3D Map Fed Robot

Mentor
Chandran Nandkumar

Members
Viren Varma | Pranshu Shukla

NIT Karnataka
IEEE Student Branch

PISTON

IEEE
Advancing Technology for Humanity

Introduction

Autonomous Localization and the future of robotics

- In the field of robotics and automation systems - localization, navigation and path planning algorithms are some of the most interesting and exciting topics which are also currently under research. Especially in military applications, we often come to situations wherein we need to localize our current location and henceforth perform navigation and other operations. These problems involving self-localization is known as Kidnapped Robot Problem and in this project, we aim to present one solution using feature maps.

Aim

- The aim of this project to use implement self-localization and path planning on a known environment using the concept of feature based and particle maps, that is, given the floor map and the location of certain features of the room in the map – we intend to determine the location of our bot and finally perform path planning using to reach our desired destination or target.

Methodology

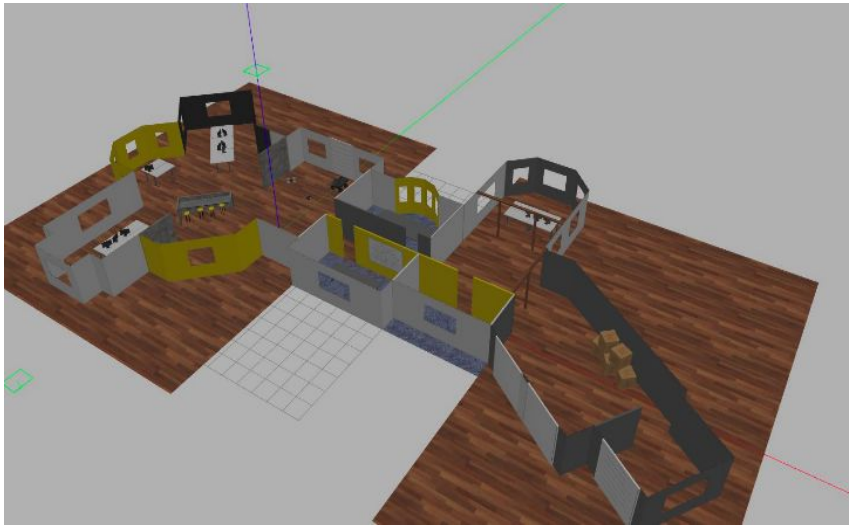
Simulation and Testing

- For the purpose of implementing this project and simulating the results, we used the Robot Operating System (ROS) framework. It is an open-source development platform for robotics comprising a multitude of packages and software's for building, writing and testing systems and processes.
- Keeping in mind the latest developments and package statuses in ROS versions - we opted for ROS Melodic Morenia, compatible with Ubuntu 18.04.

Methodology

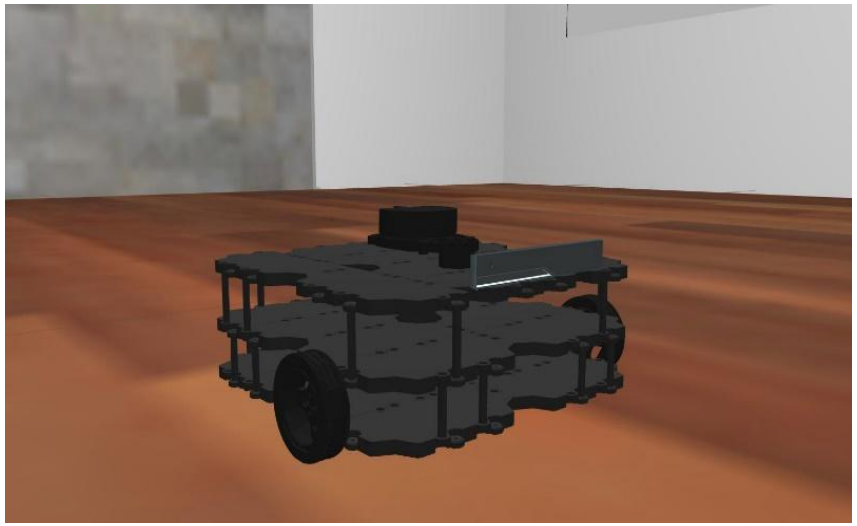
Mapping and Environment Setup

- For the purpose of simulating the project, we chose to use an open-source map “Office”. To meet with the objectives of the project of self - localization, we designated each room with different obstacles and identified them with names from Letters A to H. This was done to establish an ease of self-localization and have better results.



Methodology

Robot Selection



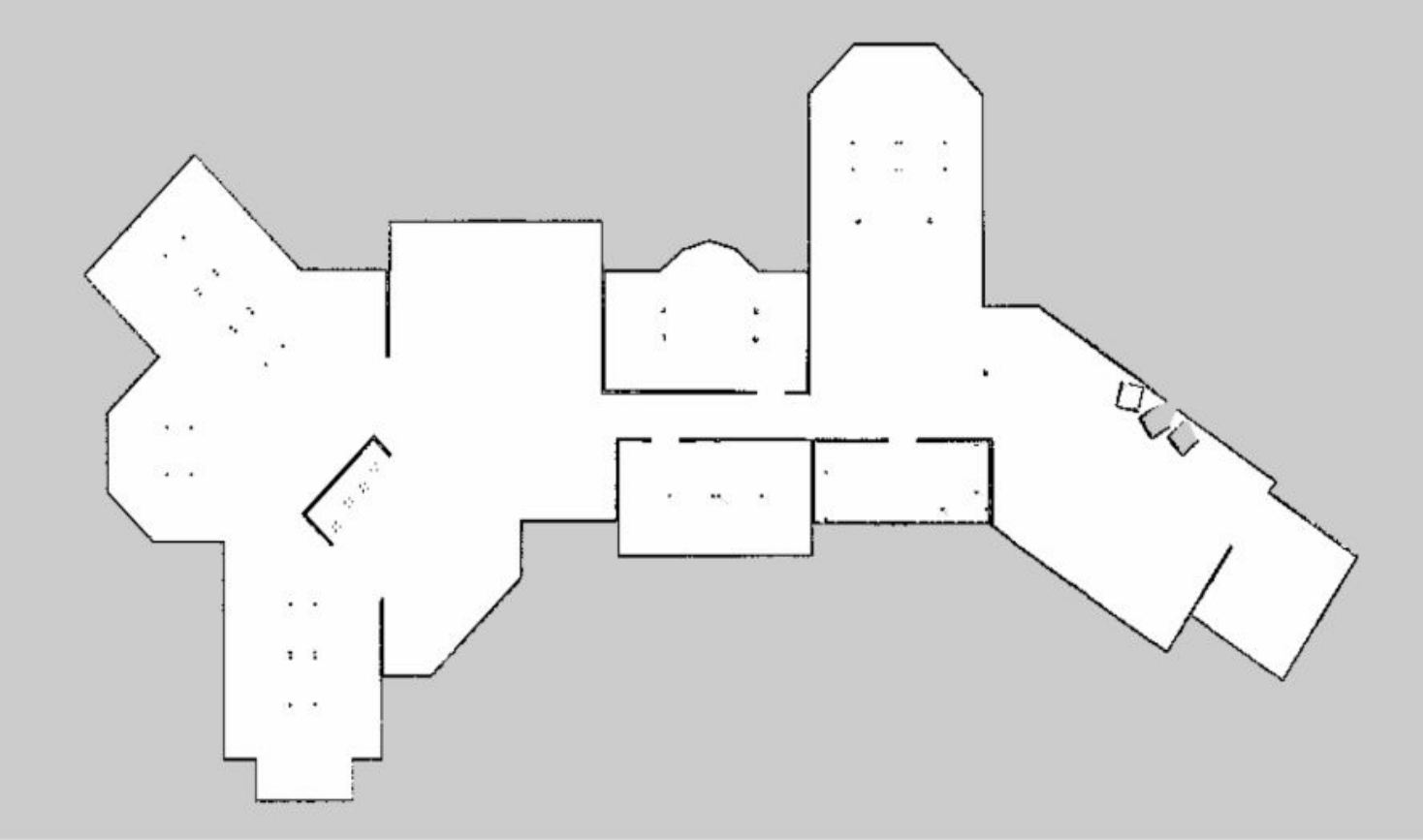
- TurtleBot3 waffle is a small and compact robot from the Turtlebot3 series. The TurtleBot3 can be customized as per requirements into various applications by reconstructing the location and behaviours of the input sensors. The Turtlebot3 package is also equipped with SLAM (simultaneous localization and mapping), Navigation and Manipulation etc.

Methodology

Gmapping and Navigation Track

- Once the robot has successfully been able to recognize its surroundings, we wanted our robot to navigate out of the surrounding environment and reach the target location set initially. To achieve this, we needed to have the map of the environment ready beforehand. A map is the representation of the environment where the robot will be operating.
- Alot of various SLAM techniques are available for such purposes as Gmapping, Hector mapping, Cartographer, Rtab mapping etc. For simplicity and robustness of the project, we opted for “Gmapping ''. This package provides laser-based techniques and is compacted as a ROS node called slam_gmapping using which a 2-D occupancy grid map can be generated.

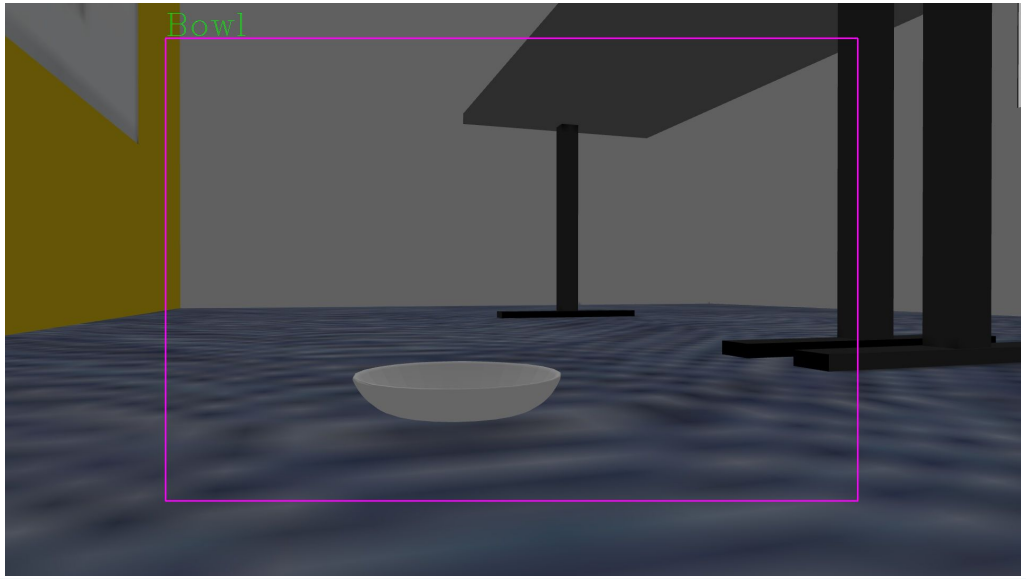
Map generated by Gmapping



Methodology

Image Processing

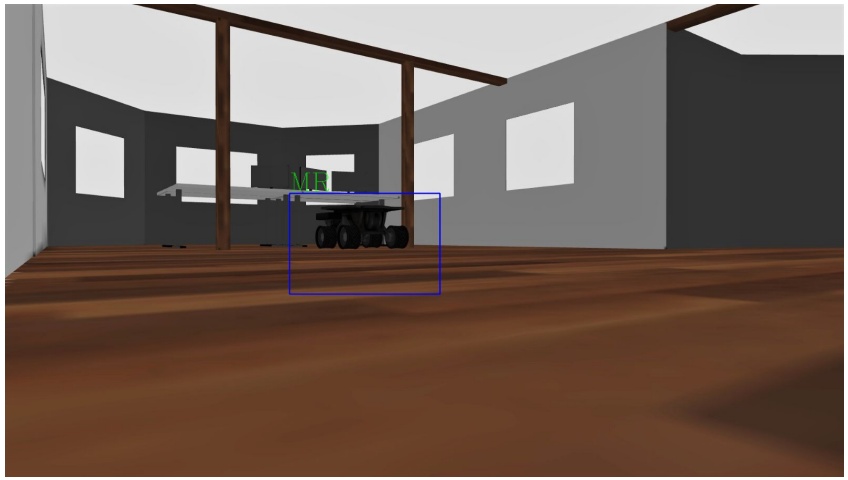
We used opencv and the os library in creating the tracker and bounding to select the object manually and the using OS library, save the object's newly clicked images in a file to be later called to train the model again to update the weights



Methodology

Image Processing

Opencv was extensively used in the project starting from taking in the video feed from the turtle bot camera and using frame by frame images of the while loop to first resize and reshape to make it compatible with CNN's input layer to classify which object does the image contain to further use Opencv to call Custom build Cascades previously built using opencv and Haarcascade builder GUI to use opencv's *detectMultiScale() function which tells us the x,y coordinate of the top left part of the area of the object detected in the frame as well as the width and height of the rectangular area.*



Methodology

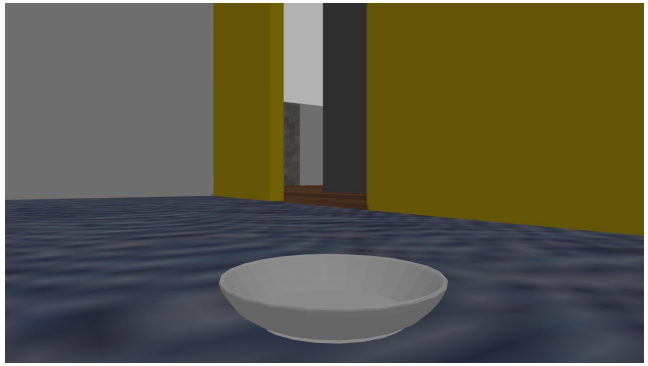
Object Detection in CNN

our custom CNN which could classify which object is present in our frame. So after a lot of trial and error with neural network architecture (where we first made a model too complex for the images which saw massive overfitting and further we reduced the complexity so that there is just enough overfitting that can be countered by adding more varied data later by our scanner which I will explain further). So once we got the 96% accuracy on our training data we stopped

```
_________________________________________________________________
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1844: UserWarning: `Model.fit_generator` is deprecated
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/5
30/30 [==============================] - 170s 5s/step - loss: 20.1827 - accuracy: 0.4388 - val_loss: 0.4710 - val_accuracy: 0.8500
Epoch 2/5
30/30 [==============================] - 12s 415ms/step - loss: 0.7428 - accuracy: 0.8138 - val_loss: 0.2159 - val_accuracy: 0.9000
Epoch 3/5
30/30 [==============================] - 9s 311ms/step - loss: 0.2645 - accuracy: 0.9159 - val_loss: 0.1931 - val_accuracy: 0.8500
Epoch 4/5
30/30 [==============================] - 9s 298ms/step - loss: 0.0833 - accuracy: 0.9694 - val_loss: 0.0685 - val_accuracy: 0.9750
Epoch 5/5
30/30 [==============================] - 9s 299ms/step - loss: 0.0385 - accuracy: 0.9847 - val_loss: 0.1096 - val_accuracy: 0.9000
```

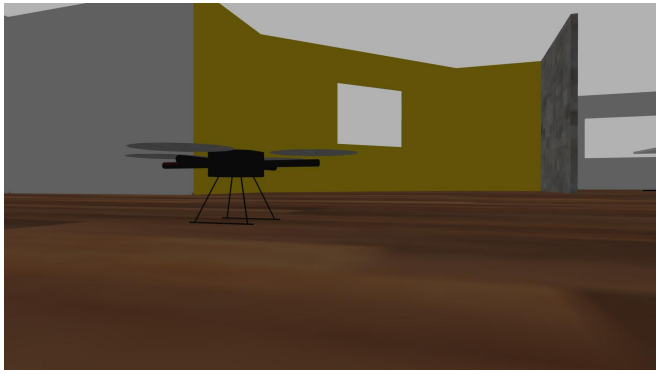
Classes Considered

The four Object Classes that we took

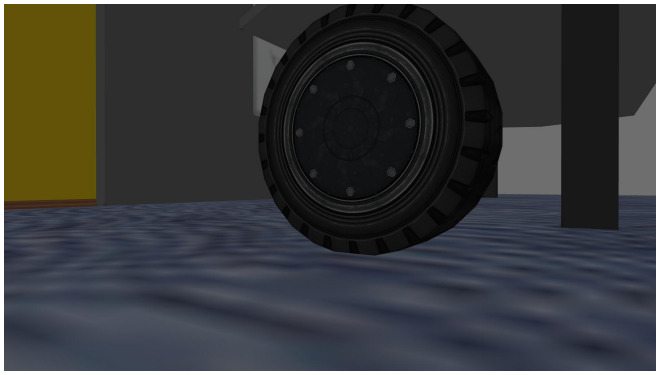


Class - Bowl

Class – Mars Rover



Class - Quadcopter

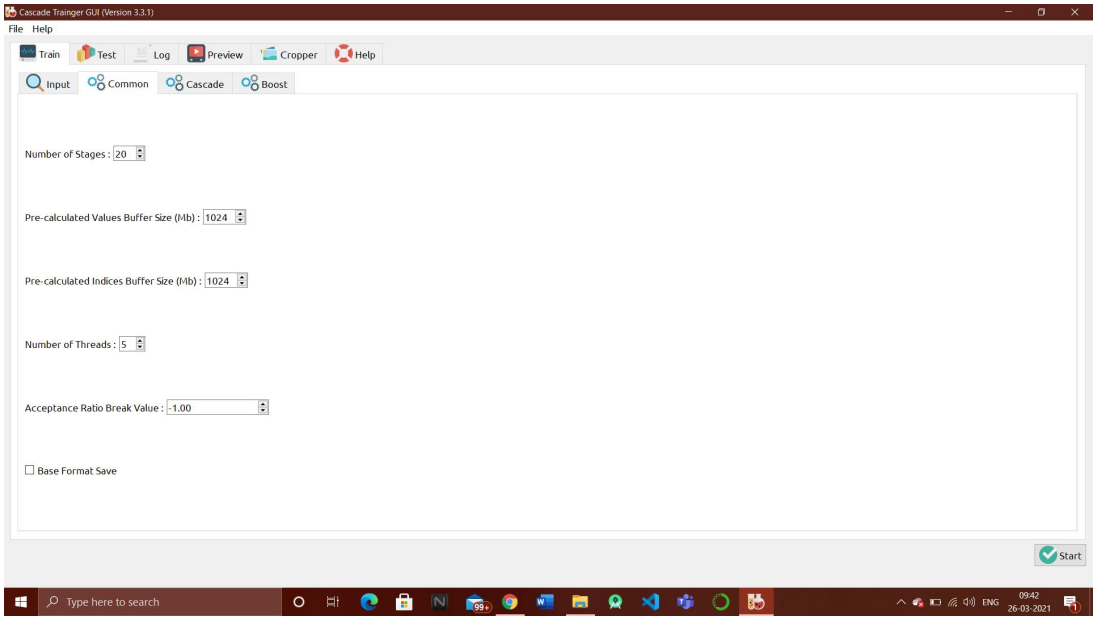


Class - Tyre

Methodology

Cascades

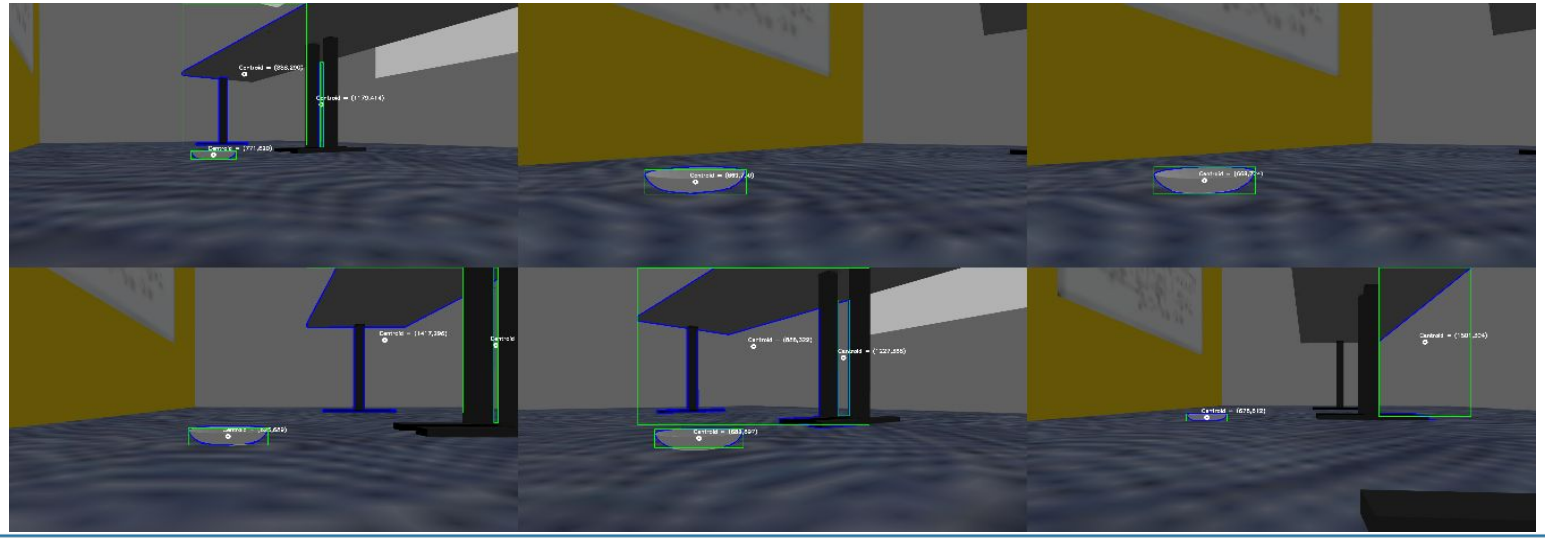
Four custom built Haarcascascades were used by us for which we used the Cascade_Trainger_GUI to train each object's cascade on a bunch of positives and negatives most of the negatives were either pictures of other objects (so as to avoid clashes in detection) or black empty room spaces.



Methodology

Centroid Detection

For centroid detection we purely used Opencv library to first detect the contour (As each object as a unique contour with respect to its surroundings) of each image followed by circumscribing that contour by rectangular box to find it's mid-point.



Methodology

Depth Readings

- After determining the object classes from the CNN and Cascades, we determine the centroid of the identified object.
- After determining the pixel coordinates of the centroid of the object in the image frame, we take the image readings from the depth topic - “/camera/depth/image_raw”
- The feed in this topic is encoded in as “32FC1”. Each pixel in this encoded image is described by a single 4-byte floating point number which represents the (estimated) distance to the nearest obstacle in that direction, measured in meters.

Results

CNN and Cascades

- Our CNN had an accuracy of 96% but to get to that reading a lot of trial and error needed to be made to reach that accuracy in terms of Neural Network Architecture for which Tensorboard was used to monitor the accuracy of detection with each epoch.
- where we first made a model too complex for the images which saw massive overfitting and further we reduced the complexity so that there is just enough overfitting that can be countered by adding more varied data later by our scanner
- At Every trial and error situation we used Google Colab's free GPU and Tensorboard to analyze the strength of our model

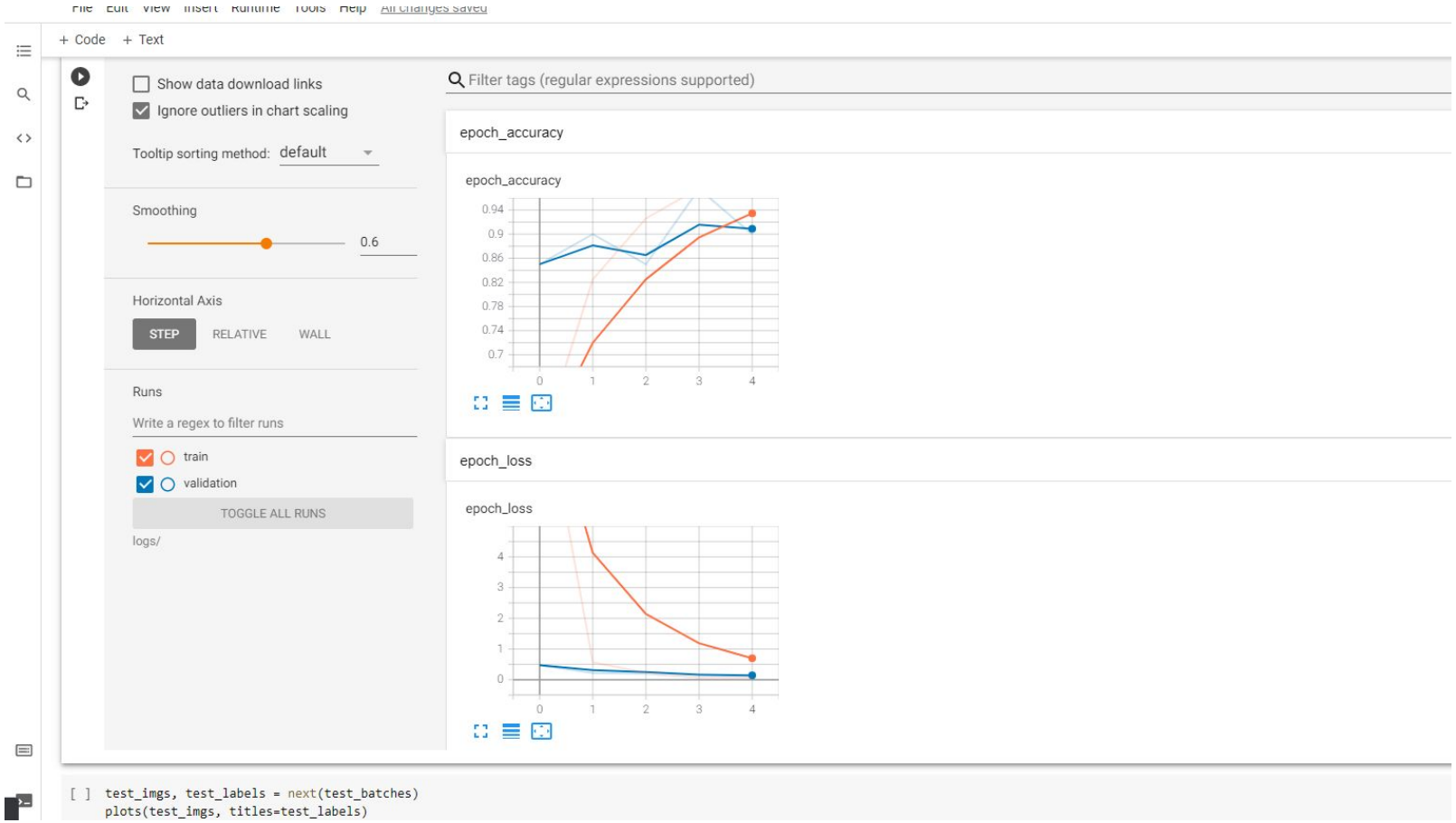
Results

CNN and Cascades

- Our Combination of CNN and Haarcascade showed promising results but assuming the combination fails to detect an object correctly a due to lack of varying data in CNN we have also created a tracker that takes the images where we have manually inputted the object
- And store it in a folder to create a separate dataset to further train our model to update it's weights and increase its accuracy

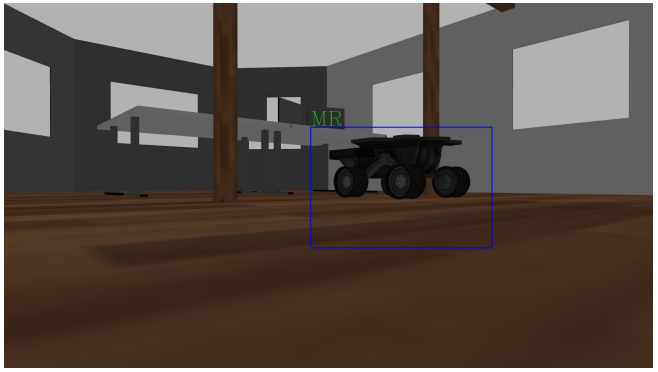
Results

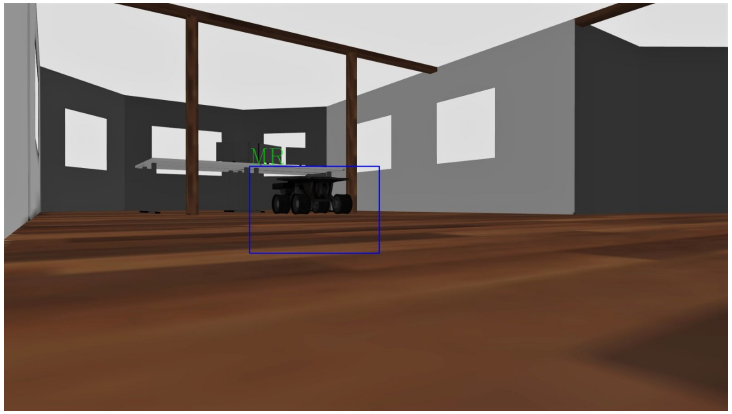
CNN and Cascades

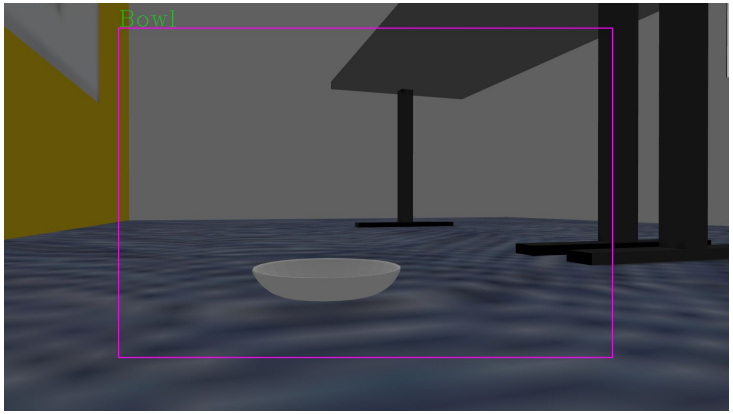


Results

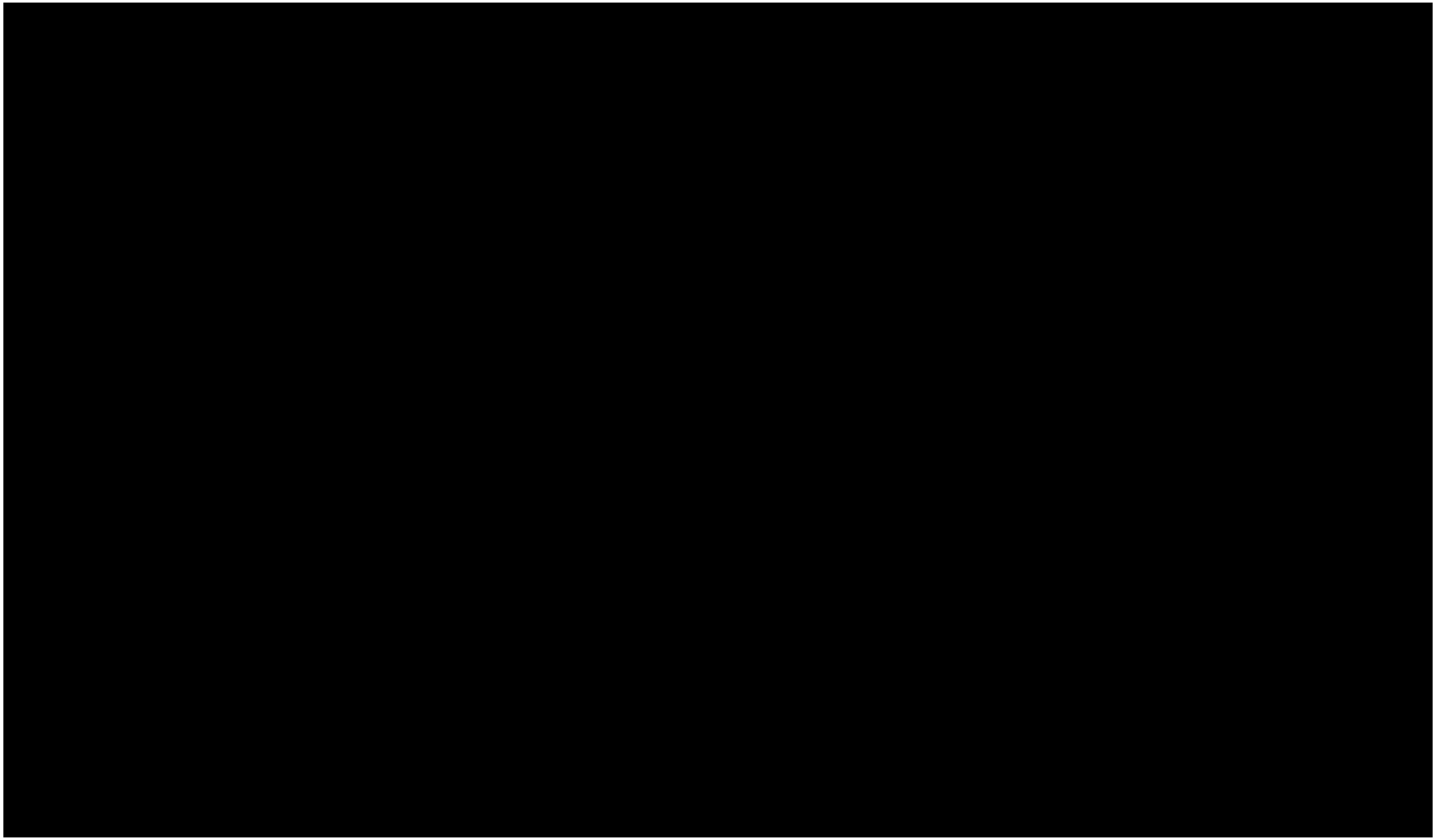
CNN and Cascades







TEST RUN



Future Scope

1. Apply the concept for drones and reprogram for 3D movement as opposed to 2D
2. Find more efficient algorithms and faster response rate
3. Research paper

THANK YOU