CS556 Final Project Report

Sebastián Ibarra-Pérez and Gabriela Calvo

1 Project Proposal

1.1 Problem Statement

Our robot performs an automatic traversal of an enclosed area, avoiding obstacles, identifying and collecting randomly placed trash items, and returning to its charging station.

1.2 Sensors

Our Pololu robot makes use of two types of sensors.

Sonar Sonar is an ultrasound sensor that is attached to the servo motor on top of the robot body. For our purposes, the servo takes readings to record how close the walls of the environment are. These readings are taken on the sides or the front of the robot depending on which part of the task we are in.

Infrared There are four IR sensors attached to the bottom of the robot body that are used to recognize different colors on the floor of the environment. The sensors record a value between 0 and 1000 and throughout the implementation we use the average of the four sensor readings to represent the color that is under the center-front of the robot.

1.3 Behaviors, Automation, and Control Architecture

STATES

State 1: Setup and Calibration This state includes the necessary calibration and positioning for the robot to complete its tasks. It begins with the rotating in place to calibrate the line sensors in order to detect trash and home squares later in the implementation. After this, the robot moves one square forwards, off of the home square, and transitions to the next state. Aside from the color values on the floor that are read during the calibration, this state is entirely hard-coded.

State 2: Right Side Wall Following This state is an updated version of our wall following code from a previous lab made to look to the right side of the robot rather than the left. This state has dynamic goals, as the layout of the environment is not known to the robot as it is making its way through. During this state, the robot uses a PID controller to maintain a distance of 10 centimeters from the wall on its right side. During this state, values from the odometry_update method for consecutive loops are used in the distance formula to estimate the robot's movement. Every 20 centimeters traveled, the sonar faces forward to take a reading. If there is anything closer than 10 centimeters away, the state transitions to State 4: Obstacle Avoidance.

State 3: Left Side Wall Following This state is the same implementation as State 2 but adjusted to look to the left so that the robot maintains a distance of 10 centimeters from the wall on its left side. As with State 2, this state has dynamic goals and moves the sonar forwards for a front reading every 20 centimeters traveled to check for obstacles.

State 4: Obstacle Avoidance This state is mainly meant to handle corners and areas where the robot might be backed in by walls. During this state, the sonar remains facing forwards and the robot continues rotating until there is no longer an obstacle detected within 10 centimeters. This state includes a parameter representing the previous state because it acts differently depending if we are coming from State 2 or State 3. If the robot was previously in State 2, the robot turns left as long as the front-facing sonar reading remains less than 10 centimeters. If the robot was previously in State 3, the robot rotates right as long as the front facing sonar reading is less than 10 centimeters. Once the front facing sonar has a reading greater than 10 centimeters, the robot returns to its previous state which will either be State 2 or State 3.

State 5: Trash Collection This state is also dynamic and triggered when the IR sensors detect the color value representing trash. In this state, the robot turns in place to represent collecting the trash and displays the current number of trash items collected on its OLED display. Similar to State 4, this state includes a parameter for the previous state so that once the trash is collected, the robot can return to whichever version of wall following it was performing.

State 6: Return Home This state is triggered when the robot has collected exactly 3 pieces of trash meaning it is the result of a hard-coded goal. In our implementation this state is only reachable from a Trash collection state that was entered from State 3: Left Side Wall Following. In order to return home, the robot performs a 180° turn and uses right side wall following to essentially retrace its steps. We chose to implement it in this way because it would involve less backtracking than allowing the robot to complete its entire left side wall following loop through the map. This state continues until the home square is

detected. Although the home position will always be in the same spot, our robot is not using a map and will not recognize the home position until it is on top of it, making this a dynamic goal.

State 7: At Home Upon entering this state, the robot will check whether all of the trash has been collected. If there are remaining trash items, the robot will enter State 3: Left Side Wall Following in order to reach the squares that were not traversed during State 2: Right Side Wall Following. If all three trash items have been collected then the robot has completed its task and does not need to traverse the rest of the environment. At this point the robot plays a success tone to indicate that it has completed its trash collecting task. Since we are expecting a defined amount of trash, this is a hard-coded goal.

TRANSITIONS Our implementation contains thirteen possible transitions.

State 1 → State 2 IF calibrations and set are complete

State 2 → State 4 IF Front facing sonar reading is less than 10 cm

State 4 → State 2 IF Front facing sonar reading is greater than 10 cm AND previous state is State 2

State 2 → State 5 IF trash detected

State 5 → State 2 IF trash collection display complete AND previous state is State 2

State 2 → State 3 IF home square detected

State 3 → State 4 IF Front facing sonar reading is less than 10cm

State 4 → State 3 IF Front facing sonar reader is greater than 10cm AND previous state is State 3

State 3 → State 5 IF trash detected

State 5 → State 3 IF trash collection display complete AND previous state is State 3

State 5 → State 6 IF three trash items have been collected AND previous state is State 3

State 7 → State 3 IF trash count is less than 3

State 8 → State 9 IF home square detected

1.4 1.4 Finite State Automata (FSA) Diagram

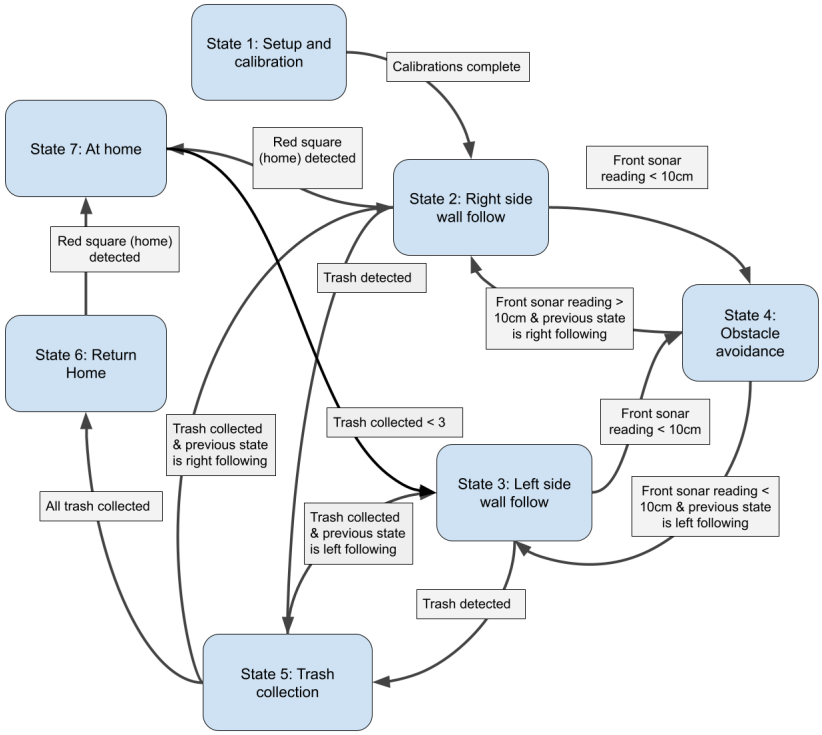

Fig. 1. Relationship of all transitions and states in our implementation.

1.5 1.5 Control Architecture

We have one PID controller that is used to facilitate the wall following states. This is a reactive control approach because the robot reacts in response to the sonar reading which is the actual distance away from the wall compared to the goal state of 10 centimeters. We chose to use PID rather than P or PD because it was the smoothest movement from our in lab on wall following.

2 Integration of Semester Modules

Lab 1 We make use of the Lab 1 navigation primitives throughout our implementation for predefined movements. In State 1, we use the motors.setSpeeds() and delay() methods to turn the robot in place during sensor calibration and to move the robot off of the home square to start its traversal. The motors.setSpeeds()

method is used throughout the rest of the states but its use aligns more with later labs because we are not using delay() statements to control the distance that the robot travels.

Lab 4 The concepts and update_odometry() method from Lab 4 are extremely important for this implementation. We use odometry during States 2, 3, and 6, to have a constant estimation of how far our robot have traveled. Since the units of the map are 20 x 20 cm squares, we have programmed our robot to move the sonar forward and take a front-facing reading to check for obstacles after every 20 centimeters moved. The distance moved is calculated using the distance formula on the x and y values returned from the update_odometry() method. Once the distance between the current and previously stored x and y values reaches 20 centimeters, the sonar takes a front-facing reading and the previous x and y variables are set to the current x and y values.

Lab 5 We reused most of our code from Lab 5 because wall following is the core movement for our implementation. We modified the PD controller to instead be a PID controller so that we could fine tune our robots movement to be smoother, especially around corners.

Lab 6 We incorporated our Go-To-Angle PID controller code during the trash collection state to make sure that our robot turns exactly 3 times. At first we had this portion of the implementation hard coded using the motors.setSpeeds() and a delay() statement. This updated solution reusing the Lab 5 code is an improvement from our original because ensures that the robot turns exactly 3 times regardless of the base speed. This lab was also our first use of a PID controller which is the type of controller that we used throughout our solution.

Lab 9 The line-following with obstacle avoidance lab was very helpful in this final project because it was our first use of States to control the robots decisions. We reused our switch-case approach to program flow from this lab to facilitate our 7 States and all of their transitions. We also reused the calibration code for our robot to be able to recognize the trash and home squares.

3 Challenges, Error Handling, and Reliability

3.1 Challenges

Distance formula issues: One challenge was the distance formula for checking for obstacles. We wanted a way for detecting walls in front of the robot while doing right wall following, so we came up with doing a check every 20 cm for a wall in front with the idea that those checks would always be in the center of each square. But because of corners, the robot might not reach the center of each cell and the checks wouldn't always land on the center of the cells, which

might cause the sonar checks to be miss aligned, potentially causing the robot to miss obstacles and bump into the wall.

Recognizing Trash vs regular floor vs Home square: We needed a way to classify each one of the squares as well as the regular floor. Errors here could cause missed trash collection, prematurely end the run by detecting the home square on accident, or trigger state changes when they aren't necessary.

3.2 Error Handling

Distance formula issues: To try to prevent any missed obstacles, we used distance thresholds for detecting distances. If the forward reading was less than 15 cm then the robot would keep turning until it was more than 15 cm, then it would continue wall following. This way even if it were to bump into a wall, the front sonar would read a distance of almost 0, which would still allow the robot to detect the wall and avoid it afterwards.

Recognizing Trash vs regular floor vs Home square: Since the sensor values fluctuate a lot, we tested our robot on the three surfaces, printing the IR readings from each of the 5 sensors. We then used a threshold based check to detect the home and black squares. Our robot performs readings every 5 cm traveled, and only triggers when at least 3 out of the 5 sensors read above the threshold limits we set. For the regular floor the readings were all below 100, for the home square the readings were all above 100, but below 200, and for trash the readings were always above 900. For preventing repeated triggers, we added flags that prevent the detection of trash or home squares until the robot is out of the square it just detected.

3.3 Reliability

Since our solution is not implemented using the map, the robot is less susceptible to unexpected changes to the environment or small differences to its movement between runs. The robot's performance is entirely in reaction to sensory data and adjusts well if it gets slightly off course.

4 Testing and Validation

4.1 Testing Strategy and Metrics

Since the main control of our robot is a switch-case block with different cases representing each of our states, a majority of our testing was making sure that the state transitions were occurring properly. This was done by printing the state to the Serial Monitor while the robot was running and isolating the points of error while we were doing test runs through the course. Additional testing included taking readings of the floor colors to know the ranges corresponding

to each possible floor color. Since we had previously implemented wall following in the labs, the majority of our testing was done on the corners of the maze. We did a lot of trial and error with different k_p, k_i, and k_d values to make the robot's handling of difficult areas as smooth as possible.

4.2 Edge Case Testing and Modifications

One drawback of not using a map is that we did not have a set location for the home position so we needed to use the IR sensors to recognize the blue color of the home square. One edge case we encountered was when the robot was approaching a black square it would sometimes return a reading that corresponded to a home square reading. We suspected that this was because we were constantly taking in IR readings of the ground and when the robot was partially on the black square it would give an in-between reading which would trigger the At Home state. To address this, we modified our code to only take in IR readings of the floor at intervals and we also studied the IR values more closely to make the home square IR range as small as possible.

5 Innovation and Creativity

As seen in our FSA, the obstacle avoidance state is shared by both wall following states and we needed a way to distinguish which side wall following the robot should return to once the obstacle is cleared. To address this, we included a variable for the previous wall following state.

We initially faced difficulty with the robot bumping into walls when backed into a corner. For regular corners it was fine to use our initial approach of turning the robot 90 degrees and then continuing with the fall following but this didn't work in situations where the robot is surrounded on three sides. Our solution to this problem was to change the way our robot avoids obstacles. We now have the robot constantly taking front-facing sonar readings while turning with 100 millisecond stops added after every reading to prevent the robot from over-correcting. The robot does not return to regular wall-following traversal until the path in front of it is cleared.

Another challenge we faced was the trash and home squares being detected multiple times. In order to address this we added boolean variables trash_detection_enabled and home_detection_enabled which must be true for the robot to register a reading. When the robot detects the IR value corresponding to one of these special squares its corresponding boolean variable is set to false until the robot detects an IR reading that does not fall in the range of that special square. This ensures that the robot will not register a detection of trash until it is fully off of the trash square and likewise for the home square. The home and trash squares correspond to different IR thresholds so they do not conflict.

6 Communication and Team Collaboration

We were able to reuse a lot of our lab code which helped keep our in-class time for testing and running the robot within the maze. Our initial approach to the project was to have separate implementations and compare them when we met in person but as the time went on we stuck with one version and would review any changes that were made away from class time at the start of each session.

We made a GitHub repository of our project to keep track of the current version as well as a shared google doc to keep note of our changes in approach as we built out more of the project. Talking out our individual changes was extremely important for us successfully completing the project because we both fully understood the implementation and how it correlated with our FSA and state organization. This led to our in-class time being used well because we could equally contribute to solving any errors we encountered. Throughout the semester we organized all of the labs in a shared Google Drive which made it very easy to reference any old code that we wanted to reuse.

7 Use of Resources

We used all of the available resources in order to complete the project on time. This included office hours, class discussions, online materials, instructor feedback and previous labs. One of the most useful resources was a Stack Overflow page that suggested using a switch-case block to handle different states. This was the backbone of our implementation and extremely helpful with translating our FSA to actual code. Attending every office hours section was also extremely important because our time with the robot was limited and it was not possible to test our code outside of class. It was also very helpful to talk to our classmates about the errors they were encountering even if we had different overall approaches.