RIS PROJECT 2023

Collision Avoidance

Emmanuel Mutuku and Irfan Karmali

Collision Avoidance

- A collision avoidance systems is a safety system designed to warn, alert, or assist drivers to avoid imminent collisions and reduce the risk of incidents.

- Implemented in the Duckiebot our goal was for the Duckiebot to automatically stop when it was about to hit an obstacle or when it was about to go out of lane.

Collision Avoidance Broken Down

The tasks/prerequisites we thought were important to successfully implement these were the following.

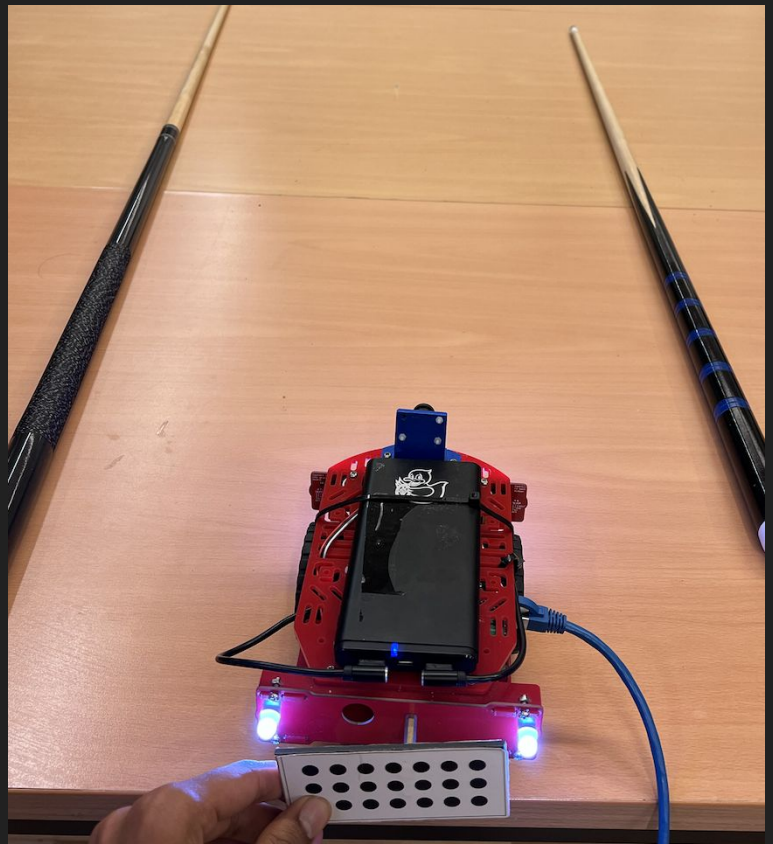
- Object Detection (to detect other Duckiebots or ducks)
- Lane Detection (to ensure it stays within its lane)
- Lane following (to allow it to autonomously move)

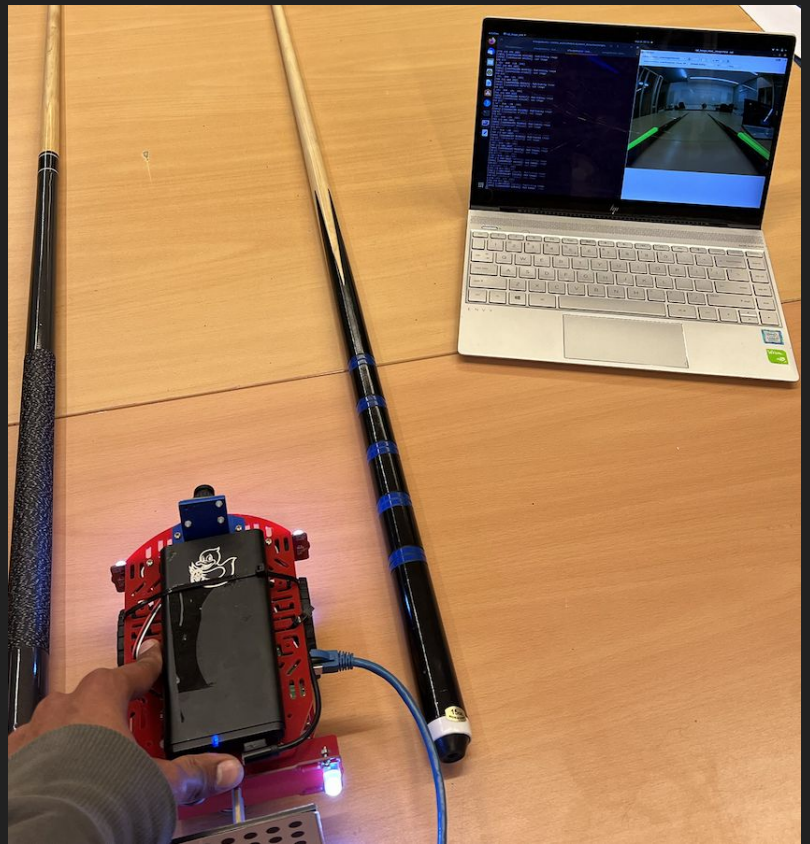
LANE DETECTION

Lane Detection

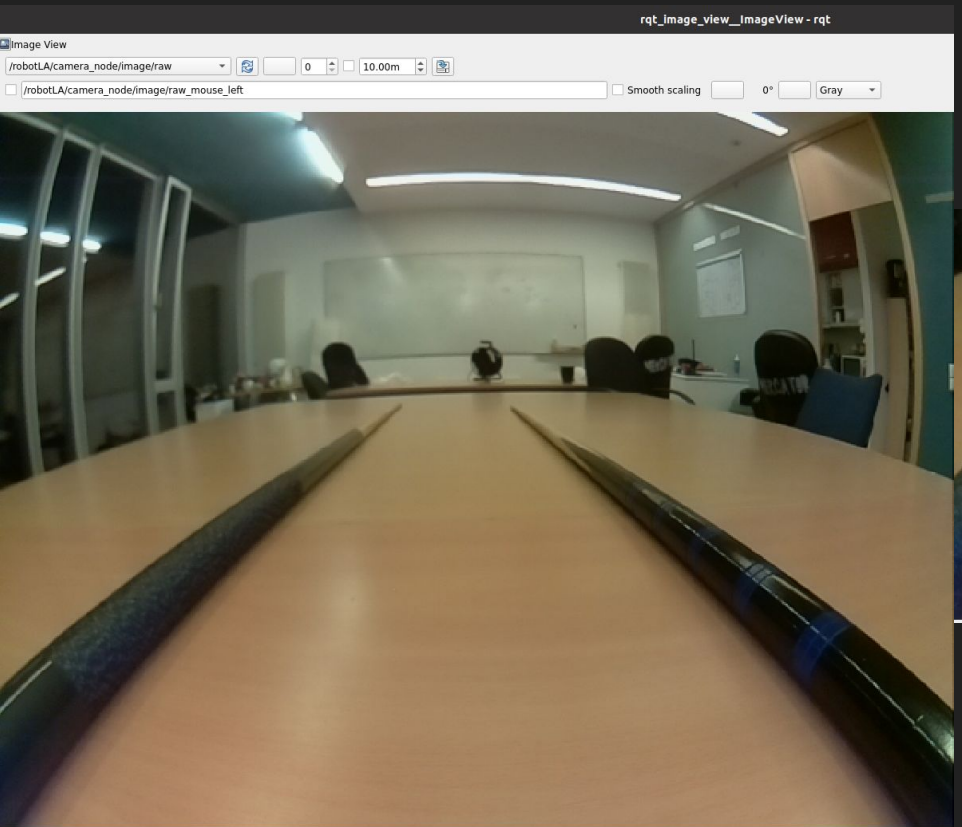
- Lane Detection is a Computer Vision task that involves identifying the boundaries of driving lanes in a video image or road scene.
- For our project Lane Detection was implemented on our Duckiebot called RobotLA so it could detect lanes in the duckietown.

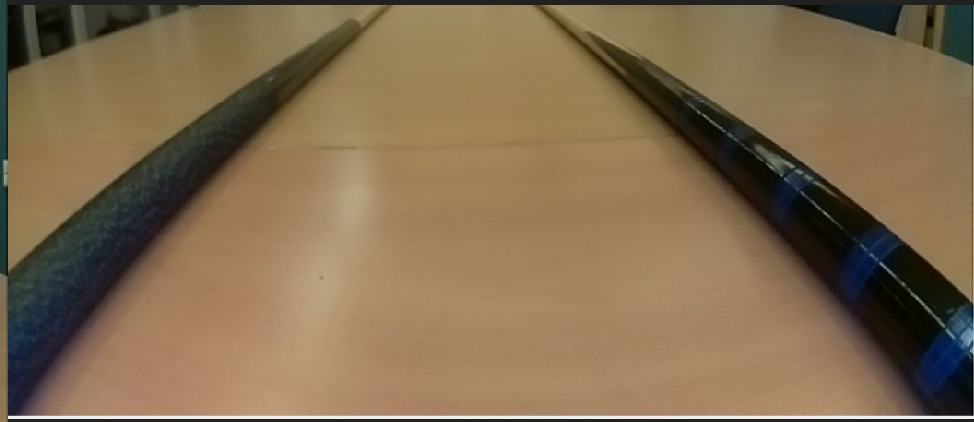
Behind the Scenes for Lane Detection



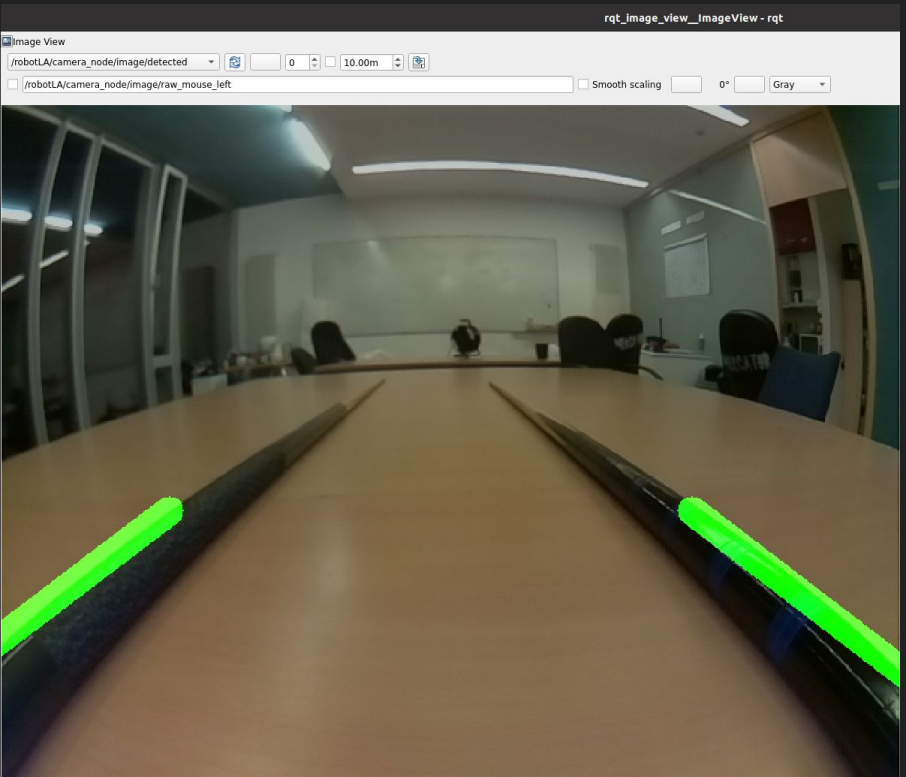


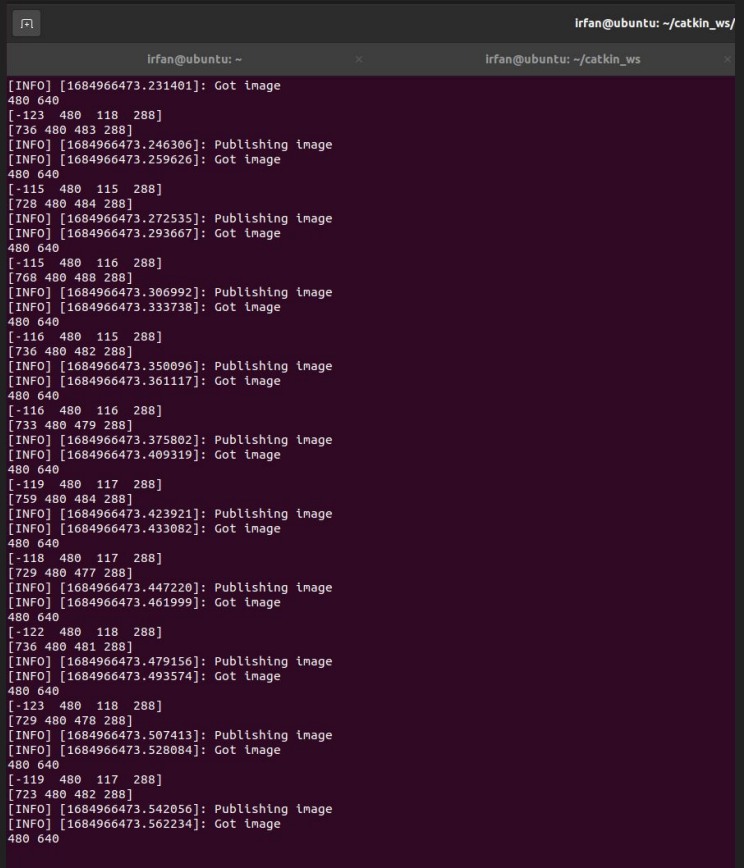
Input as seen from the Duckiebot





Output as seen from the Terminal and Duckiebot





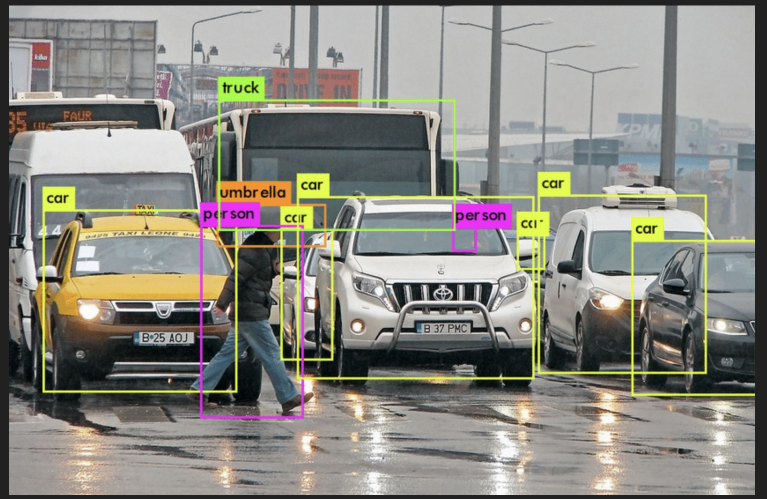
Explaining How Lane Detection was Implemented

- Image was converted to grayscale
- Gaussian blur was applied to reduce noise
- Canny edge detection algorithm was used to detect images
- Finally hough transform was used to detect lines within the defined ROI.
- The detected lines were then averaged to obtain a single line

OBJECT DETECTION

Object Detection

- Object Detection is a Computer Vision technique that allows us to identify and locate objects in an image or video.
- The goal of this Object Detection was to identify ducks, stop signs, and other duckie in the duckietown.

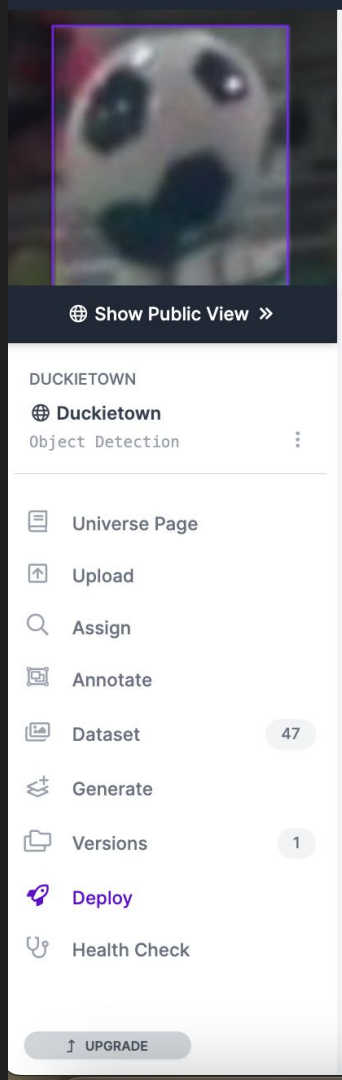


Training the Duckiebot

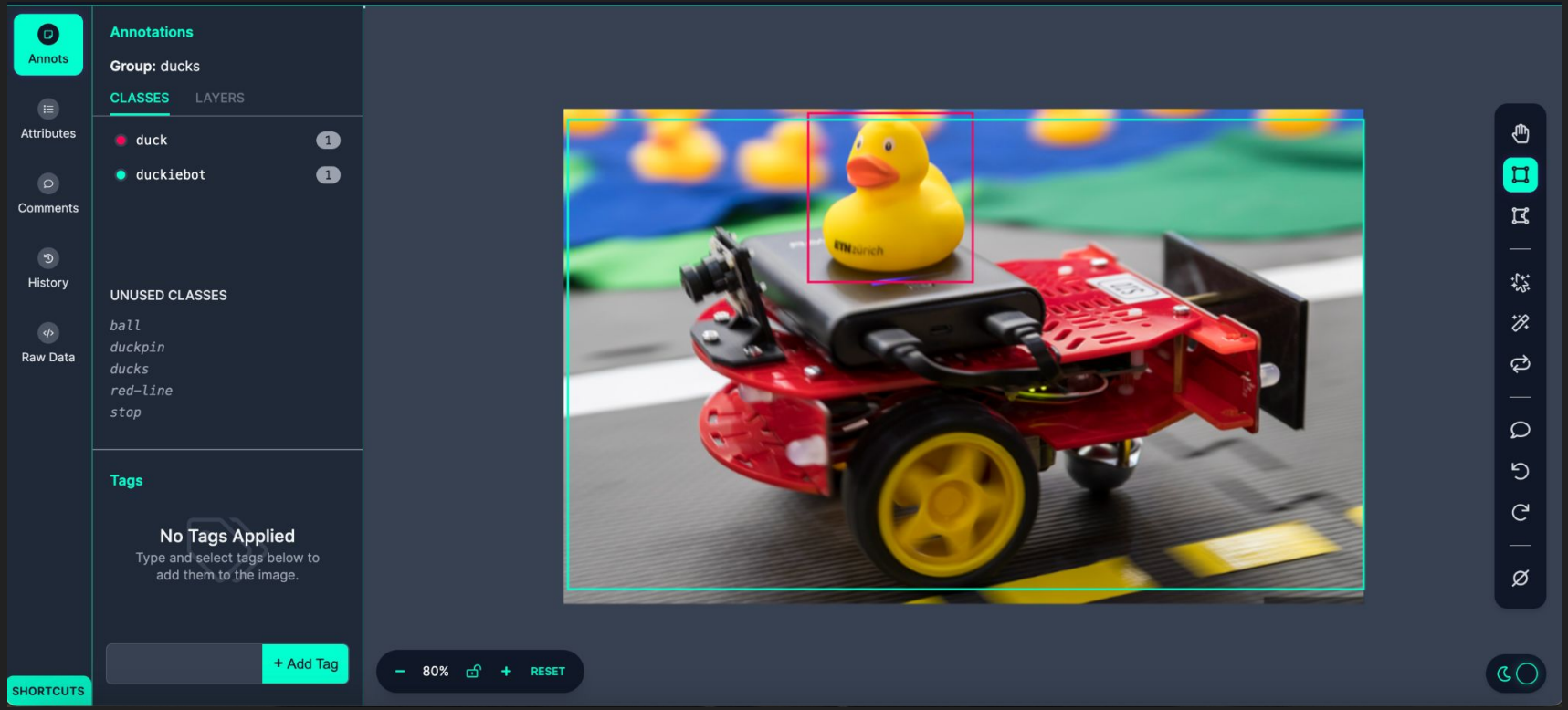
- For this project the model was trained using Roboflow.
- Pictures were a collection of both, pictures taken from the Duckiebot and external images were used.
- Images were annotated manually to train the dataset and finally a validation set was used to test the model.
- The model was further tested in our own Duckietown. The results of which you will see soon enough.

Working with Roboflow

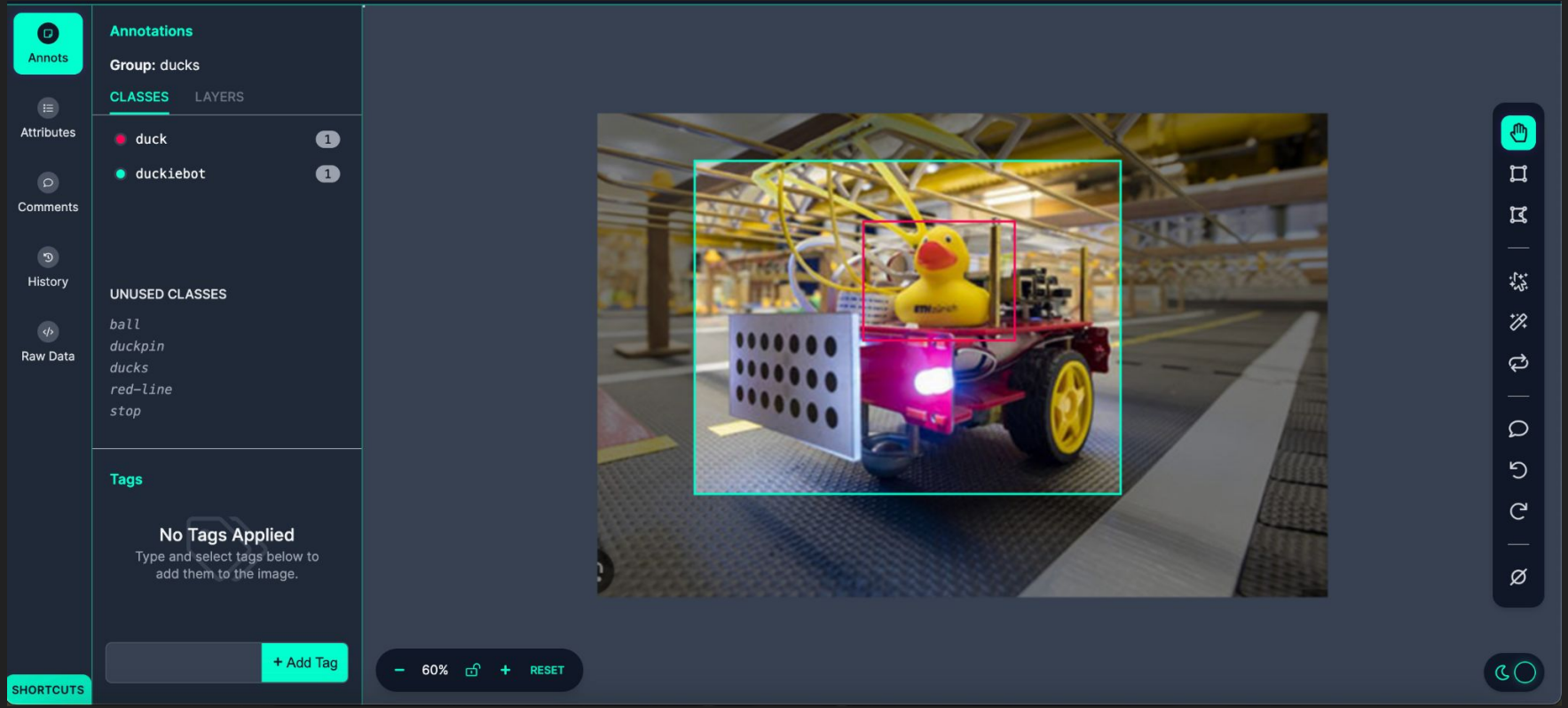
- Step 1: Create a project
- Step 2: Upload images onto Roboflow
- Step 3: Assigning Images
- Step 4 : Annotating Images
- Step 5: Creating classes
- Step 6: Now a Dataset is created
- Step 7: Generating the dataset



Roboflow and annotation of Images

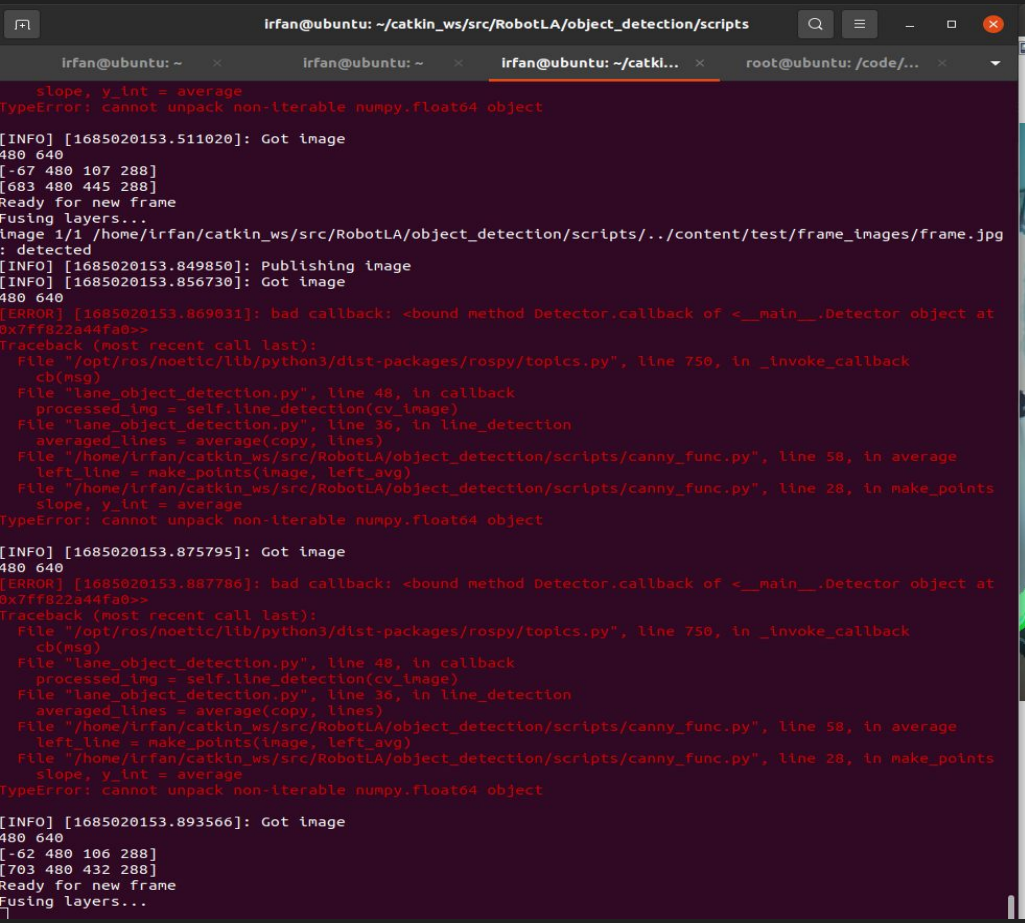


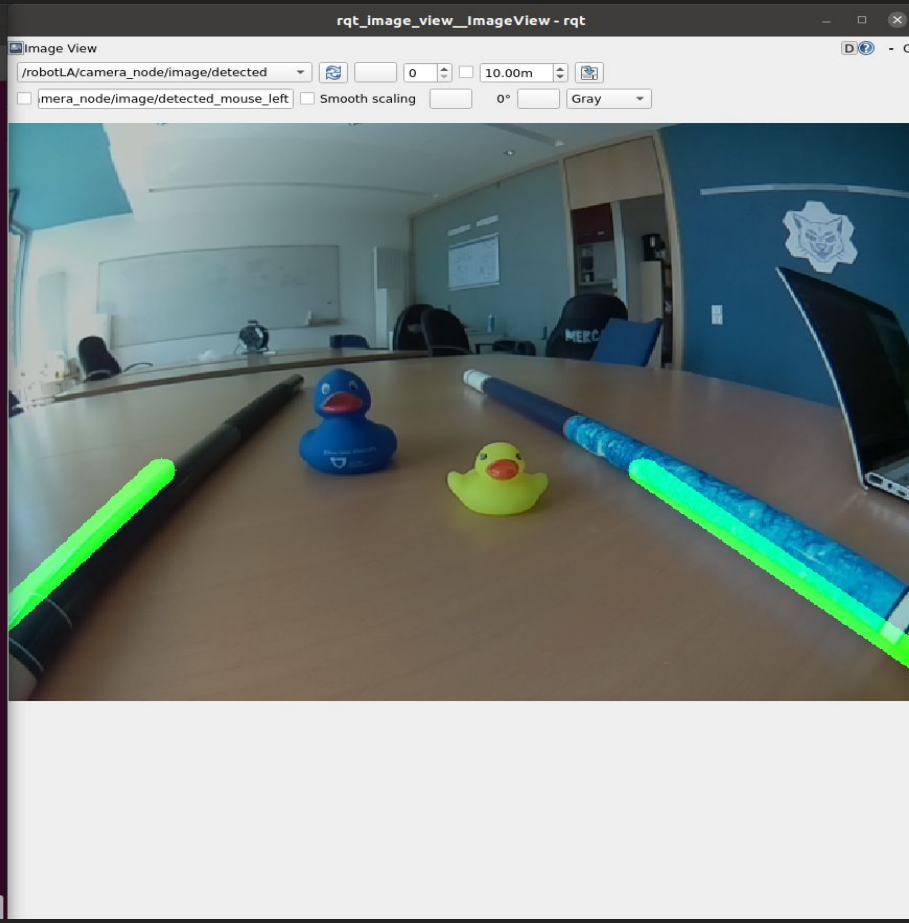
Roboflow and annotation of Images



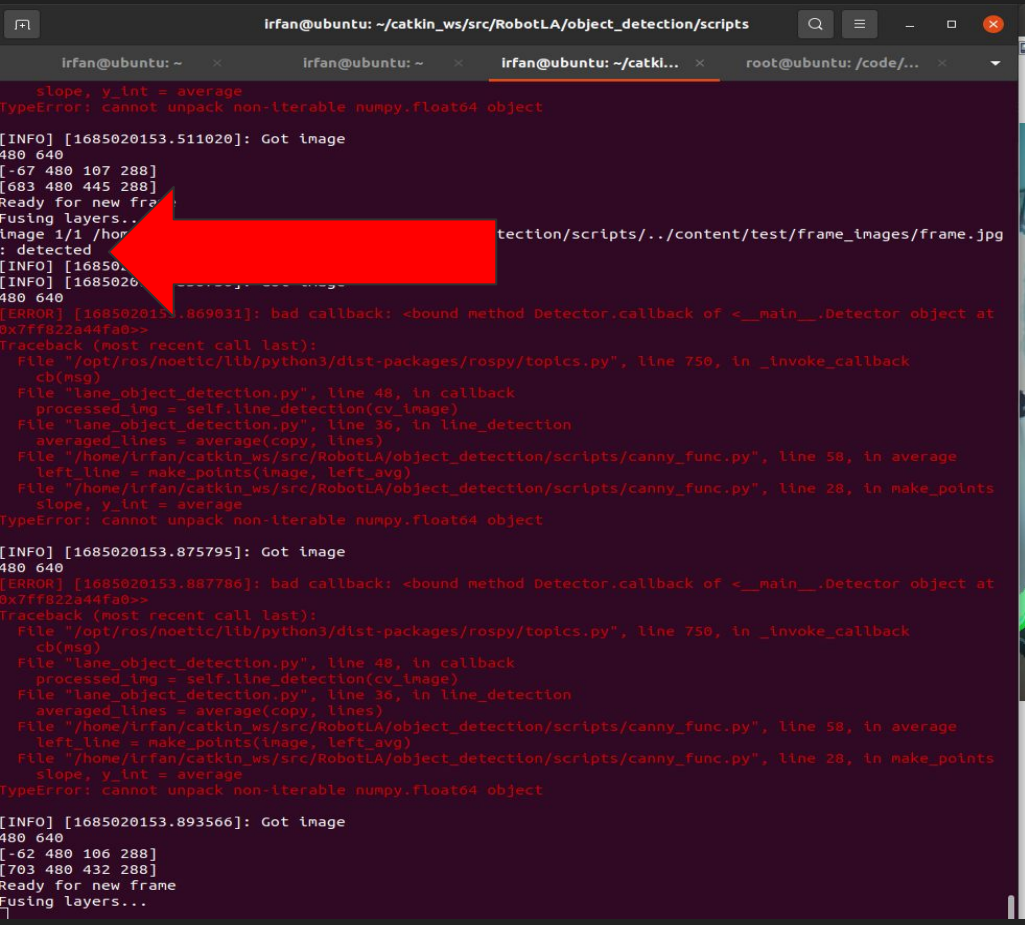
Behind the Scenes for Object Detection

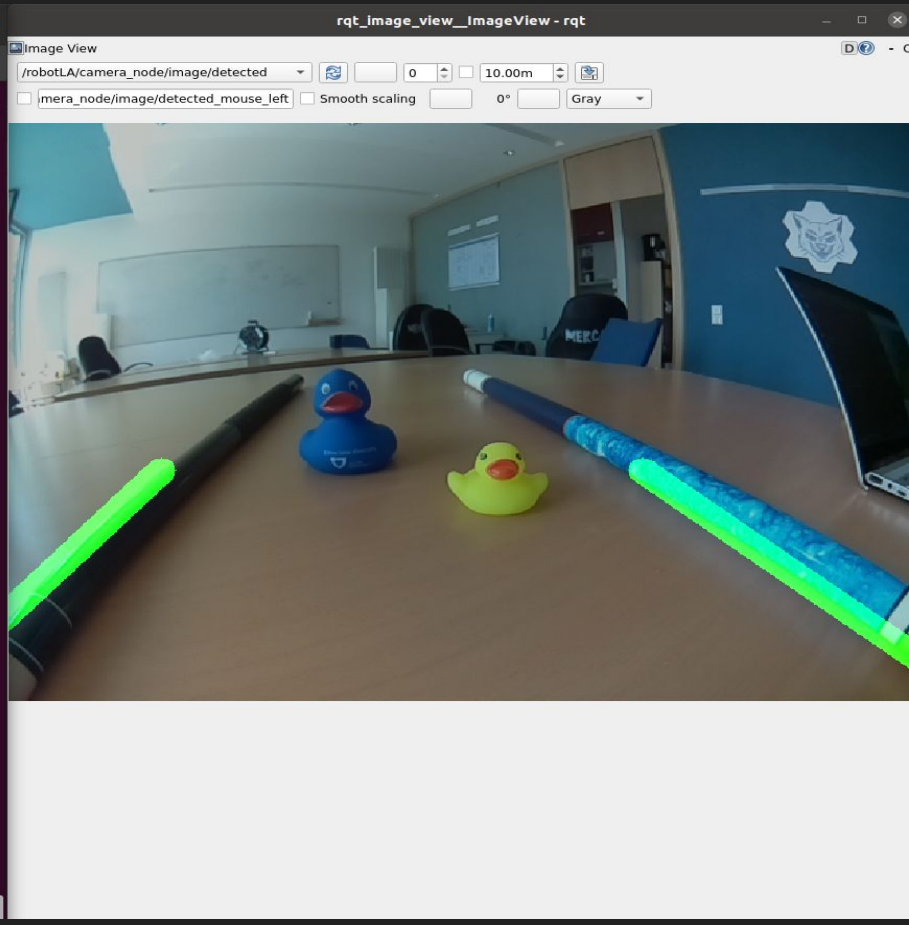
Output and Input for Object Detection (1 Duck)



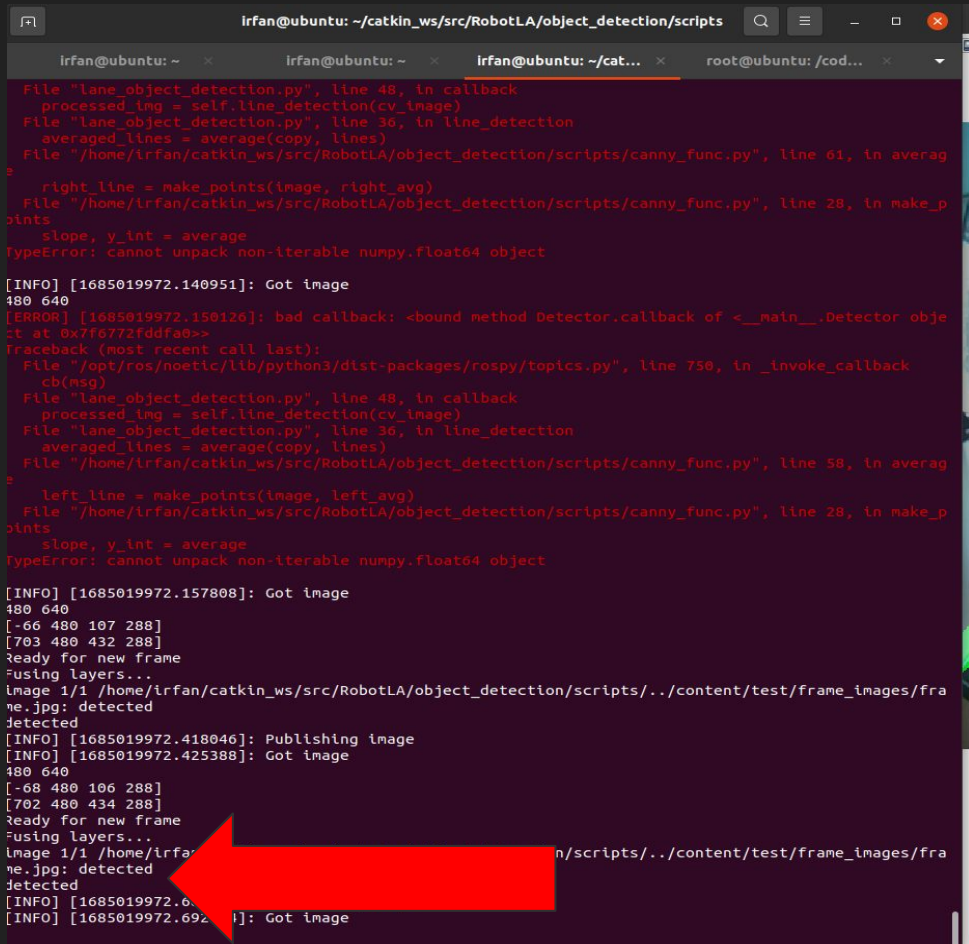


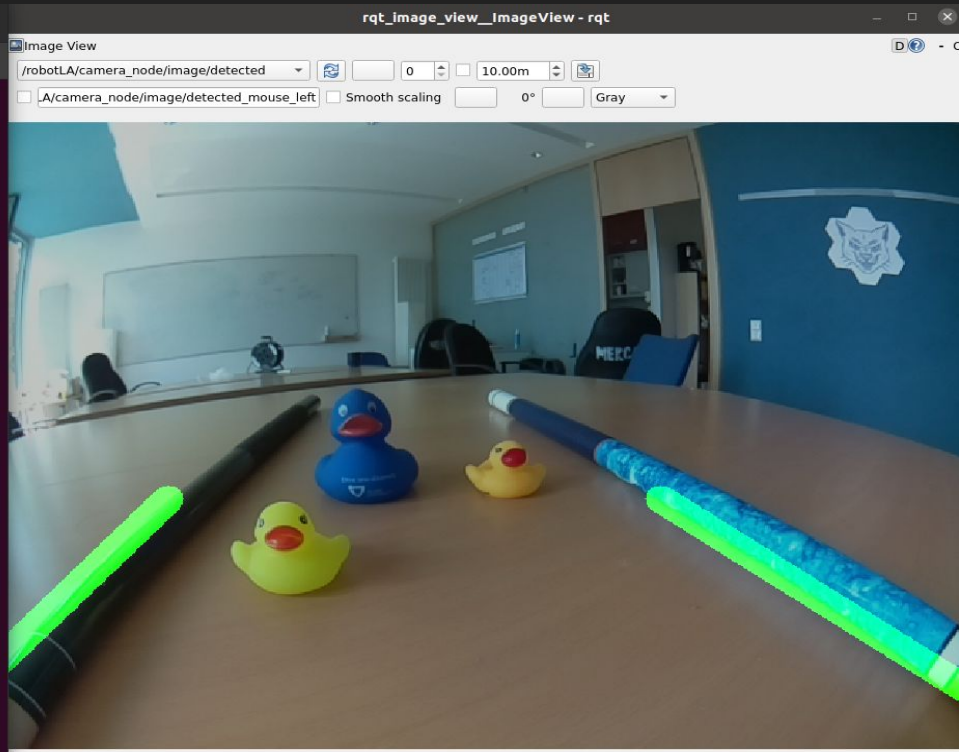
Output and Input for Object Detection (1 Duck)





Output and Input for Object Detection (2 Ducks)





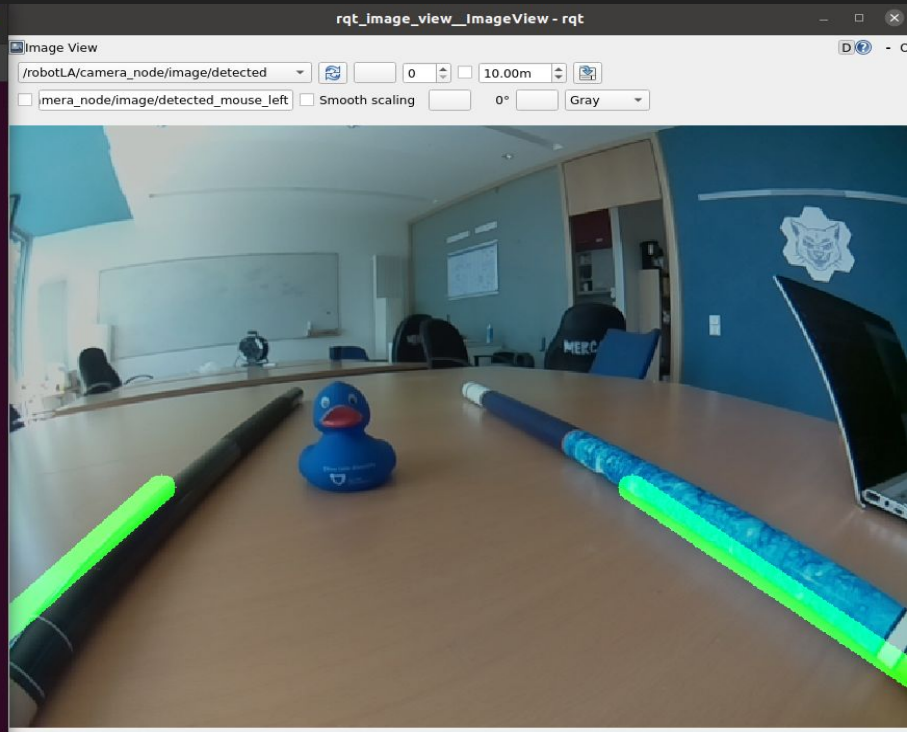
Output and Input for Object Detection (Only Blue Duck)

```
irfan@ubuntu: ~/catkin_ws/src/RobotLA/object_detection/scripts
irfan@ubuntu: ~ | irfan@ubuntu: ~ | irfan@ubuntu: ~/catki... | root@ubuntu: /code/...
    left_line = make_points(image, left_avg)
  File "/home/irfan/catkin_ws/src/RobotLA/object_detection/scripts/canny_func.py", line 28, in make_points
    slope, y_int = average
TypeError: cannot unpack non-iterable numpy.float64 object

[INFO] [1685020137.756845]: Got image
480 640
[-84 480 107 288]
[693 480 439 288]
Ready for new frame
Fusing layers...
image 1/1 /home/irfan/catkin_ws/src/RobotLA/object_detection/scripts/../content/test/frame_images/frame.jpg
: [INFO] [1685020138.005305]: Publishing image
[INFO] [1685020138.011787]: Got image
480 640
[ERROR] [1685020138.025888]: bad callback: <bound method Detector.callback of <__main__.Detector object at
0x7ff822a44fa0>>
Traceback (most recent call last):
  File "/opt/ros/noetic/lib/python3/dist-packages/rospy/topics.py", line 750, in _invoke_callback
    cb(msg)
  File "lane_object_detection.py", line 48, in callback
    processed_img = self.line_detection(cv_image)
  File "lane_object_detection.py", line 36, in line_detection
    averaged_lines = average(copy, lines)
  File "/home/irfan/catkin_ws/src/RobotLA/object_detection/scripts/canny_func.py", line 61, in average
    right_line = make_points(image, right_avg)
  File "/home/irfan/catkin_ws/src/RobotLA/object_detection/scripts/canny_func.py", line 28, in make_points
    slope, y_int = average
TypeError: cannot unpack non-iterable numpy.float64 object

[INFO] [1685020138.033447]: Got image
480 640
[ERROR] [1685020138.047915]: bad callback: <bound method Detector.callback of <__main__.Detector object at
0x7ff822a44fa0>>
Traceback (most recent call last):
  File "/opt/ros/noetic/lib/python3/dist-packages/rospy/topics.py", line 750, in _invoke_callback
    cb(msg)
  File "lane_object_detection.py", line 48, in callback
    processed_img = self.line_detection(cv_image)
  File "lane_object_detection.py", line 36, in line_detection
    averaged_lines = average(copy, lines)
  File "/home/irfan/catkin_ws/src/RobotLA/object_detection/scripts/canny_func.py", line 61, in average
    right_line = make_points(image, right_avg)
  File "/home/irfan/catkin_ws/src/RobotLA/object_detection/scripts/canny_func.py", line 28, in make_points
    slope, y_int = average
TypeError: cannot unpack non-iterable numpy.float64 object

[INFO] [1685020138.055975]: Got image
480 640
[-84 480 107 288]
[702 480 433 288]
Ready for new frame
```

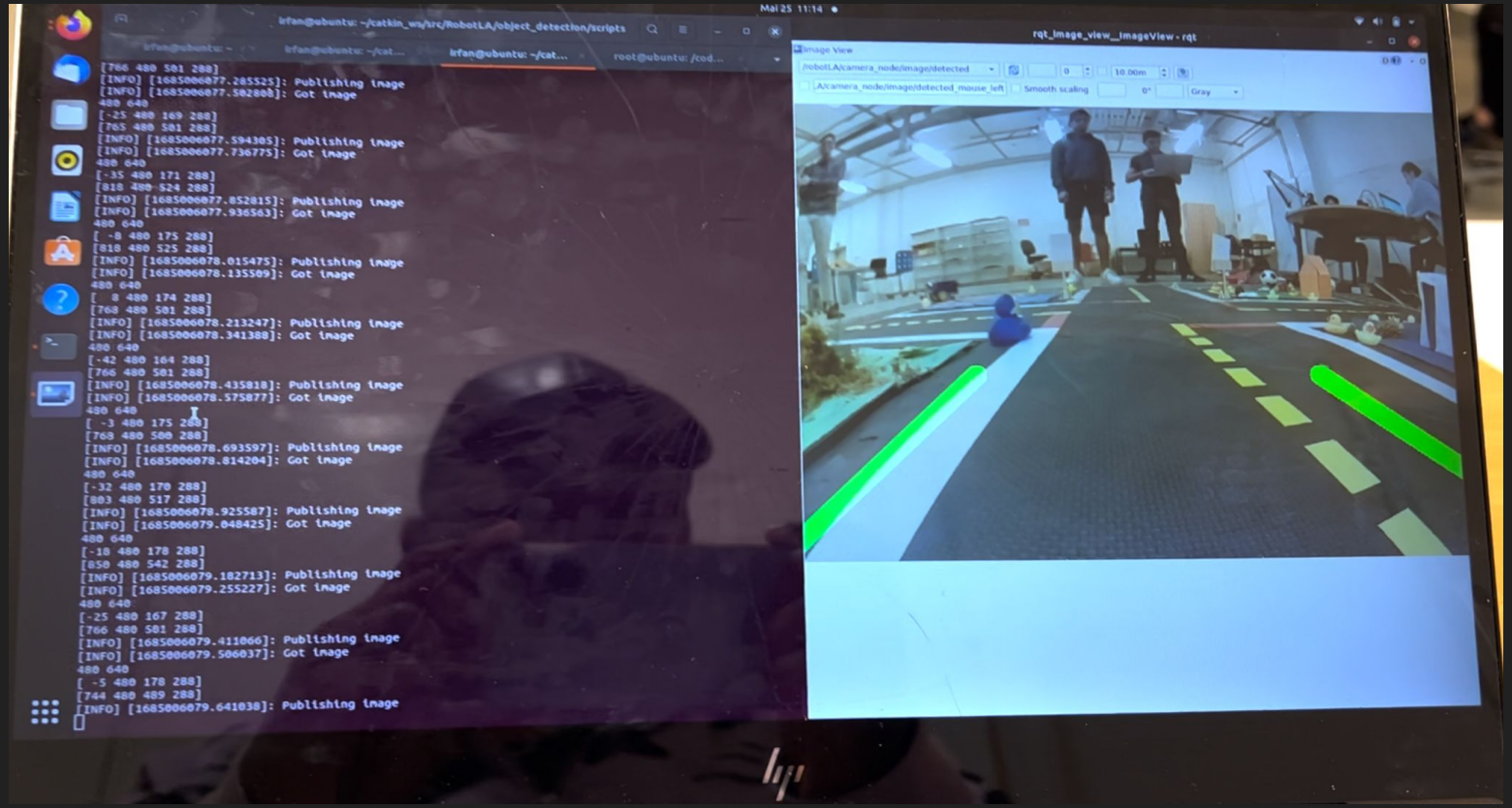


Explaining Object Detection

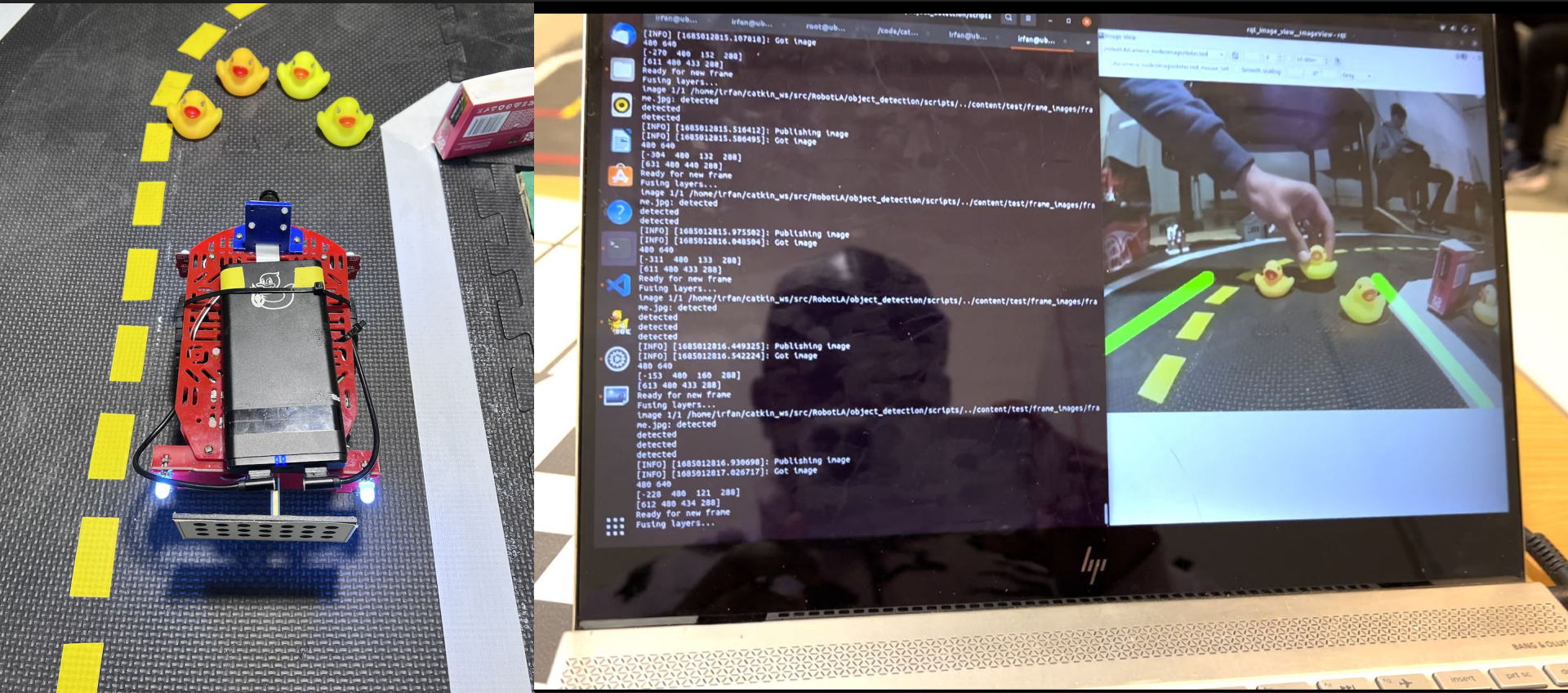
- Necessary modules and functions for object detection using a pre-trained model thanks to roboflow.
- We used a YOLOv5 model for detecting objects in images.
- Non-maximum suppression was performed and a classifier was applied to detect objects.
- The final image with objects detection results was obtained using the ‘detect function.

Testing in Duckietown

Running Lane Detection in Duckietown



Running Object Detection in Duckietown



Combining the both

CHALLENGES AND FURTHER IMPROVEMENTS

Challenges and Further Improvements

- We could’ve added distance detection and fully implemented it
- We could’ve added boxes around the detected objects
- Furthermore, we could’ve implemented lane following and further programmed it to stop when it detects an object that is near.