|  |  |
| --- | --- |
|  | Rose-Hulman  Institute of Technology |

Memo

To: Dr. Carlotta Berry

From: Ander A Solorzano \_\_\_\_\_\_\_\_\_\_\_\_\_\_ and Ruffin White \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Class: ECE425 – Mobile Robotics

Date: 12/2/2012

Title: Lab01 – Introduction to Mobile Robotics

**PURPOSE**

The purpose of this lab was to introduce us to some basic concepts of mobile robotics and to the CEENBot robotic platform. In this lab we performed some basic short distance locomotive actions with the robot to observe the odometry errors present due to systematic or non-systematic errors.

Systematic errors refer to differences in wheel diameter, misalignment of the wheels, and discrepancies in finite encoder readings. Non-systematic errors refer to errors coming from the environment such as uneven floor elevations, unexpected objects encountered in the surface of the floor, and slippage in the wheels.

After performing the basic movements, we analyzed the odometry errors and utilized the University of Michigan Benchmark (UMBark) method to quantitatively compensate for the robot’s errors. We managed to acquire correction factors for the wheels based on the measurable features and offsets of the robot (e.g. wheelbase, wheel diameter, displacement offset from start, etc.).

The purpose of this lab was to also generate an intuitive and modular code architecture that will be able to perform all necessary locomotive tasks based on user inputted actions (i.e. pushbuttons for our case).

**PROCEDURES AND STRATEGY**

In this lab, we programmed the CEENBoT robot, named Arkin after the robotic pioneer responsible for behavioral control, to move in a square, circle, and figure eight. The CEENBoT is equipped with stepper motors necessary for locomotion at low speeds. The stepper motor rotates in discrete steps or angular movements that are divided equally in one revolution of the wheel. For our particular robot, the stepper consists of 200 steps per revolution and each step consists of 0.134 cm.

Square Locomotion (Part 2):

Here we were asked to program the robot to move in a square pattern with each side consisting of 2 to 3 feet. For this part we used the “step and wait” command and computed the number of steps necessary to perform a 90o zero-point turn.

Once the code was done, we had to perform 5 clockwise and counterclockwise runs in order to measure the odometry error and compensate for the systematic and non-systematic errors observed. *Refer to Error Analysis section for data regarding the 10 square traversals and scatter plot data.*

Circle and Figure ‘8’ Locomotion (Part 4)

Here we were asked to program the robot to move in a circle pattern with a diameter of 2 to 3 feet in length. For this part we had to realize that each wheel traversed different distances at different velocities but finished at the same time. Using basic geometry and physics concepts, we computed the length of travel for the inner wheel and outer wheel knowing the desired diameter (3 ft. for our case) and the measured wheelbase (21.3 cm for our case). We then chose that the time taken for one circle revolution should be 10 seconds. Thus, knowing the distance that each wheel needs to travel and the given time, we computed the velocities for each wheel.

To reverse the direction of the wheel (clockwise or counterclockwise) we simply swapped the calculated values for the right wheel and left wheel. For this part we observed almost no change in error even after tuning the correction factors.

To create the Figure ‘8’ motion, we programmed the robot to complete one clockwise circle and then complete one counterclockwise circle from its starting point. Barely any change was observed after tuning the robots correction factors.

**ERROR ANALYSIS**

Odometry Error Analysis

|  |
| --- |
|  |
| Figure 1: Odometry error offsets for our CEENBoT robot based on a CW and CCW square locomotive pattern. The offsets from the initial starting position (origin) are due to systematic and non-systematic errors. *The positive x-axis correlates to the front of the robot and the y-axis to the sides of the robot.* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | CW Square |  |  | | **Run** | **xOffset(cm)** | **yOffset(cm)** | | 1 | -14 | 11.5 | | 2 | -13.25 | 10.25 | | 3 | -8.75 | 13.5 | | 4 | -11.5 | 10.5 | | 5 | -14 | 11 | | Avg | -12.3 | 11.35 | | |  |  |  | | --- | --- | --- | | CCW Square |  |  | | **Run** | **xOffset(cm)** | **yOffset(cm)** | | 1 | -5.25 | 11.75 | | 2 | -3 | 11.75 | | 3 | -5.5 | 11.25 | | 4 | -4 | 11 | | 5 | -5.5 | 10.75 | | Avg | -4.65 | 11.3 | |
| Figure 2: Odometry error offset values measured. | |

*Refer to the MAPLE document (attached) for the calculation of the correction factors.*

Based on the errors computed above, we were able to come up with the correction factors. The correction factors for both, right, and left are fairly close to 1. Thus changes in the behavior of the robot were barely noticeable.

Additional Questions

1. What are the sources of the odometry error?

These errors are most likely a combination of systematic errors (encoder discrepancies, differences in wheel diameter, and differences in coefficients of friction in the stepper motors) and non-systematic errors (wheel slippage and uneven floor terrain).

1. How can you correct for this error?

Instead of manually correcting for this error, we can use the UMBark method to account for the errors in odometry that we observed. This method will yield correction factors for our wheels that we can use to tune the robot.

1. How can you improve the three motions (square, circle, figure ‘8’)?

We can come up with closer mathematical approximations that closely resemble the motion we are trying to generate. We would also need to perform several runs in order to account for any errors.

**CONCLUSION**

We were able to observe various types of odometry error involved in the locomotion of mobile robotics. We learn that the offsets can be due to systematic and non-systematic errors. To account for the errors we observed, we use the UMBark method to compute the offsets from the initial starting point and generate a set of tuning parameters for our wheels. The tuning parameters barely affect the locomotion of our robot since they are close to 1 and because the robot is actually traveling a small distance.

In the future, we want to add subroutines that will calculate the number of steps to perform any single point turns or arc lengths to perform any geometric pattern.