b. Our first method for measuring the error we measure the actual heading by using a protractor. We compare this heading to the tachometer-calculated heading. As in part a), the protractor measurement is taken as the actual heading and the tachometer heading as the computed value. Error is normalized against the actual heading.

Heading error for various power levels is shown in the following figure:



Our second method for measuring the rotational error uses two distance sensors. Both sensors are placed on the robot facing forward. They are placed near the location of the wheels in order to calculate the heading based around the robots axle. Both sensors are then used to measure the distance from the wall and are checked every 10 milliseconds. The distance values are then used to calculate the change in heading every 10 milliseconds. This calculation is very similar to calculating the heading using the tachometer, but instead of using the distance each wheel travelled to get the heading change at each time step, we use the change in distance sensor readings. All of the changes in heading are added up to calculate the final heading. This heading is then compared with the tachometer-computed heading to find the error (and again normalized, assuming the distance sensor heading is the actual heading).

Error at various power levels is shown in the figure below:



Between the two methods the error is significantly different. The error measured using a protractor is relatively small, while the error measured by the sensors is really large. This is expected for a few reasons. First of all, we expect the heading measured by the protractor to be very accurate. So the only source of error is in the heading measured from the tachometers. The distance sensors used in the second method on the other hand can also introduce error. Thus it's possible for the overall error to be larger due to two sources of error. We expected the second method to be less accurate but it was quite inaccurate. We believe that it is so inaccurate for a couple of reasons. First of all, the sensors may not have been directly aligned with the axle. This alone would result in an inaccurate measure of the changed heading. In addition, when travelling in a line, the robot curves. As a result, one of the sensors moves away from the wall, and one closer towards the wall. The problem is that because the sensors are no longer perpendicular with the wall. The measurements are offset. This can introduce a huge error in the changed heading measured via the two sensors.

Based on the measurements using a protractor we are able to determine that there is indeed an error in the heading measured using the tachometers. Interestingly, the error is a higher percentage for lower powers. This is consistent with the error measurements for part a), and we hypothesize that this is because friction within the motor becomes a larger factor if less power is supplied; the

number of ticks measured by the tachometer in each motor is apparently more accurate as more power is applied.

5. Our implementation for teleoperation of the robot is fairly simple. LeJOS PC projects allow commands to be issued to the robot while the java code is executed on the PC. Thus, we implemented a simple Swing GUI that uses a KeyListener to capture user key-presses. To control the robot, we execute code to power the motors in a forward direction when the forward key is pressed and stop the motor when the key is released. To allow full control, we bind the forward and backward direction of the left and right motor to 4 different keys. Thus, the remote operator can control the robot much like a tank, steering by moving the two motors in opposite directions.
   The primary problem we encountered with this solution is that the robot's motor tends to stutter even though the operator is pressing a key continuously. This is, we hypothesize, due to delay in the Bluetooth transmission of commands, or the KeyListener thinking a key is released when it is not (and then immediately seeing the key as pressed again but still issuing an intermittent stop command to the robot).
   Overall, remote operation of and data transmission to and from the robot seem really straight-forward, but the transmission delay is something to keep in mind. If, later in the course, we intend to use the PC to do complex computations it will be necessary to recall that the data transmission rate may still be a bottleneck.