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

Memo

To: Dr. Carlotta Berry

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

Class: ECE425 – Mobile Robotics

Date:

Title: Lab02 – Random Wander, Obstacle Avoidance

**PURPOSE**

The purpose is to develop random wander and obstacle avoidance behaviors for the CEENBoT. We used *submission architecture* for the design of our program to incorporate several basic levels or *layers* of obstacle avoidance. Layer 0 of our control architecture included *collide* and *run away* behaviors to keep the robot from hitting obstacles. Layer 1 of our control architecture incorporates the *random wander* behavior which moves the robot a random distance and/or heading every *n* seconds.

**PROCEDURES AND STRATEGY**

Before we actually started programming the robot, we started testing the range sensors of the robot. Our CEENBoT is equipped with 4 short distance IR sensors that the robot can use to perceive information, in this case distance, from the environment.

We verified that the range of our sensors is valid from at least 5 cm and up to 45 cm away from each of the sensors. Although the robot can sense objects farther than 45 cm away, the feedback signal is too noise to make sound computations. *Refer to Figure 1 for a table displaying the IR analog voltage values for each of the IR sensors.*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Distance (cm)** | **Distance (")** | **IR Front** | **IR Left** | **IR Right** | **IR Back** | | 5 | 1.97 | 5.50 | 5.35 | 5.15 | 5.27 | | 10 | 3.94 | 11.09 | 11.19 | 10.04 | 10.29 | | 15 | 5.91 | 15.50 | 15.75 | 14.92 | 14.92 | | 20 | 7.87 | 20.90 | 21.42 | 19.96 | 19.67 | | 25 | 9.84 | 25.50 | 25.98 | 25.25 | 24.79 | | 30 | 11.81 | 30.00 | 30.00 | 29.35 | 28.14 | | 35 | 13.78 | 37.00 | 37.00 | 34.98 | 34.98 | | 40 | 15.75 | 41.00 | 40.63 | 41.85 | 41.23 | | 45 | 17.72 | 53.98 | 50.12 | 47.57 | 44.53 | |
| Figure 1: Range Sensor calibration data for the IR sensors. The minimum range observed was 5 cm away from the sensors while the maximum consistent range was 45 cm away. |

From the data measured above, we were also able to compute some basic error analysis for our sensors and verified that sensors, including sensors with same make and model, are not 100% identical. Thus these minute differences can yield to systematic errors that can yield great errors for robots that operate at large scales (i.e. not dead reckoning).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Distance (cm)** | **IR Front** | **IR Left** | **IR Right** | **IR Back** | | 5 | 0.50 | 0.35 | 0.15 | 0.27 | | 10 | 1.09 | 1.19 | 0.04 | 0.29 | | 15 | 0.50 | 0.75 | -0.08 | -0.08 | | 20 | 0.90 | 1.42 | -0.04 | -0.33 | | 25 | 0.50 | 0.98 | 0.25 | -0.21 | | 30 | 0.00 | 0.00 | -0.65 | -1.86 | | 35 | 2.00 | 2.00 | -0.02 | -0.02 | | 40 | 1.00 | 0.63 | 1.85 | 1.23 | | 45 | 8.98 | 5.12 | 2.57 | -0.47 | | **Stand. Deviations** | 2.7793275 | 1.5203738 | 1.043093689 | 0.818191 | |
| Figure 2: Error analysis and standard deviations computed for the IR sensors of our CEENBoT. From this data we can see that the IR sensor with the most error is the front sensor while the most reliable IR sensor is the back sensor. |

Layer 0: Collide and Run Away Behaviors

For this layer of architecture, we implemented the most basic obstacle behaviors for a robot. In the *collide* behavior, the robot ran forward and stopped 3” to 6” away from a front obstacle. The robot would then keep moving forward if the obstacle was removed. This is can be seen as the *aggressive kid* behavior.

For the *run-away* behavior, the robot would sit still in the middle of the room and only move away from obstacles that entered its IR threshold space. The robot must be able to detect obstacles from all sides at this point as well as corners. This type of behavior is known as the *shy kid* behavior.

The 2 types of behaviors that make up layer 0 of our submission architecture are the most primitive examples of obstacle avoidance.

Layer 1: Random Wander

For this layer of architecture, we programmed the robot to move in a random pattern when no obstacles are present. We used the rand() command to randomly move the robot by changing the wheel positions and velocities.

Once we got this part of the code working, created a smart wander behavior that would allow the robot to randomly wander the space until it detected an obstacle within its thresholds. The robot would then behave as an obstacle avoidance robot and then resume random wander behavior once its obstacle is no longer detected.

Additional Questions

1. What was the general plan you used to implement the random and obstacle avoidance behaviors?

We called the random function to randomize distances and velocities for each wheel. The random wander only ran when no obstacle was detected and then check the obstacle avoidance sub-routines once an obstacle was within the robot’s IR threshold.

1. How did you create a modular program and integrate the two layers into the overall program?

We broke each of the behaviors in as separate sub-routines and then had the higher level-subroutines initiate the other behaviors. Sort of like a bottom-up approach.

1. Did you use the servo turret to create a redundant sensing on the robot’s front half?

No. We did not use the servo turret due to the reliability of the front IR sensor.

1. How could you create a smart wander routine to entirely cover a room?

We used weighted random values that would potentially drive the robot forward. However, the robot has a 20% of driving backwards in case it gets stuck in corners. This maximizes our chances for exploring an entire room.

However, if you want to efficiently cover a room, we would then use D-star or A-star approaches to map out areas of the room that we have explored and areas that remain unexplored.

1. What kind of errors did you encounter with the obstacle avoidance?

We had to correct the avoiding/moving away behavior and the threshold sensitivities of the side IR sensors.

1. How could you improve the obstacle avoidance?

By tuning the threshold sensitivities of the side IR sensors, we managed to keep the robot from oscillating in corners. We made the robot fully turn in one direction and then move forward or backward to avoid the obstacle.

1. Were there any obstacles that the robot could not detect?

Any obstacle in the blind spots of the robot was hard for the robot to detect. We also had difficulties when detecting smooth or highly reflective surfaces like a mirror.

1. Were there any situations when the range sensors did not give you reliable data?

In bright or direct sunlight conditions, the sensors would read false data. Also the sensors were misleading when an object was too close or too far away. When it was too close the IR detector did not fully detect the emitted light while at long distances, the IR detector read a lot of noise.

1. How did you keep track of the robot’s states in the program?

We fixed this by using our subroutines to return values indicating our last behavior performed. By keeping track of the last behavior performed, the robot made the right decisions for the following behavior based on obstacle proximity.

1. Did you encounter any “stuck” situations? How did you account for those?

Being stuck inside a chair was truly problematic for the robot. The robot would sometimes not see the legs due to its blind spots or oscillate too much when trying to get out of this jail.

**CONCLUSION**

In this lab, we implemented a basic obstacle avoidance behavior to our CEENBoT using *submission* *architecture* for our design. We created some basic layers of obstacle avoidance followed by a random wander behavior. We integrated both layers in a high-level subroutine that would run the random wander while checking for the other subroutines. Prior to this level of architecture that we used, we used complicated finite state machines that did not prove to be modular or broken down into several layers of architecture. The type of architecture learned in this lab proved to be far more efficient that the previous approach.

*LOOK AT THE NEXT PAGE FOR SCANNED COPY OF SIGNATURES.*