EECS 464: Hands-on-Robotics Project 1 Report

Green Team
Luyang Hu, Josh Azrin, Yuesheng Ren, Yuanbin Zhu

November 2022

Contents

1 Background

1.1 Project Task Specification

As specified in the 2022F Project 1 Task Specification, we were asked to build a robot that could manually move to the first waypoint and then autonomously reach the remaining 3 waypoints in a specified order within the 2m $\times$ 2m arena shown in Figure 1. At the same time, the robot would need to continuously point a laser along the positive y-direction to a projection screen. In order to satisfy the structural limitation of the robot tag being less than 1m, we set the height of the robot to 20cm. The following three situations would count as a disqualification: 1) if the robot went out of the boundary of the arena, 2) if the robot failed to reach all the waypoints within fifteen minutes, or 3) if the laser illuminated in a direction with a negative Y component. To receive a passing score in the project, the robot needs to stay in the arena and reach the waypoints autonomously in sequence.

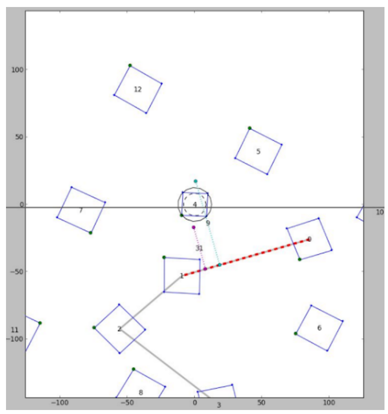

Figure 1: The Diagram of the Arena

1.2 Possible Designs

We considered three design concepts. The work of previous teams inspired some of our ideas as shown in our resource document. We will briefly go through each design idea in the following subsections.

1.2.1 Holonomic Robot With Rotating Tag

After reading through reports from previous teams, we found that motion tends to be implemented by using a ”zig-zag” approach in the x and y directions, whereby the robot eventually reaches its destination by continuously moving a small distance along the x or y direction. Inspired by this idea, we came up with this idea of designing a holonomic robot with rotating tag as shown in Figure 2. The tag would be fixed on the top of the robot and it would rotate at a certain angle after reaching each waypoint, so that the sensors on both sides of the robot tag would be distributed on both sides of the virtual line between the current waypoint and the next waypoint.

1.2.2 Two Wheeled Buggy With Rotating Laser Turret

The idea of our two wheeled buggy with a rotating laser turret was inspired by the Winter 2022 Blue team design as shown in Figure 3. The forward moving and steering of this robot design were easier to implement compared with the holonomic robot illustrated above. One motor was mounted in the middle layer of this robot for rotating the laser turret. The tag was fixed on the top of the robot. This robot adjusted the speed of the side motors and the angle of the laser turret by collecting and analyzing the sensor data and position data

Figure 2: The schematic diagram of the holonomic robot

when hitting the next waypoint. Some factors to take into consideration include the extra steps for elaborately calculating the angle adjustment of the laser turret and the noise when estimating the position information of the robot.

1.2.3 Wheels in the Air to Rotate

The inspiration of this design came from the Winter 2022 Purple Team as shown in Figure 4. We utilized two motors for moving forward and one motor for steering. The tag was fixed on the top of the robot. The robot would change the directions of movement by lifting the wheels and rotating the wheels. We also brainstormed to increase the friction at the bottom to ensure that the base will not slide while the robot was rotating.

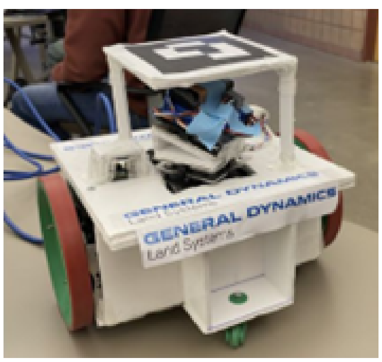

Figure 3: 2022W Blue Team's Robot

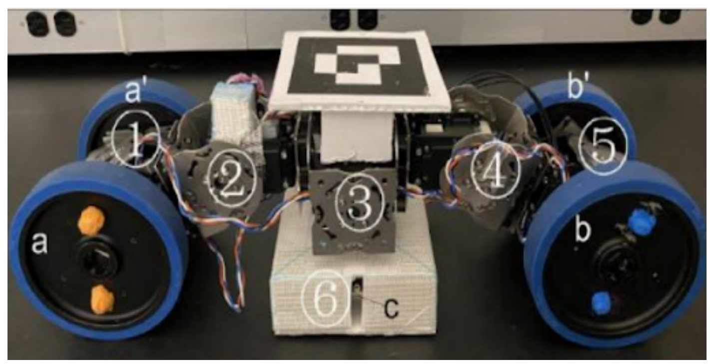

Figure 4: 2022W Purple Team's Robot

1.3 Final Design

We eventually settled on the design of a three-wheeled robot with a rotating tag. The following two subsections will describe the mechanical and software design of the robot, more details can be found in our how-to document.

1.3.1 Mechanical Design

As shown in Figure 5, our robot consisted of four main parts: a base fiberboard, a upper fiberboard, three Omni-directional wheels, and a rotating tag. The base board would support the bottom of the four motors, while the upper board would fix the top of the four motors, in order to increase the stability of the whole structure. To make the connection between the wheel and the motor stronger, we cut some spacers with laser cut, and attached the wheel to the motor with bolts. We utilized four copper pillars to attach the rotating tag to the motor.

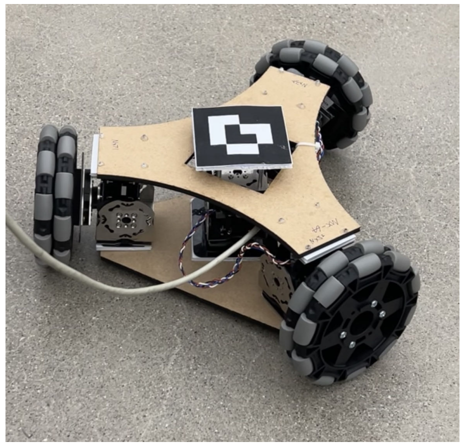

Figure 5: 2022F Green Team's Robot

1.3.2 Software Design

Our code was intentionally organized using several separate files. In this way, we were able to quickly alternate between testing in simulation mode and testing on the real arena. We structured our code with a top-level program (refer to robotSimulator.py in the src folder in the project archive) that coordinated the robot's JoyApp with the waypoint tag streamer. This file combined the other three files into a robot object in code that uses the same navigational algorithms (found in robotPlan.py) with simulation code of the robot (in robotSimIX.py) or with hardware-level code for the physical robot (in robotRealIX.py).

Since we used a three omni-directional wheel car, we could program the car to move in any angle. The control algorithm would allocate different rotation speed to each wheel to achieve that. We mainly referred to this article [1] and this online tutorial [2]. Manual control takes keyboard inputs from the user and moves the robot in fixed step sizes. The controls for the robot are shown below in Table 1.

The autonomous navigation algorithm used proportional feedback control based on the difference of the two sensor measurements to keep the robot near the line connecting the two waypoints. For each step, the robot moved with a velocity that consisted of a component parallel to the line, and the other component perpendicular to the line. The parallel component was held constant, and the perpendicular component was proportional to the difference between the sensor measurements. So, the farther away the robot was from the line, the more it traveled towards the line instead of along the line.

When the robot hit a new waypoint, the tag rotated such that the sensors were both on opposite sides of the line connecting the new current waypoint to the new next waypoint. Then, the robot took three steps along that line, without employing feedback, to prevent either sensor from being off of the end of the new line. Next, the robot began employing feedback.

The software didn't have a recovery behavior for handling situations when the robot misses the next waypoint. Also, the software didn't have state estimation.

The software had a state machine. In the manual state, the robot would respond to the WASD keys to move the robot in steps along the X and Y axis. In the automatic state, the robot didn't respond to WASD key presses, and attempts to autonomously navigate between the waypoints. The space bar starts automatic mode, and the M key starts manual mode.

Input Key	UP	DOWN	LEFT	RIGHT	SPACE	M	R
Action	+x	-x	+y	-y	Auto	Manual	Recover

Table 1: Key Settings for Robot Control

2 Results

All the results for P-Day of Fall 2022 Project 1 can be found on this page. All the videos for the team performance on P-Day can be found here. Two different sets of waypoint positions were adopted on P-Day so we separated all the results into round 1 and round 2. The results including the number of reached waypoints and Root Mean Square(RMS) score are organized in Table 2. Robots from the blue team and the maize team passed the requirements autonomously, whereas robots from other teams passed the requirements manually. The plot of the best performance of every team's robot in terms of the number of waypoints reached and then RMS score are shown in Figure 6.

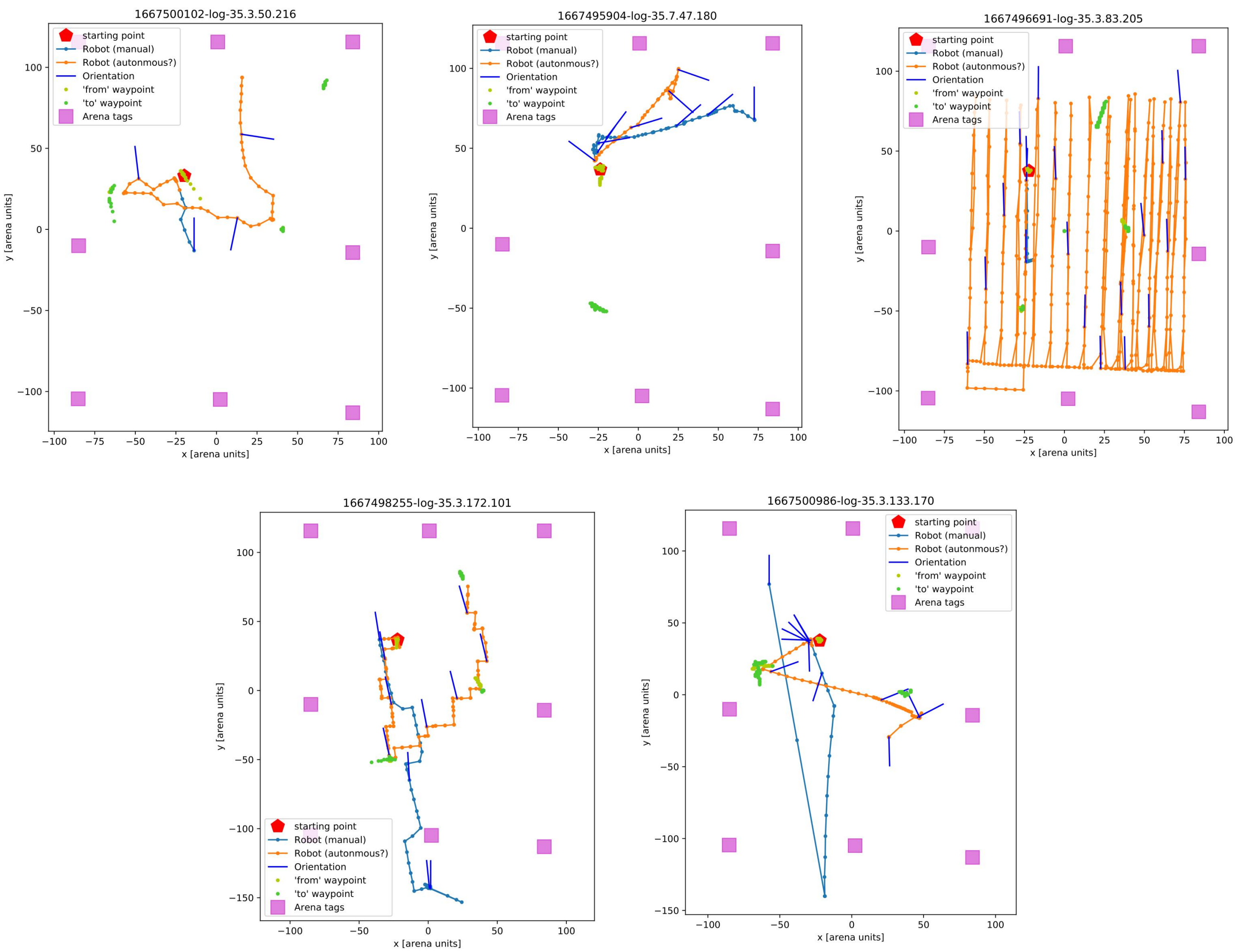

Figure 6: The plot of the best performance of robots from the green, red, blue, maize, and purple teams respectively in terms of the number of waypoints reached and then RMS score. The pink squares represent the boundary tags in the arena. The blue line represents the manual path of the robot, while the orange line represents the autonomous path of the robot. The navy blue line represents the robot's orientation. The green dots represent the computer vision system's depiction of the next waypoint, while the yellow dots represent the computer vision system's depiction of the previous waypoint.

3 Discussion

Team	Pass/Fail	Round	Trial	Num Waypoints Reached(MD+AD)	RMS Score
Green	Pass(MD)	1	1	1	3.07
			2	1	3.91
		2	1	1	1.47
			2	3	7.38
			3	3	3.79
Red	Pass(MD)	1	1	1	8.14
			2	1	3.96
Blue	Pass(AD)	1	1	4	2.45
Maize	Pass(AD)	1	1	2	2.51
			2	4	5.06
Purple	Pass(MD)	1	1	1	6.42
			2	1	9.48
			3	1	5.74
		2	1	2	13.4
			2	1	9.33

Table 2: Summary of P-day Results of Project 1. This table notes down the number of waypoints reached in total and RMS score for all 2 rounds and all the trials. Pass/Fail column denotes whether the robots passed the Pass/Fail requirements manually or automatically. Any trial marked in red indicates successful completion autonomously. (Key: MD = Manually Driven, AD = Autonomously Driven)

3.1 Overall Analysis

The two most consistent robots in terms of event completion times were from the Maize team and Blue Team. The Maize team designed an XY robot with 4 self-made wheels. Nevertheless, since the arena was not flat and the robot lacks a suspension system, it was not moving straight enough. Their particle filter then became very crucial for the success. Of course, they also implemented a recovery behavior that searches the area circularly from a small diameter to a larger one. Another design was from the Blue team. They made an XY plotter machine using rulers. Without the help of any input from the visual system, they simply scanned around the entire arena 4 times to get all waypoints. It also had great accuracy pointing the laser to the white board, since it could not have any rotation at all.

3.2 Our Design

Our group did not finish all four waypoints successfully, but we were able to finish three waypoints consistently after fixing a small bug. In the first round, our robot always adjusted in the opposite direction due to a flipped sign. The reason why we failed getting to the fourth waypoint was due to the parallel error. When our algorithm detected the robot getting to the third point, the robot still had a distance, approximately 4 centimeters, to the line between the last and the third waypoint. Since we did not design any recovery behaviors, we had a very low possibility of finishing it. After the competition, we tried to make the robot move 2 more steps after meeting the waypoint, and we were able to finish the entire arena.

In general, our algorithm used only the sensor input from the professor's visual system, and the accuracy was not as noisy as we thought it would be. By making the robot tag as close as possible to the line, the robot could adjust reliably with respect to the sensor input. Our original automatic algorithm used a particle filter, but we wasted a lot of time on tuning the parameters and finding the reason why it was not working properly. When we decided to switch to the new design using sensors data, we didn't have enough time testing and implementing a recovery behavior.

3.3 Other Designs

3.3.1 Team Maize

The maize team's robot had four wheels that allowed for movement in both the X and Y axis without rotation of the robot. The robot used a particle filter for state estimation to navigate between waypoints. The robot failed to autonomously navigate to the last waypoint autonomously in the first attempt on P-day, and navigated successfully in its second attempt. The maize team could try to improve the reliability of their robot by implementing a rotating tag. For example, the tag could rotate 90 degrees to obtain 2 sets of sensor measurements. These measurements could be used to estimate the orientation of the robot, and thus correct for accumulated errors in orientation. Additionally, the tag could rotate every time a new waypoint is hit to place each sensor on opposite sides of the line, which may improve the reliability of the particle filter.

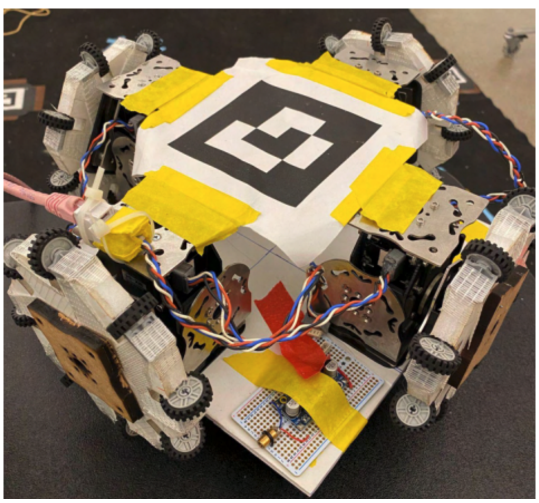

Figure 7: 2022F Maize Team's Robot

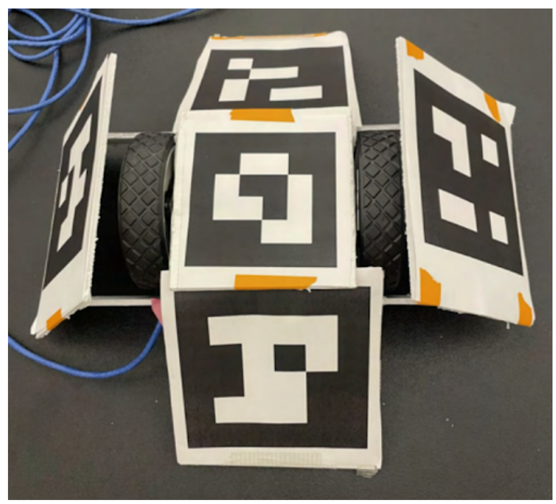

Figure 8: 2022F Red Team's Robot

3.3.2 Team Red

The red team adopted a buggy design that was very similar to the purple team's design. Instead of using a particle filter, the red team set up their own computer vision system to estimate the robot's location by detecting multiple tags that were mounted around the robot. Their algorithm worked by calculating the homography to convert the detected coordinates of four corners to Professor's computer vision system coordinates. However, their robot reached no waypoints autonomously during the P-day. Their robot moved in a seemingly random direction that was keeping away from the direction of the second waypoint. There were two potential reasons accounting for this. One possibility is that their computer vision systems didn't calculate the coordinates of their robot correctly, or the implementation of the homography was incorrect. The other possibility is that the movement command sent to their robot was incorrect, under the assumption that their computer vision system worked correctly. They could consider refining their algorithm from these two respects indicated above for future improvement.

3.3.3 Team Blue

The blue team's robot scanned the entire arena with the tag multiple times until the computer vision system detected that the tag passed over all of the waypoints in order. Occasionally, when the tag appeared to clearly pass over the next waypoint, the computer vision system did not detect that the tag passed over the waypoint. The blue team could try to improve the reliability of their robot by moving the tag at a slower speed, and observing if the computer vision system detects that the tag hit the next waypoint more consistently. However, moving the tag at a slower speed at all times may cause the robot to exceed the 15 minute time limit in some cases. Therefore, the blue team could try to improve both the reliability and speed of their robot by only moving the tag slower when one of the sensor readings is above some threshold.

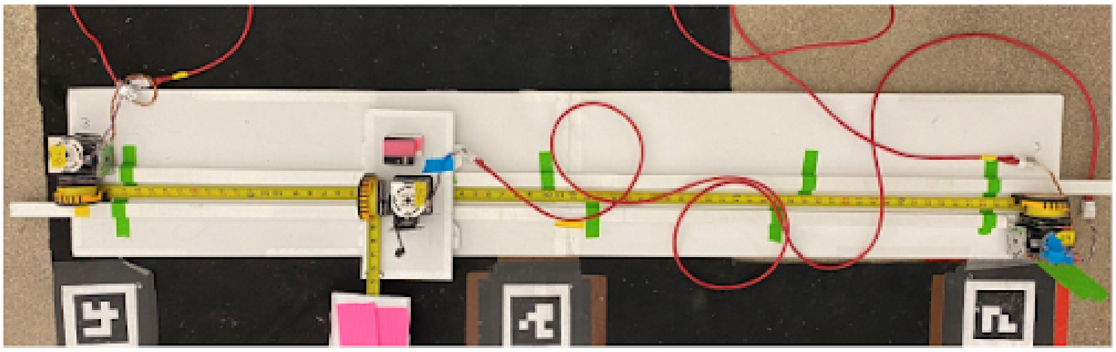

Figure 9: Blue Team's Robot

3.3.4 Team Purple

As shown in Figure 10, The purple team's buggy design allows for moving forwards and backwards in any direction with the facilitation of rotation. The purple team implemented the particle filter to estimate the robot's location and uses the estimated location to navigate between waypoints. Nevertheless, this buggy robot reached zero waypoints autonomously in the first round. Among their three attempts in the first round, the robot approached the second waypoint very closely twice without hitting it. With the advantages of the buggy rotation, they implemented a search plan as well when their system detected that the robot had been very close to the next waypoint. Unfortunately, their robot rotated around the second waypoint repeatedly with an unreliable search plan. They could improve the performance of the robot by refining the search plan, such as using the bowtie search. Moreover, they could also consider fully utilizing the rotation of the tag to improve the robustness of their particle filter, and further ameliorate it by introducing a Bayesian nonparametric outlier model.

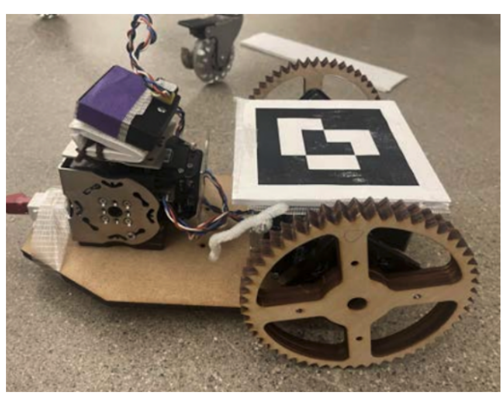

Figure 10: Purple Team's Robot

4 Reference

1. Geometry and Kinematics of the Mecanum Wheel
2. How to use omni directional wheels