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

Memo

To: Dr. Carlotta Berry

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

Class: ECE425 – Mobile Robotics

Date:

Title: Lab03 – Wall Follower, PD Control

**PURPOSE**

The purpose is to develop a wall following behavior on the CEENBoT platform using feedback control architecture. The CEENBoT range sensors (IR sensors and contact sensors) were used to sense the world around the robot.

The robot should keep away from walls at a distance of 4” to 6” inches and track the walls consistently. The robot should also be able to avoid obstacles, turn around corners, and adjust its distance to the wall if it gets too close or too far. If the robot detects no walls around it, it should operate in a random behavior until it finds a wall and starts tracking it.

**PROCEDURES AND STRATEGY**

For our overall procedure and strategy, we decided to create some basic reactive behaviors for the robot based on detecting obstacles and random wandering. These were the most primitive behaviors of the robot that were called by the higher level functions in the subsumption architecture design.

Specifically, our program ran the *“moveAway”* behavior which kept the robot from hitting obstacles. Then the *“moveWander”* function of the robot would call the *moveAway* function and if no obstacles were detected in the robot’s proximity range (i.e. *moveAway* returns a FALSE Boolean), the robot would then proceed to randomly move in its environment. If the robot detected an obstacle, the code would then suppress the random following behavior and allow the robot to inhibit a wall following and tracking wall behavior (i.e. *moveWall*). By inhibiting some high-level functions and suppressing the primitive behaviors, the robot is able perform the tasks of wall following behavior using the subsumption architecture theory.

Once we had implemented the correct behavior for our robot, we decided to start tuning the controllers of the robot so that the robot would gracefully follow the walls, turn around corners, center itself when it sees walls on both sides, and avoid obstacles as it is tracking the walls. The main aspect we decided to incorporate to our robot was the use of a full PID controller instead of just a PD controller.

The PID controller comprises of proportional (P), integral (I), and derivative (D) controller. The proportional controller takes the instantaneous error, or offset from the wall, and multiplies it by a scalar. This will cause our robot to oscillate back and forth as it follows the wall. The derivate controller will take the difference of the errors and multiply it by a scalar. In short, if the error difference is really high due to the robot being really close or really far away from the wall, then the robot will sharply turn towards the wall or move away from the wall. Finally, the integral controller takes the summation of errors over time and multiplies it by a scalar. Together with the P and D controller, the I controller will further help reduce oscillations over time due to the gradual changes in error.

Further performance tests and tuning revealed that the performance of the robot was greatly increased if we reversed the order of the motion. In other words, the actual front of the robot is actually the back. With this orientation, the differential drive wheel is now located in the back axis which makes it slightly more difficult for turning around corners. It also moved the contact sensors towards the back which in our case made these sensors useless. The only geat advantage that we observed with this change was the changes in IR sensor reading. With this orientation, the IR sensors will properly read if the robot is getting farther away or closer to the wall.

**QUESTIONS**

1. What does the diagram for the third layer of the subsumption architecture look like?

*Refer to question 11 for a diagram showing all layers including layer 3.*

1. What did the robot do when it encountered a corner while wall following?

When the robot encounterd an inside corner, the back sensor (which is now the front sensor) will read the wall in front and interpret the repulsive fields from the wall that is tracking and the wall in front of it to make the right turn. For outside walls, the robot would attempt to turn towards the discontinuity due to the summation of errors but due to the new orientation of the robot, the back differential wheels would sometimes catch on the walls. The main aspect about our robot while it turned corners is the fact that it never had to stop. The use of potential field theory along with arching around corners was sufficient.

1. What did the robot do when it encountered doorways while wall following?

When the robot encountered a doorway, it would act as encountering an outside corner. The robot would turn to the discontinuity and center itself when it passed through the door if both sides of the door are within 4” to 6” away from the robot.

1. When tuning the P and/or D controller, did the robot exhibit any oscillating, damping, overshoot or offset error? If so, by how much?

The robot did change drastically when tuning the P,I, and D controllers. Oscillations were greatly affected by changes in the P controller. Increasing the P controller would cause the robot to instantenously turn more which eventually made it lose the wall or crash towards the wall.Increasing the D controller prevented the robot from crashing the wall or running to far away from the robot. Finally the I controller had only visible effects over time by dampening oscillations overtime and making the robot follow a more “straight” path.

1. What were the results of the different P , D, and I controller gains? How did you decide which one to use?

The following P, D, and I controllers were for our own robot and may significantly vary between robots due to processing speed and sampling rate of the program.

|  |  |
| --- | --- |
| **Proportional Controller Gain** | 1 |
| **Derivative Controller Gain** | 5 |
| **Integral Controller Gain** | 0.001 |

1. How accurate was the robot at maintaining a distance between 4” to 6”?

Our robot was fairly consistent when maintaining its distance 4” to 6” away from a wall. When turning inside corners, the robot did look like it was a bit outside the range but then it managed to correct itself again.

1. Did the robot ever lose walls?

The robot tracked all walls just fine. The only problem with losing walls was when the robot made an outside turn. The sensor location along with the motion of the turn sometimes made the robot lose the walls.

1. Compare and contrast the performance of the *Wander* and *Avoid* behaviors from last week.

|  |  |  |
| --- | --- | --- |
|  | **SIMILARITIES** | **DIFFERENCES** |
| ***Wander* Behavior** | Randomly wander when no obstacles are detected. Random behavior affects speed and turning of the wheels. | Now outputs a boolean that indicates whether it is in a wander state or not. |
| ***Avoid* Behavior** | Works like the shy kid behavior. | Instead of running away from objects it now keeps a fixed distance to the objects and tracks the objects if they attempt to get away. |

1. What was the general plan to implement the feedback control and subsumption architecture on the robot?

*Refer to the Procedures and Strategy* *section for a discussion of the overall plan.*

1. How could you improve the control architecture and/or wall following/follow center behaviors?

We can fine tune the prefilter and remove the print to LCD statements to make the robot more reactive. The only drawback from this improvement is that the controllers would have to be re-tuned due to a sampling time change.

1. What does the overall subsumption architecture diagram with all layers look like?

[PASTE DIAGRAM HERE]

1. What was the pseudocode and flow chart for the program design?
2. Did you use any suppresion and inhibition with the higher levels of the subsumption architecture?

We did use suppresion and inhibition behaviors in the higher level functions by using low-level functions that output a boolean type. The TRUE or FALSE output would then trigger other behaviors and suppress others.

**CONCLUSION**

In this lab we created a wall following behavior for our robot. We used the subsumption architecture style to inhibit and suppress low-level behaviors that were created in the last lab. Finally, we also incorporated a PID control scheme to our robot to reduce overshoot, dampen oscillations, and correct errors when tracking walls. In the end, our robot was able to track walls, go around inside and outside corners, center itself when between two walls, and randomly wander until it encounters a wall.