Development of an Autonomous Robot that efficiently disinfects with 3D point clouds for mapping and Deep Learning for object detection

Project H14; Laasya Chukka

Objectives + Goals

Due to the recent outbreak of COVID-19, it is now important to minimize physical contact through autonomizing certain tasks.

One important job, disinfecting and spraying down closed and open spaces, can be done through new advancements in autonomous vehicles. Additionally, outside of this plausible application, autonomous vehicles can be deployed in various settings to do menial tasks, which can greatly improve productivity.

My goal is to implement an autonomous robot that can cover a space efficiently, and use object detection to determine whether it should disinfect obstacles or not.

This would first include creating a suitable environment with various indoor objects. Then I intend to create algorithms that can effectively navigate these obstacles using sensory input. A neural network will be implemented for object detection to determine whether the object should be “sprayed” or not.

Software/Materials

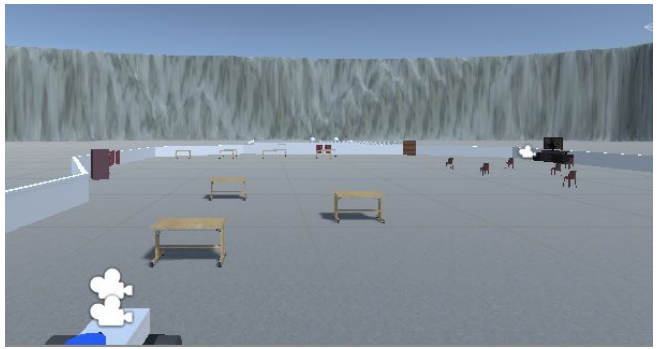
I am using the MITLL RACECAR-MN Libraries, an open source platform that consists of a basic framework for implementing a standard cleaning vehicle, which I will reference as “c-robot.”

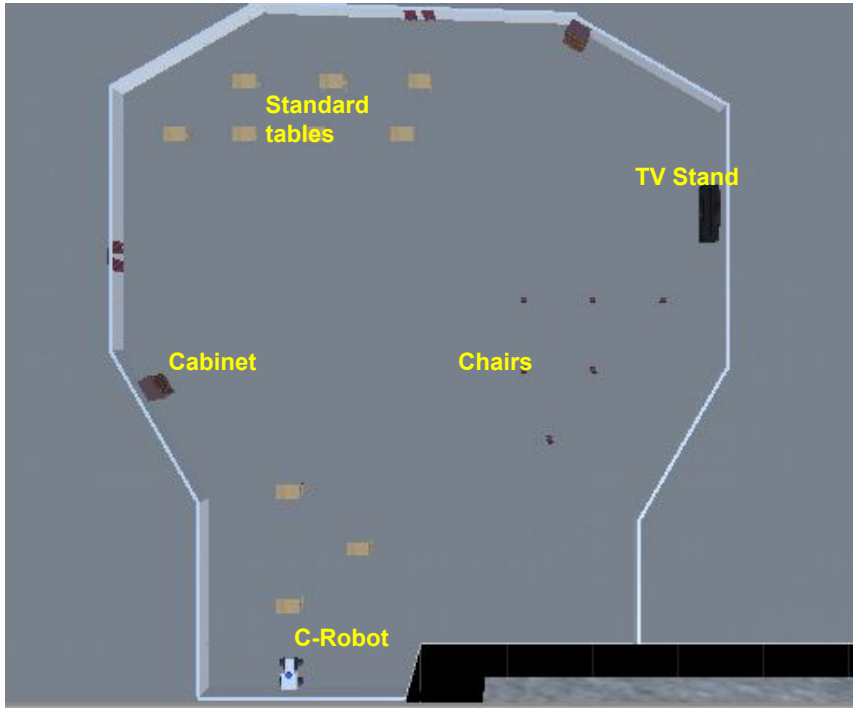
This c-robot is equipped with a color camera, depth camera, and LiDAR. The vehicle can move with various speeds, and has 180 degree wheel motion. The environment is based in Unity Engine, something I used to create my own simulation.

I also used the TensorFlow ResNet convolutional neural network for identifying objects.

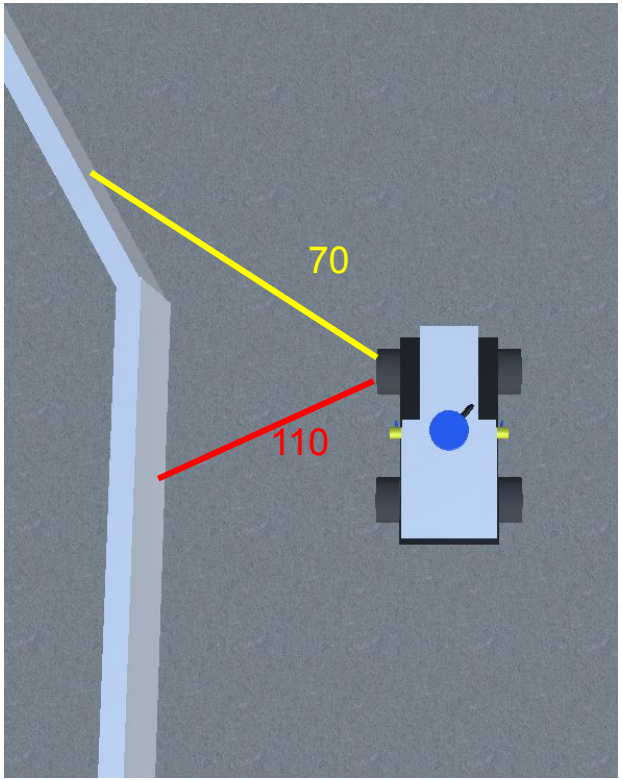
The Simulation

In order to test my code, I created my own environment on Unity, which simulates an indoor space. The environment is standard, and is enclosed with walls. In it are various obstacles the c-robot must navigate.



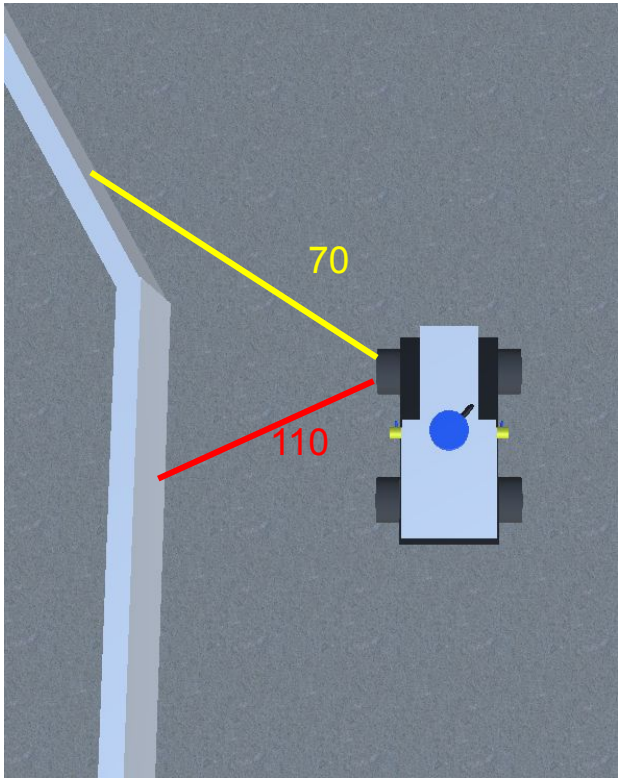


State 1: Wall Following



In order to navigate the walls, I created an algorithm that allows the vehicle to “follow” either side. Using the LiDAR, I get the distances from 70 and 110 degrees. The reason why I get these two distances is to account for changes in the wall’s angle (based on the diagram, since the yellow ray’s distance is greater, we know that the vehicle needs to turn left).

State 1: Wall Following (2)



Left_difference = 70_dist - 110_dist

Left_distance = Closest distance from 10 to 60 degrees

Value = -1* Left_Difference + (60 - Left_distance)

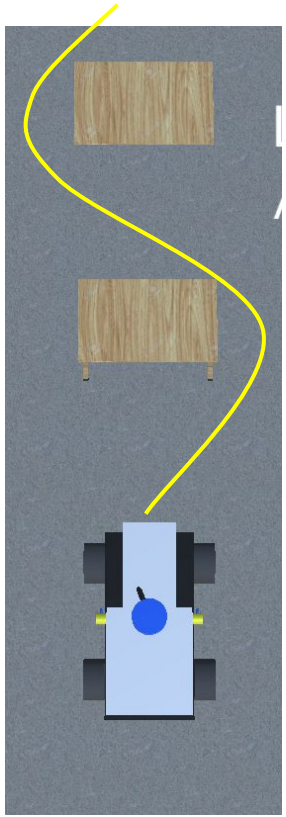
This value is then proportioned between a turn angle from -180 to 180 degrees.

The 60 is there to account for maintaining a consistent distance between the wall and the racecar (to avoid collisions)

The speed proportional to the closest distance in front of the c-robot, to again, avoid collisions. The closer the vehicle was to the wall, the slower it would move.

I tested various degree windows besides 70 and 110 degrees. A larger window, such as 50 and 130 caused the c-robot to make abrupt turns, and a smaller window caused overly small turns that didn't give the vehicle time to adjust to sharper turns.

State 2: Navigating the objects



In order to navigate obstacles, I implemented an algorithm that can “slalom” through these objects using the LiDAR sensor.

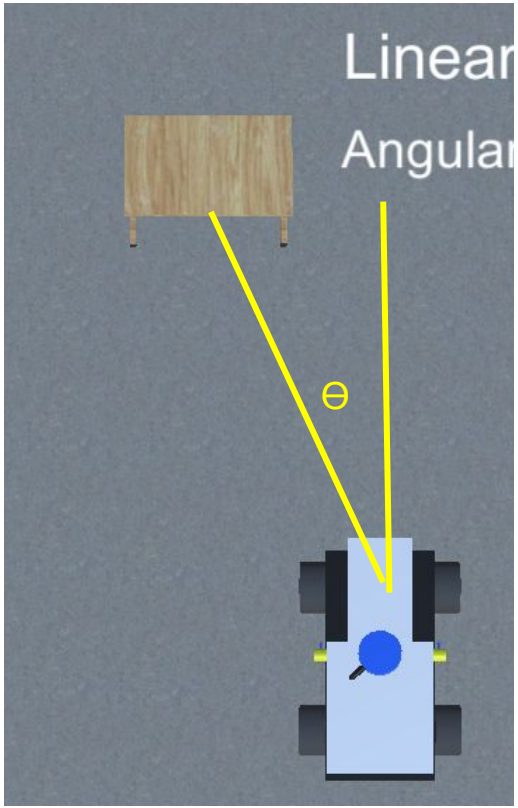
The code consists of multiple states based on proportional control in relation to the angle and the distance from the vehicle.

1st state: the c-robot “aligns” itself with the object through angling itself with the object’s center.

2nd state: Once a certain alignment, along with distance is reached, the vehicle takes a direct right (or left) turn around the object proportional to the angle the obstacle is at (full turn at 0 degrees to no turn at 90 degrees).

For the next object, the same code in the opposite direction until there is no sprayable object detected. Then, the vehicle shifts back to State 1, wall-following mode.

State 2: Navigating the objects (2)



Alignment:

Using LiDAR to get Θ of closest distance from object.

Based on this angle, adjust the turn power proportionally until Θ = 0 degrees is reached and a certain distance is reached.

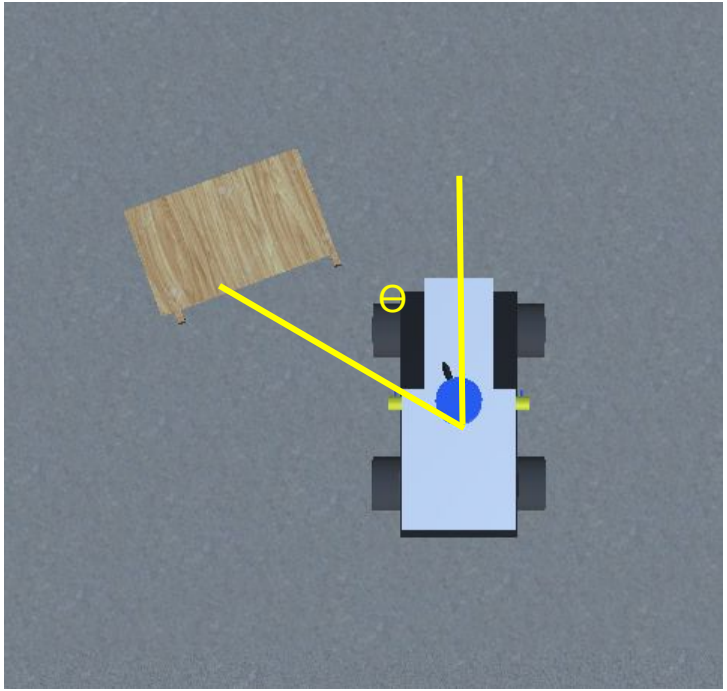
Θ: 0 → 80

Turn: 0 → 1

Θ: 0 → -80

Turn: 0 → -1

State 2: Navigating the objects (3)



Making the turn:

Using LiDAR to get Θ of closest distance from object.

Based on this angle, adjust the turn power proportionally until Θ = ± 90 degrees is reached.

Θ: 0 → -90

Turn: -1 → 0

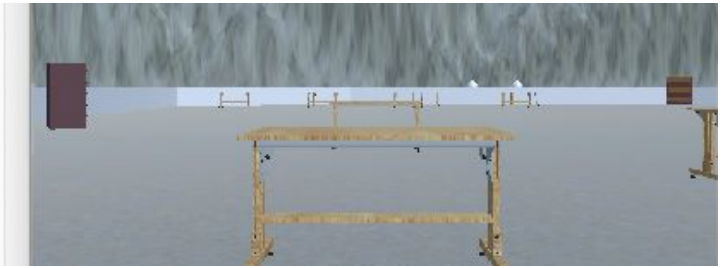
Θ: 0 → 90

Turn: 1 → 0

Object Detection

In order for the c-robot to determine whether to spray (or navigate through) the objects or not, the vehicle uses the color camera to capture and process the image through the ResNet convolutional neural network. This code is continually run in State 1, and in this case, a table or a chair triggers a switch to State 2.

```
ompiler flags.
2021-03-04 17:55:30.137525: I tensorflow/compiler/mlir/mlir_graph_optimizatio
n_pass.cc:116] None of the MLIR optimization passes are enabled (registered 2
)
Predicted: [('n03201208', 'dining_table', 0.118010245)]
[('n03201208', 'dining_table', 0.118010245)]
104.77388671875 60.48988891601562
44.28399780273438 74.03257751464844 -1
(410, 640)
line following state left
Predicted: [('n03201208', 'dining_table', 0.12630954)]
```



Results/Drawbacks

While my algorithms worked in ideal cases, the robot struggled to navigate tighter spaces or when various obstacles were too clustered. For example, the vehicle was able to navigate the tables in the front, but couldn’t effectively align itself with the space given, and navigate the narrow turn and path with the tables in the back. Additionally, the vehicle wasn’t able to slalom in between chairs because they were smaller and harder to detect using the neural network or the LiDAR.

Additionally, the neural network could also only effectively identify single objects in an image, something not practical outside of a simulator.

Future Work

In the future, I hope to make more revisions by testing in various environments (possibly in the real world with the right resources) and using more sophisticated algorithms to make the vehicle more adaptable.

Specifically, I would like to use image segmentation to improve upon the neural network and make it possible to identify and navigate obstacles quickly. I would also like to use a PID control system for my vehicle, which can make movements more efficient and accurate for my wall following code.

Finally, I want to implement an actual spray to track how much of the environment is covered, which will give me more insight on how to traverse the area efficiently.

Works Cited

[1] Kiam Heong Ang, G. Chong and Yun Li, "PID control system analysis, design, and technology," in IEEE Transactions on Control Systems Technology, vol. 13, no. 4, pp. 559-576, July 2005, doi: 10.1109/TCST.2005.847331.

[2] Maurya, D., Gohil, M.K., Sonawane, U. et al. Development of Autonomous Advanced Disinfection Tunnel to Tackle External Surface Disinfection of COVID-19 Virus in Public Places. Trans Indian Natl. Acad. Eng. 5, 281–287 (2020). https://doi.org/10.1007/s41403-020-00141-7

[3] H. L. N. N. Thanh and S. K. Hong, "Completion of Collision Avoidance Control Algorithm for Multicopters Based on Geometrical Constraints," in IEEE Access, vol. 6, pp. 27111-27126, 2018, doi: 10.1109/ACCESS.2018.2833158.

[4] Rastelli, Joshué Pérez, and Matilde Santos Peñas. “Fuzzy Logic Steering Control of Autonomous Vehicles inside Roundabouts.” Applied Soft Computing, vol. 35, Oct. 2015, pp. 662–669, 10.1016/j.asoc.2015.06.030. Accessed 26 June 2020.

[5] Geneva; World Health Organization; 2020. (WHO / 2019-nCoV / Disinfection / 2020.1). in English, Russian, Arabic, Chinese | WHO IRIS | ID: who-332096 Responsible library