MECH 6631

Final Project Report

Amirhossein Dehestani

- **Student ID:** 40225981
- **Contribution:** Attacking scenario algorithm and tracking opponent's robot.

Sourena Morteza Ghasemi

- **Student ID:** 40171622
- **Contribution:** Development of the laser shooting algorithm / Assembling Hardware

Navid Masoumi

- **Student ID:** 40256062
- **Contribution:** Development of the obstacle avoidance/3D print and modeling

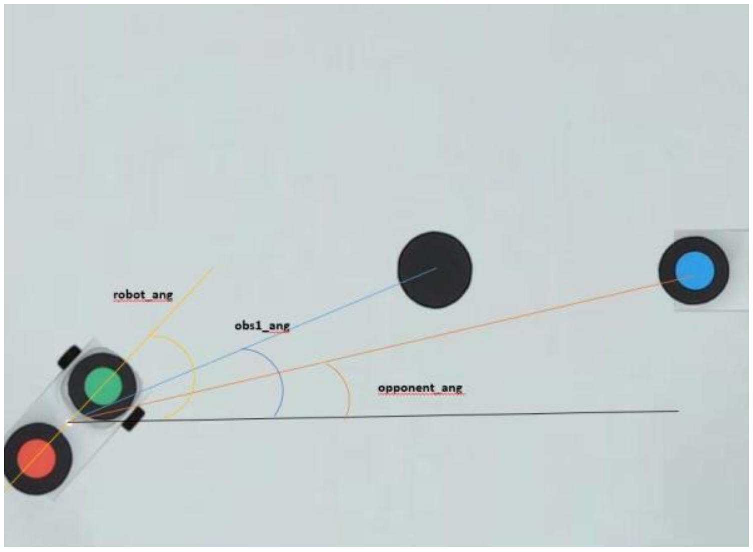
Mohammad Mazidi

- **Student ID:** 40256119
- **Contribution:** Defence scenario/3D design and assembling.

Additional: Building the physical robot (Videos and images are attached in the Moodle.)

The simulation phase of our project includes four major objectives: 1- Attacking scenario 2- Accurate laser shot 3- Obstacle avoidance 4- Defence scenario

For our attacking strategy, we aim to keep the angle between our robot and the opponent robot as small as possible and move towards the opponent to shoot our laser when there is no obstacle between the two robots. To do this, we first need to define the angles between our robot and the opponent robot, as well as the angles between our robot and any obstacles in the environment. All the angles are shown in the picture below:



When the robot's distance from obstacle is more than 120 pixels, we can begin to chase to robot. First, we need to rotate our robot toward the opponent, so when the angle difference is more than 90 deg we have a rotation to turn the robot toward the opponent. When the angle difference becomes between 3 and 90 degrees we are rotating and moving the robot toward the opponent. And in other cases, we just move the robot forward to get closer to the opponent. The point of this process is to keep our robot angle with opponent less than 3 degrees to minimize the chance to miss the shot.

By keeping the angle between our robot and the opponent robot small, we can move towards the opponent in a more direct path and reduce the time it takes to reach them. Additionally, by shooting the laser at the right time when there are no obstacles in the way, we can increase our chances of hitting the opponent and achieving our goal.

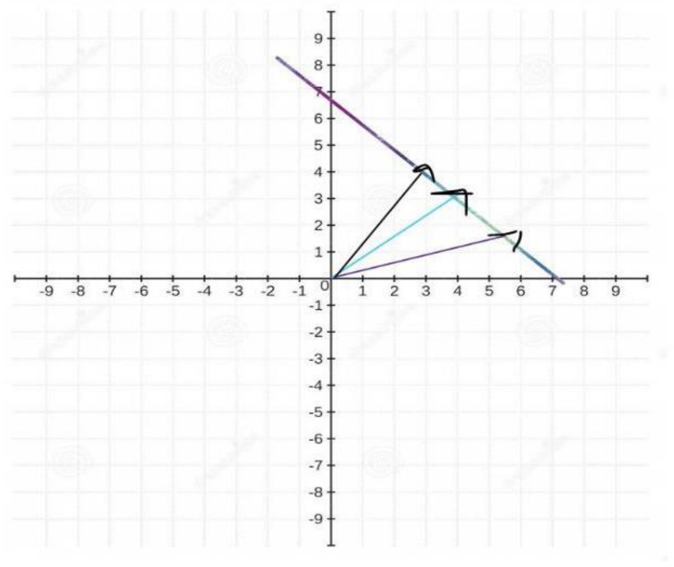
Accurate laser shot is one of the most important part of the project, as the team only has one shot to hit the target. This requires precise control and accuracy, and our team developed an algorithms to ensure that the laser hits the target. In the plot below, three vectors can be seen pointing to the obstacle, attacker robot, and defender robot. The obstacle is in the middle and the angle between our robot and the obstacle is the same as that between our robot and the defender robot. This information is critical for calculating the attack angle, which is used to determine the optimal path for the attacker robot to take while pursuing the defender robot. To calculate the attack angle, we have:

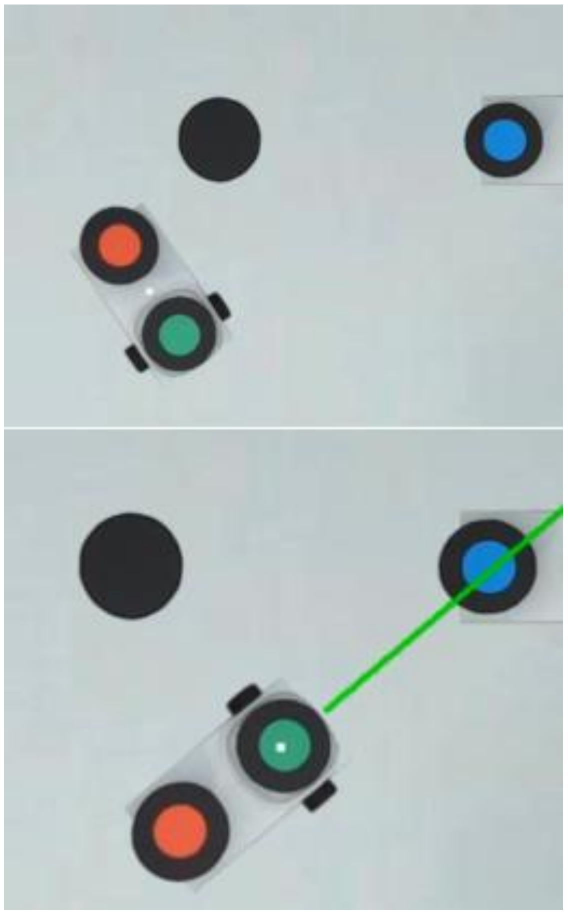
att_ang = robot_ang - opponent_ang;

So, we came up with an algorithm that whenever these three objects are align with each other, it checks that if there is an obstacle between. Therefore we use:

$$\sqrt{x^2 + y^2} = R^2$$

Also, it should be mentioned that if the size of the obstacle is between the sizes of two robots it means that obstacle is in the middle so robot should not shoot its laser.





One of the main challenging parts of this project was detecting the positions of the obstacles. In this regard, we used the given materials from the course to process two snapshots from 1st and 10th frame and recognize the positions of the obstacles by comparing these two frames. In this method, first we use the 1st and 10th frame as inputs of the simulation for the code. Then after converting them to a gray scale image, the binary version of them could be extracted. Using "Label image "function we can give some labels to all of the shapes (including robots, obstacles, etc.). In addition, the centers of each label could be found using Centroid function given in the materials. By having each cluster and the center of each cluster we can easily understand which ones the obstacle and which ones are the robots by comparing the 1st and 10th frame image. Those which remained in their position are obstacles and the rest are the robots. With this method, we would have the center point position of each obstacle, and they could be used in later parts of the project e.g. obstacle avoidance.

For the defence scenario, we were trying to come up with a way to escape from an attacker in defense mode. For that, we had various ideas. For example:

1. The defender can try to escape the attacker by rotating around the battlefield trying not to receive laser shots.
2. The defender can try to stay behind the obstacle without moving, hoping that the attacker cannot take a shot.
3. The defender can try rotating around an obstacle to avoid taking a shot from the attacker.
4. The defender can try rotating around an obstacle while tracking the movements of the opponent to switch the direction of rotation based on the attacking angle in order for it to make sure that the obstacle is always in between.

Among the mentioned defending criteria, we believe that rotating around an obstacle and preferably having the obstacle in between all the time would be the best option.

In order to rotate around an obstacle, we would have to first find the nearest obstacle and then approach it. For that we started by calculating our robot's angle in the code and find the center of the two colored circles on our robot and use the coordinates of these two points to find out the angle. Then we would calculate the angle between our robot and the nearest obstacle using the exact same methods and function. By having two angles we can rotate our robot until we are facing the obstacle and then we set the speed of the robot's wheels to approach the obstacle.

Once we get close enough to the obstacle (a distance of 130 pixels), we would want the robot to rotate so that the direction angle is tangential to the obstacle. That happens when the angle between us and the line connecting the obstacle to the center of our robot becomes 90 degrees. However, in our code when we set the angle to be 90 degrees, the result was not the best due to reasons that will be covered later.

So, we tried to keep the angle between 70 and 90 while we were rotating around the obstacle. For rotating around an obstacle, we set the speed of the wheels at 2000 and 1700 so that it would follow a circular trajectory while avoiding the obstacle. With that we would rotate around the obstacle to escape the attacker and to avoid collision at the same time.

For making a decision on the direction of rotation, we were thinking that it would be the best to change the direction of rotation based on the attacking angle, so when the other robot is attacking ours from right side we would want to rotate CW and if it's attacking from left side we would want to rotate CCW.

Our first approach to obstacle avoidance was based on the fact that the robot should halt anytime it was reasonably close to an obstacle, rotate till the angle between the obstruction and our robot was between two adjusted angles set by us, and then start moving around the obstacle until it was passed. This approach was relatively a good fit and we obtained results, but it was not consistent in various situations. Therefore, we decided to take another route in this matter, and we used a similar method to our defence strategy. In this situation, robot checks the distance to the obstacle and if it is less than our set number which is 110pxs, it would act. If the difference between robot angle and obstacle angle is less than 60 degrees, robot would be stopped and rotate till the suitable angle. For angles higher than 70, the rpm of the left wheel would be lower to rotate.