TUM School of Computation, Information and Technology
Technical University of Munich

PROJECT REPORT ON

"Self-Driving & Obstacle Avoiding Car"

Submitted By:
Ilia Panayotov, Jing Yao Seow, Nour Hadj Fredj,
Esin Mehmedova, Esra Mehmedova, Dimitar Gavov

Under The Guidance Of:
Prof. Dr. Amr Alanwar & Instructor Ahmad Hafez

Submission Deadline: 20.08.2024

Table of Contents

1. Abstract

This paper will document our project of creating a self-driving car capable of not only following a line but also detecting obstacles and reacting to them by stopping, then continuing when the way is clear. We will discuss possible approaches before diving into detail about our solution and methodology where we will present our code before we finish by exploring future possibilities and improvements.

2. Introduction

Our project revolves around making a car able to follow a line and detect obstacles. For this, we decided to use the JetRacer ROS AI Kit, which is a high-performance AI intelligent robot for many reasons, like the 160-degree field of view camera manufactured by Sony, ensuring high-quality image capture. For the Mainboard, it has Jetson Nano which offers powerful AI capabilities, surpassing Raspberry Pi due to integrated NVIDIA graphics. Finally, it is equipped with two brushed motors with reduced gears for more torque and a lower response rate. The first step before we began working on the car was to build it, which combined mechanical and electronic components, such as the chassis, wheels, motors, sensors, and the Jetson Nano board. The second step is setting up the software which involves Installing and configuring ROS on both the Jetson Nano and a virtual machine to enable seamless communication and control. After finishing these two steps we are finally able to start focusing on the main part of our project.

3. Problem Statement

In the field of autonomous driving, the vehicle has to be able to navigate an environment to reach its desired destination. This task involves many variables such as road-following and obstacle avoidance among many others. For this project, we aim to address the challenges of road-following and the ability to detect and respond to pedestrians.

3.1. Road following

The challenge of road-following is that the model car must autonomously and accurately follow a predetermined path, following specific road markings while staying within the bounds of the road. This requires the vehicle to be capable of ignoring ‘noise’ such as stray objects in the field of view and irrelevant road markings.

3.2. Pedestrian Detection and Response

The vehicle must be able to detect pedestrians and take appropriate action to avoid collisions. This involves first recognizing pedestrian presence and then stopping the vehicle in a timely manner. This can be extended to not only pedestrians but other objects as well such as traffic lights.

4. Possible Approaches

4.1. Convolutional Neural Network

For credibility's sake, this approach was previously done by a member of this group. This approach involves collecting data samples using the camera to capture images as the car traverses the path. During the data collection, a user will manually control the car to navigate the path in an ideal way. Each image is linked to the orientation of the wheel and the throttle speed at the moment the image was taken. False data can be sanitized manually but data is not annotated. We then execute unsupervised learning to train the model using a preconfigured linear regression algorithm.

Since the model is trained using a behavioral cloning technique, the output of the model is directly based on the actions taken during data collection. In short, it means staying within the road boundaries for the duration of the data collection would result in the model learning such. Therefore, for the purpose of obstacle avoidance, we could simply stop the car while a 'pedestrian' is in view of the camera as this would register 0 throttle in the dataset every time.

The reasoning behind not going with this approach is that the model is fragile and slight variance in the appearance of the environment will break the model. As we aim to prioritize reliability and performance over capabilities, utilizing line-following and color detection is the most robust solution given the level of hardware available.

5. Chosen Solution Approach

To address the challenges of road navigation and pedestrian detection in our autonomous model car, we implemented a two-pronged approach focusing on line following for road navigation and color-based detection for pedestrian identification.

5.1. Line Detection and Following

The primary objective of this study is to develop a robust system capable of detecting and following a red line on the road. This system is crucial for the autonomous navigation of our vehicle, enabling it to move independently by adhering to predefined paths in the area. We utilized the predefined ROS package in which we used the line-following algorithm that provides the capability to track and follow a specified color. This algorithm was chosen for its robustness and ease of integration with our model car platform. It is integrated with the car’s control system, allowing the model car to make real-time adjustments to its steering and speed based on the detected road markings.

One of the most significant challenges in line detection is the variability of lighting conditions. Both natural and artificial lighting can fluctuate, altering the appearance of the red line. These variations can result in shadows, reflections, or changes in brightness, which can lead to potential misdetections. Another challenge concerns the difficulty in distinguishing between the red lines on the road and other red objects, such as signs, markers, or vehicles. Failure to accurately differentiate between these elements can result in erroneous navigation. To address these challenges, we implemented a combination of color filtering and edge detection algorithms. Color filtering enables the system to focus exclusively on red objects, while edge detection enhances the system's ability to identify the straight edges of the road lines. By combining these methods, the system can effectively differentiate between the intended path and other red objects that are not part of the path.

To ensure the car follows the road correctly, the system must effectively handle curves and intersections. We addressed this we utilized a predictive tracking algorithm that estimates the path's curvature based on the detected segments of the red line. This algorithm uses mathematical models to predict the trajectory of the line. We included a Proportional-Integral-Derivative (PID) control system to preserve smooth and continuous movement without instabilities, which manually adjusts the vehicle's steering angle. With the PID system, we have found a midpoint that balances between corrective actions and smoothness, ensuring the vehicle moves straight and steadily. In addition, we have real-time feedback tools, which help us monitor the vehicle's performance and modify the control parameters dynamically. This enables the vehicle to adapt to varying conditions while maintaining optimal driving performance.

5.1.1. Color Calibration and Detection

In our simulated environment, we pre-selected a distinct color to represent the road markings. This color was chosen to ensure high contrast with the surrounding environment, ensuring higher accuracy by the car's vision system.

We calibrated the line-following system to recognize the chosen road color. This involved fine-tuning the color detection parameters to ensure the model car could reliably identify the road markings under varying lighting conditions within the simulation.

The detection algorithm was tested extensively to confirm that it could accurately follow the predefined road color, ensuring consistent navigation through the simulated environment.

5.1.2. Sensor Integration and Calibration

In this system, we integrate sensors, such as cameras, to detect specific colors for navigation and stopping, which are crucial for the proper and reliable operation of the vehicle's autonomous functions. Sensor calibration is important part of keeping the accuracy of these sensors, ensuring consistent performance across different environments and over time.

To address this, we selected a high-resolution camera with a 160-degree viewing angle equipped with a fish-eye lens. This broad field allows complete coverage of the vehicle's surroundings. While beneficial for wide-angle coverage, the fish-eye lens introduces distortion that must be corrected for accurate image processing. For the camera, we utilized ROS-integrated software for calibration, addressing the distortion introduced by the fish-eye lens. A chessboard-like calibration pattern was used, enabling the software to correct the image and eliminate distortion. This calibration process ensures that objects within the camera's field of view retain their true shape and position.

5.2. Pedestrian Detection And Stopping

To facilitate pedestrian detection, we defined a specific color range for the yellow rubber duck pedestrian in our simulation. This range was carefully selected to account for variations in the yellow hue that might occur due to changes in lighting conditions.

```
self.lower_yellow = np.array([22, 93, 0])
self.upper_yellow = np.array([45, 255, 255])
self.yellow_detected = False
```

Figure shows HSV color range

We extended the ROS library by developing a custom color-based detection algorithm capable of identifying objects within the defined yellow color range and triggering a stop when such an object is detected. Once the object is out of range, the vehicle resumes navigation. This algorithm processes real-time visual input from the car's camera module, continuously scanning for the presence of the yellow pedestrian.

```
if self.set_yellow_color:
    self.lower_yellow = np.array([H_min, S_min, V_min])
    self.upper_yellow = np.array([H_max, S_max, V_max])
```

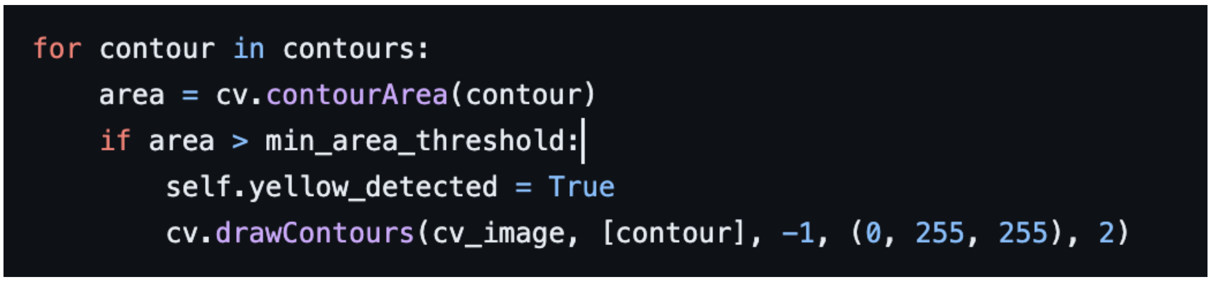
Figure shows the upper/lower bounds, which define the tolerance for "yellow"

This detection system was integrated with the car's control logic to enable an immediate stop when a pedestrian is detected. This involved modifying the control code to include a stopping command triggered by the detection of yellow within the car's field of view.

```
else:
    self.cmd_pub.publish(Twist())
    cv.putText(cv_image, "Duck detected!", (30, 70), cv.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 255), 1)
```

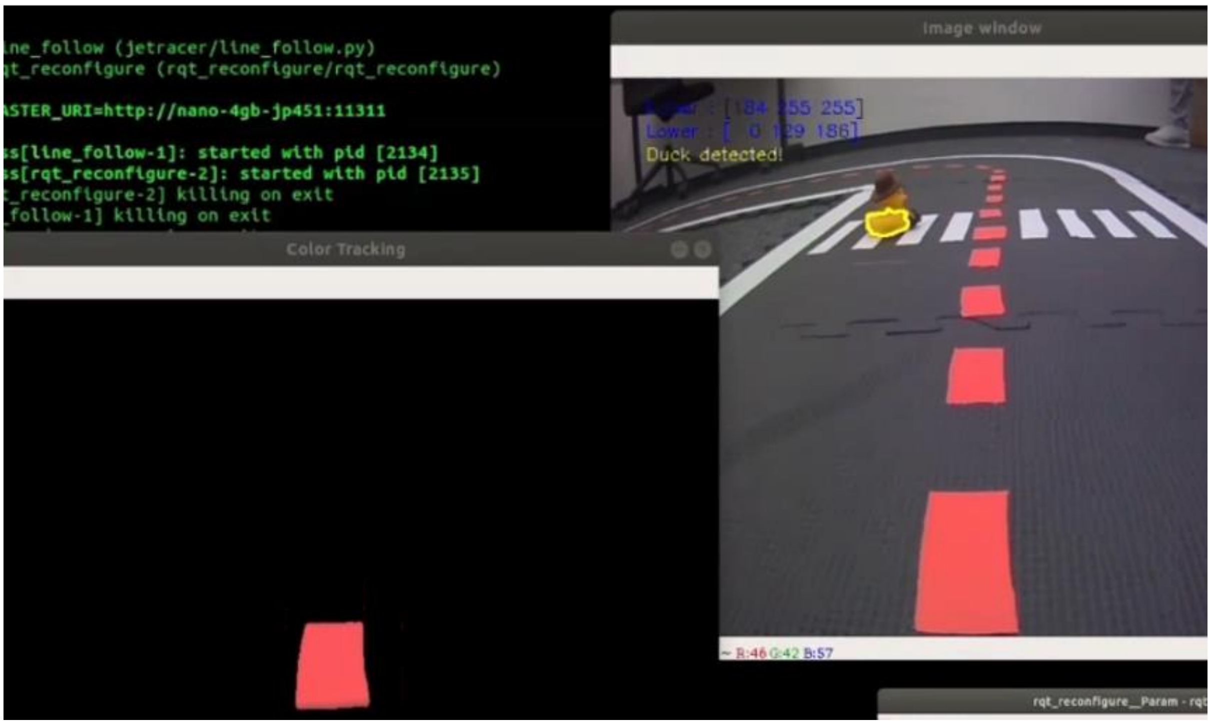
Figure shows the output when an object that falls within the boundaries is detected. Twist() method works to stop the vehicle while the console outputs relevant text of the boundaries.

The computer vision method outlines the object it identifies as a duck, which is visible in the live feed. This feature is useful for troubleshooting, as it allows for easy identification of objects that are mislabeled by the algorithm.



```
for contour in contours:
    area = cv.contourArea(contour)
    if area > min_area_threshold:
        self.yellow_detected = True
        cv.drawContours(cv_image, [contour], -1, (0, 255, 255), 2)
```

Figure shows a code snippet of the 'outlining' logic.



```
ine_follow (jetracer/line_follow.py)
qt_reconfigure (rqt_reconfigure/rqt_reconfigure)

ASTER_URI=http://nano-4gb-jp451:11311

ss[line_follow-1]: started with pid [2134]
ss[rqt_reconfigure-2]: started with pid [2135]
t_reconfigure-2] killing on exit
_follow-1] killing on exit
```

```
Lower : [  0 129 186]
Duck detected!
```

Figure shows the live feed of the algorithm detecting a yellow duck.

6. Conclusion

In this paper, we have discussed in detail how we were able to achieve our goals of creating a self-driving car that is able to follow a line, detect obstacles, and react to them. We have seen how the project started with building the robot and setting up the software before we presented the problem statement. We followed by presenting our solution to the problem and explaining in detail how we managed to configure the self-driving car. Several additional improvements can be made to enhance the car's capabilities, such as enabling it to follow a person and recognize a broader range of colors. These enhancements could further refine the vehicle's autonomous navigation and interaction with its environment, leading to more versatile and adaptive behavior that improves overall performance and safety.