Memo-ECE425-LAB2

Date: 1/16/2022

To: Professor Berry

Team: Leonardo - Nathan Lee, Shantao Cao

The purpose of this Lab was to build on top of obstacle avoidance and have the robot respond to the world built around it, shifting from a hybrid controller to a behavior-based controller. For this Lab, we used LEONARDO’s four IR sensors and two sonar sensors to read its surrounding walls. On top of the obstacles avoidance behavior, the robot was added the wall following behavior using a proportional-derivative (PD) controller. The behavior controller uses behavior fusion, behavior attribution and emergent behaviors advancing LEO to another level of robotics.

**Prelab: State Diagram and Simulation**

To begin the lab, we planned LEO’s behavior attribution using a state diagram (Appendix A) to determine which behavior method to use. The state diagram is a high level symbolic diagram for adding the obstacle avoidance behavior from last lab with the wall following behavior in this lab. The robot simulation tests the implementation for a wall following behavior (Appendix B). Wall following is a new primitive behavior added state in this lab. All other states, outside/inside corner, avoid obstacle, and random wander, recycles previous lab behaviors and code. The PD controller was planned to follow the wall proportional to the distance from the wall being followed and designed with dead zone to reach a steady state following the wall. This is shown in the simulation by collection the rows of coins along the wall.

**Wall Following**

Wall following proved to be a more advance behavior than previous behaviors because of the addition of a PD controller. After simulating a wall following behavior, we implemented a basic wall following behavior using a bang-bang controller. This initial controller takes sensor data from the walls/obstacles around it and turns a set angle away or towards the wall accordingly. Like the simulation a dead zone was implemented to have LEO reach a steady state distance from the wall. LEO’s implementation was successful and would follow the wall with oscillation.

After a simple bang-bang controller, a PD controller was implemented. The PD controller is proportional to the wall distance and the speed at which LEO approaches or veers from the wall. We wanted a quick response time, so we used a proportional gain of 10. This results in a large overshoot, this could also be the cause of the sensor not being parallel to the wall and giving us a wrong reading. The large over shoot also acts like an avoid obstacle behavior and guides LEO away from the wall. As the robot turn, the sensor turns with the robot and the next reading was a larger or smaller distance than expected. Adding a derivative controller helps reduce the overshoot and helps LEO reach steady state faster with less oscillations. We used the same derivative constant of 10 as well. The derivative controller runs into the same issue as the proportional controller but acts opposite to the proportional gain. The resulting controller lets LEO follow a wall inconsistently. There ends up being some instances where LEO overshoots away from the wall and compensates for it by turning into the wall. The side sensor then becomes parallel to the wall and cannot recover from wall following. We solved this issue by adding an artificial damping to the system by restricting how much LEO can turn for a set amount of wheel rotations. This proves to be much more reliable than the implemented derivative controller because the resulting PD controller is more consistent. The PD controller using purely the IR sensor will always have the same issue of giving a false reading due to it not being perpendicular to the wall.

**Avoid Obstacle**

The avoid obstacle behavior uses similar code to previous labs. Using a large proportional gain, we did not need to implement avoid obstacle for wall following. This means in left and right sides of the robot’s obstacle avoidance behavior because it will be accounted in the wall following behavior. The rear sensor does not need an avoid obstacle behavior because we have LEO only moving forward in all behaviors. This results in front sensor is the only sensor needed for the avoid obstacle behavior.

The avoid obstacle behavior written has LEO detect a wall in front of it and turn accordingly. LEO would turn 90 degrees left if he was right wall following state and right if he was in the left wall following state (inner corner). The inner corner behavior was different from last lab because we needed to account for how far LEO was from the followed wall. If LEO was too close to the followed wall, his body may hit the wall and result in major odometry error. The new resulting inner corner behavior is to avoid the body hit scenario and turn towards the corner and back up to allow its entire body to swing. Then turn accordingly with some compensation from the initial turn. Another front sensor obstacle avoidance scenario is when only the front sensor is triggered. LEO defaults to turning 90 degrees left. Our final planned scenario is when LEO when encounters an end of hallway. It would turn 180 degrees and leave the hall; this scenario usually follows a center following state.

**Random Wander**

The random wander behavior is taken from the prior lab and modified. Since we did not have a rear sensor avoid obstacle behavior, we changed the random wander behavior to only move forward. Following the state diagram, LEO would then follow a wall or avoid obstacle from the random wander behavior. The random wander behavior is achieved after losing a wall and testing that the wall was an outside corner. By default, LEO would assume that there is an outside corner to the end of every wall and try to maneuver around the wall accordingly. If LEO confirms that the wall is not an outside corner, it will resume the random wander state.

This set of behaviors helps LEO when wall following. With the large overshoot in the PD controller for wall following, LEO sometimes loses the wall. This then has LEO turn towards the wall if it is an outside corner and runs into the wall. The avoid obstacle behavior takes over before hitting the wall and results in a left turn into right wall following.

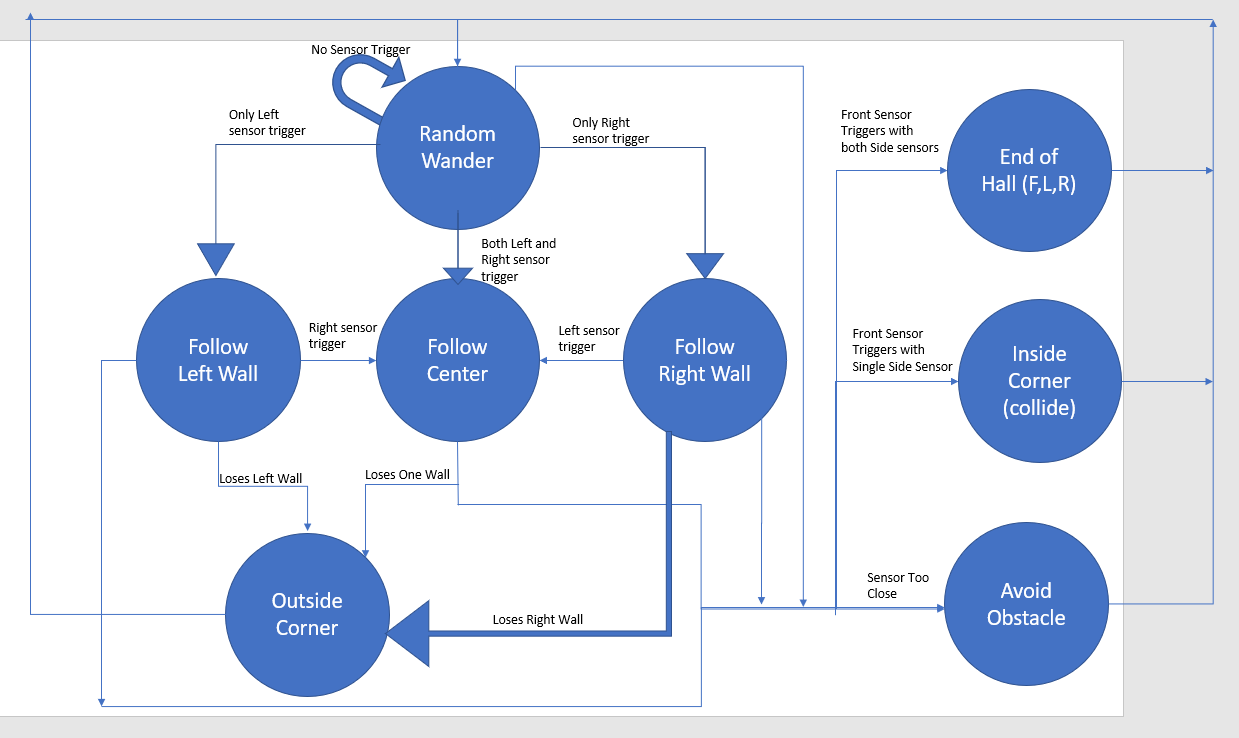
**Follow Center**

The last layer to wall following is following the center of a hallway. The follow center uses the same method as the wall following, but the error used for the proportional and derivative is set to be the difference between the left and right sides. Follow center also does not need artificial damping because LEO is surrounded by walls and cannot lose it. The proportional gain also has been reduced to 4 to avoid overshooting and driving into a wall. Follow center needed the addition of two sonar sensors to verify LEO’s left and right distances. The sonar sensors are also at a slight angle to help compensate for turns and having the IR sensors not being parallel to the wall. LEO proved to be successful at follow center and end of hallway behaviors.

**Behavior Attribution**

From a subsumption architecture stand point (Figure 1), layer 0 is the avoid obstacle behavior. Avoid obstacle is default because no other behavior would be suppressed or inhibited by this behavior. Layer 1 is wall following and suppresses avoid obstacle. Wall following is the next layer because it is used as the left and right avoid obstacle behavior. Layer 2 is follow center because it will suppress wall following. The final layer is random wandering because all behaviors have bigger priorities than random wandering. This subsumption architecture layer is like last lab and adds more layers on top of random wandering and avoid obstacle. We preferred to use a state machine due to our previous experience with machine states and their implementation. The state diagram is shown in Appendix A and shows the flow of the robot behavior. LEO’s states changes based off readings from the sensors. The following states are determined by the previous and current state.

**Appendix A: PreLab State Diagrams**



**Appendix B: Simulation Code**

#include <Servo.h>

Servo leftservo;

Servo rightservo;

const int pingPin = 5; // Trigger Pin of Ultrasonic Sensor

const int echoPin = 6; // Echo Pin of Ultrasonic Sensor

int t;

int nowall;

void setup() {

nowall = 0;

leftservo.attach(9);

rightservo.attach(10);

//set up the Serial

Serial.begin(9600);

//setupt the pin modes

pinMode(pingPin, OUTPUT);

pinMode(echoPin, INPUT);

leftservo.write(90);

rightservo.write(90);

}

void loop() {

long duration;

//clear the ping pin

digitalWrite(pingPin, LOW);

delayMicroseconds(2);

//send the 10 microsecond trigger

digitalWrite(pingPin, HIGH);

delayMicroseconds(10);

digitalWrite(pingPin, LOW);

//get the pulse duration in microseconds

duration = pulseIn(echoPin, HIGH);

Serial.println(duration);

if (duration<=6300 && duration >=5700){

nowall = 0;

forward();

}else if(duration>6300 && duration<=10000){

nowall = 0;

t = (duration-5800)/1.5;

turnleft();

delay(t);

forward();

delay(500);

turnright();

delay(t/1.2);

stop();

}else if(duration<5700 && duration >=0){

nowall = 0;

t = (6100-duration)/1.5;

turnright();

delay(t);

forward();

delay(500);

turnleft();

delay(t/1.2);

stop();

}else if(duration>10000){

if (nowall == 0){

turnright();

delay(2500);

forward();

delay(1000);

stop();

nowall = 1;

}else if(nowall >= 1){

turnleft();

delay(2500);

forward();

delay(500);

turnright();

delay(2500);

//forward();

//delay(500);

//turnright();

//delay(3000);

stop();

}

}

/\*

TASK: The coins are around 110 cm away from the top wall.

Use the ultrasonic sensor data to navigate the robot in order

to collect the coins.

\*/

delay(50);

stop();

}

void forward(){

leftservo.write(170);

rightservo.write(-180);

}

void turnleft(){

leftservo.write(-180);

rightservo.write(-180);

//delay(t);

}

void turnright(){

leftservo.write(180);

rightservo.write(180);

}

void stop(){

leftservo.write(90);

rightservo.write(90);

}