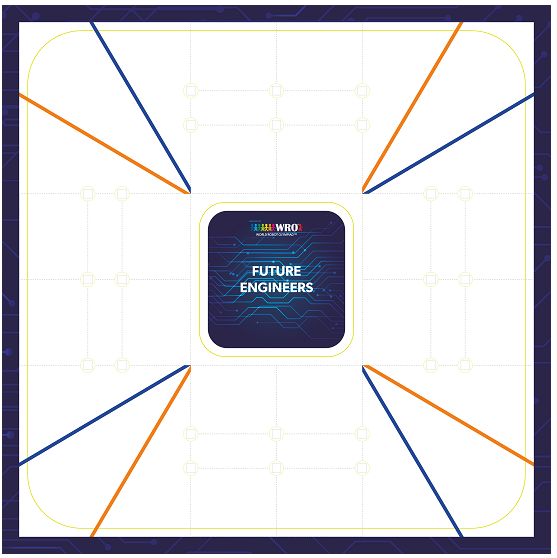
Open Challenge Algorithm:



The Open Challenge presents an interesting and exciting opportunity to push the limits of camera-detection and speed. We utilize this to our advantage and rather than using various sensors (ultrasonic and imu) whose values have to be passed down and filtered, use a single camera and a color sensor to detect the lines.

Detection [Black Walls]

We use a camera to detect the black walls on either side [color based] and accordingly set the direction of the robot. Using OpenCV, we ensure that only 1 black wall on either side (left and right) can be detected. Furthermore, we crop out the frame in front to omit detection of the front black walls. We compare the distance between either walls from the center and accordingly calculate the error distance between the walls, thus understanding the robot’s offset from the center. If at any point, both walls cannot be detected, the robot shall automatically turn enough so that it can detect the other wall. In order to account for this, we implement a PD [proportional derivative] logic. This is acted upon the steering servo which accordingly changes the direction of the steering wheels.

Corner Turning

By using the difference between the walls to set the direction, we eliminate the need for any non-sensor based logic. Upon reaching the corner, the camera can no longer detect one of the walls. As a result of this, in order to maintain its center position, the robot turns towards the side where the wall cannot be detected. As it keeps turning, it manages to find the equality between both walls again and continues using the PD algorithm to maintain a center position.

Round Count

In order to understand how many rounds have elapsed, we use the color sensor to detect the corner lines. In order to omit any sort of detection error, we look for both blue and orange. After detection, we disable the sensor for some time so that multiple detections do not occur. After counting 24 such lines, we establish that 3 rounds have been completed. [2 detections per corner, 4 corners, 3 rounds].

Obstacle Challenge Algorithm:

The Obstacle Challenge, provides a challenging problem to which several algorithms must be implemented to ensure a consistent success rate. We take reference from both the black walls, and the obstacles themselves, in order to find the correct path to navigate through the walls and obstacles without any failure.

(Clockwise):

Detection [Red & Green Obstacle]

If both obstacles are detected, we establish which one is closer by comparing the Y value and accordingly take reference from the closer one. After that we use the algorithm of either the green or red obstacle.

Detection [Red Obstacle]

When the red obstacle is detected, we immediately take reference from it. It is detected by using a combination of both HSV values and a minimum area requirement. A boundary box is then drawn around it. Our field of view from the camera is 1020 x 300. We implement a number of steps to ensure consistent maneuvering. Firstly, we check the distance between the camera and the obstacle by comparing the Y value. The higher the value is, the closer it is to the robot. [If the Y value is too high, we reverse the robot until the Y value reduces enough]. According to the distance between the robot and the obstacle along with the X value of the obstacle, we turn it accordingly to the left. My multiplying the Y value and the 1020-X value by a constant amount (j), we accordingly turn the robot. The distance between the obstacle and turning effect is inversely proportional. We tune the ‘j’ value so that the turning effect is successful. This turning effect uses a PD Algorithm.

Detection [Green Obstacle]

When the green obstacle is detected, we immediately take reference from it. It is detected by using a combination of both HSV values and a minimum area requirement. A boundary box is then drawn around it. Our field of view from the camera is 1020 x 300. We implement a number of steps to ensure consistent maneuvering. Firstly, we check the distance between the camera and the obstacle by comparing the Y value. The higher the value is, the closer it is to the robot. [If the Y value is too high, we reverse the robot until the Y value reduces enough]. According to the distance between the robot and the obstacle along with the X value of the obstacle, we turn it accordingly to the left. My multiplying the Y value and the X value by a constant amount (j), we accordingly turn the robot. The distance between the obstacle and turning effect is inversely proportional. We tune the ‘k’ value so that the turning effect is successful. This turning effect uses a PD Algorithm.

Detection [No Obstacle]

If no obstacle is detected, we immediately switch to the wall following logic and ensure constant distance between the walls.

Detection [Black Walls & No Obstacle]

We use a camera to detect the black walls on either side [color based] and accordingly set the direction of the robot. Using OpenCV, we ensure that only 1 black wall on either side (left and right) can be detected. Furthermore, we crop out the frame in front to omit detection of the front black walls. We compare the distance between either walls from the center and accordingly calculate the error distance between the walls, thus understanding the robot’s offset from the center. In order to account for this, we implement a PD [proportional derivative] logic. This is acted upon the steering servo which accordingly changes the direction of the steering wheels.

Detection [Singular Black Wall]

If at any point, both walls cannot be detected, the robot shall automatically turn enough so that it can detect the other wall, due to the PD logic.

Round Count

In order to understand how many rounds have elapsed, we use the color sensor to detect the corner lines. In order to omit any sort of detection error, we look for both blue and orange. After detection, we disable the sensor for some time so that multiple detections do not occur. After counting 24 such lines, we establish that 3 rounds have been completed. [2 detections per corner, 4 corners, 3 rounds].

Detection of Parking

During the 3 rounds we take, we ensure that the red obstacles and magenta parking walls are not confused using a special logic. Since the HSV values are very similar, we also implement a minimum area logic. The area of the walls are less compared to the obstacles and are thus, omitted while the rounds are going on. After completion of the rounds, we reverse the logic and ensure that only if the area is less than a certain value, is the parking detected.

Parking

Due to our great turning radius, parking becomes very simple in our case. By using our camera to ensure we are in the middle of the walls, we then ensure that the back of the robot is facing the outer wall of the mat (not of the parking walls). We then bang against the wall to ensure a fixed position. Then, we use a fixed set of a combination of 3 turns to park.