**ME 439 Robot Follow Wall Until Find Friend**

Trevor Wallis, Joshua Ho, Vijay Shah, Shardul Shrikhande, Hongyuan Qi, Matthew Dudzic

**Introduction:**

The goal for our project was to use two mobile robots to work together to solve a maze. We wanted to implement the two robots to each navigate half of a maze, meet in the middle of the maze, communicate their half to the other robot, and ultimately follow the other robot’s path to finish the maze the other way. This would require us to implement a wall-following program for the mobile robots that we used in class. It would also require recognition and communication between robots. The robots would follow the left or right wall based on ultrasonic sensors until they locate one another, using a camera and Aruco code and computer vision. Then they would relay the most optimized path to each other and find their route out. We thought this project was a great example on how a swarm type robot can be used to map out unknown environments in a group. This can be used in search and rescue of collapsed buildings or exploring shipwrecks and much more.

**Design:**

One of the new components of the project that we wanted to implement was wall following. In order to make wall-following easiest, we installed a total of three ultrasonic sensors to each of our robots. The sensors were aimed to the right, left, and front of the robot. When programming the robot, we decided to check the sensors in a certain order so that we would always follow a wall along the side of the robot. For the remaining explanation, we will assume a left wall follower. We would first check the left sensor and see if there was a wall in close proximity. If there was a wall detected to the left, we would check the front sensor. If there was a wall detected to the left and front, we would check the right sensor. If there was a wall detected to the left, front and right, the robot would turn 180 degrees around and drive away because it has reached a dead end. If one of the sensors that were checked in the order above did not register a wall, the robot would turn that direction and drive that way. For example, if there was no wall detected to the left, the robot would turn to the left 90 degrees and drive forward. If there was a wall to the left, but no wall to the front, the robot would drive straight forward. If there was a wall to the left and front, but no wall to the right, the robot would turn right 90 degrees and drive forward.

It took a lot of time to calibrate the ultrasonic sensors so that they would recognize walls that were within a limited distance and not read walls too far away or read signals from the other sensors. This is due to the fact that the MaxBotix sensors we used are designed to have a minimum reported distance of 6 inches which would have made our maze massive. At first, we tried converting the ultrasonic sensor analog data into a measured distance in meters. We found that this was less accurate than simply using the analog sensor data, because the distances in meters would be anywhere from 0.1-0.3 meters at the most while the analog sensor data was between 10-150. We determined that it is easier to use the analog sensor data and set a limit for wall distance limit because there is a larger range of values that the sensor can read.

Another new aspect of the project that we wanted to implement was recognition of the robots using computer vision. We wanted to use an Aruco code and web camera to have the robots identify each other. Going beyond what we did in class, we wanted to use the identification to trigger a different program that will enable communication between the robots. The Aruco codes will be pieces of paper mounted on the front of each robot. The web camera will always be on and looking for an Aruco code. Once the robot sees an Aruco code, it will use the distance and location information to move towards the robot that it sees. It will then stop when it gets close and the communication program will be triggered. This required us to process the message sent by the Aruco OpenCV node and extract the specific ID number and distance from the message. Using that information, we want to perform certain tasks such as move towards the other robot and begin the communication portion of the goal.

We also wanted to incorporate a system for the robots to optimize the path that they followed. What we decided to do was break down the path that each robot followed to the waypoints. Then, when the robots needed to communicate with each other, they would simply send the set of waypoints that they followed, and the other robot would follow the same waypoints in the reverse order. In order to select waypoints from the path that the robot traveled, we used waypoint seeking along the way to move the robot. Each time the robot took a sensor scan, it would determine which direction to travel, and it would set a waypoint for that direction at a specific distance away. It would also set smaller set points around the robot for optimizing the path later. For example, if the robot determined that it needed to travel to the right, it would set a waypoint for 0.2 meters to the right of its current location. It would traverse to that waypoint, and upon reaching it, the robot would take another sensor scan and determine the next waypoint that it needed to set. We realized that this may not result in the optimized path, if for instance, the robot ran into a dead end, but we figured that this would be the easiest way for the robots to find their way out of the other side of the maze.

**Results:**

We were able to accomplish some of our goals for this project, and there were some goals that we were not able to fully implement. We were able to get the robots to follow along walls on the left side and turn or continue straight depending on what the sensor readings were. This allowed the robots to explore the maze without crashing into walls. Along the way, the robots stored the waypoints that they navigated along with points that were open space determined from the ultrasonic sensors in order to be communicated to the other robot at a later time.

We were able to get the robots to recognize when they were in close proximity to each other using the web cameras and Aruco codes that we mounted on the robots. Once the robots recognized each other, we were able to get the robots to stop using waypoint seeking and sensor data to move. Essentially, we killed the robots when they got to the center of the maze.

We were not able to accomplish the goal of robot communication, but we think that is a viable next step. If robot communication was successfully implemented, we think that it would be easy for the robots to finish the maze, because they would simply need to follow the other robot’s waypoints in the reverse direction.

**Discussion:**

Throughout this project, we learned a lot about controlling ROS nodes, program timing, and using some new tools in ROS and Python. This project also introduced members of our group to using GitHub to allow for parallel collaborative code development. We got to practice creating our own ROS nodes using Python that were able to perform certain tasks, such as reading and processing sensor data and making decisions about where to move.We also learned how to turn on and off certain ROS nodes by using boolean variables in the Python code. We used a boolean variable to control when the robot would perform wall following versus when we would use the camera and Aruco code to find the other robot. Once the Aruco code was seen by the camera, that variable was switched to tell the robot to stop following walls and move closer to the other robot.

There are a few next steps that could be continued in the future that we did not have time to implement. We wished that we could have implemented the robots communicating the waypoints with each other and completing the maze. We did not have the time to get the robot communication to work due to issues we had getting the sensors to properly function, so this is a goal we did not accomplish. If robot communication had been implemented, the robots would simply have to follow the waypoints solved using Dijkstra’s algorithm and communicated from the other robot.

Beyond the goals for this project, it could be interesting to learn how to implement more than two robots working together to accomplish a task. Our goal for this project was to demonstrate the capabilities of two robots working together to accomplish a task, which in this case was to solve a maze. In the future, it could be possible to implement many robots in a foreign environment to explore and create a map of the environment more efficiently than just a single robot could accomplish. It would be especially interesting to figure out how to implement a swarm of an undefined size.

One large issue that we ran into was calibrating and using the ultrasonic sensors to measure accurate distances away from the walls. It could be interesting to implement a LiDAR sensor in the future that will provide more accurate measurements and detail than the ultrasonic sensors can provide. This would require a rework as to how valid positions are determined in our position graph and would allow the robots to traverse an environment that is more realistic than just a maze.

**Contributions:**

| Team Member | Trevor Wallis | Vijay Shah | Joshua Ho | Shardul Shrikhande | Hongyuan Qi | Matthew Dudzic |
| --- | --- | --- | --- | --- | --- | --- |
| Planning and organization | 100% |  |  |  |  |  |
| Maze design and construction |  |  |  |  | 100% |  |
| Robot modifications |  |  |  | 100% |  |  |
| Sensor calibration and data processing |  | 50% |  | 50% |  |  |
| Maze navigation |  |  | 50% |  |  | 50% |
| Launch file |  |  |  |  |  | 100% |
| Position graph and solution algorithm | 50% |  | 50% |  |  |  |
| Troubleshooting | 33% | 33% | 33% |  |  |  |
| Computer Vision |  | 100% |  |  |  |  |