Self driving car - Simulation

Anh Phan Ngoc, Thinh Duy Ngo, Truong Nguyen Nhat
Advisor: Trung Nguyen Quoc

Abstract: In this report, we present methods to solve the problem of self-driving cars running on 1-lane roads and enforcing signs of signals in the real environment. Based on the image from the real-time camera, the system will process and give a suitable set of parameters containing the car's direction control parameters and the car speed adjustment parameters. We divide the problem into two main parts, lane detection and traffic sign recognition, respectively. For the lane detection part, we used the data set consisting of 3 classes of approximately 7000 images each to train the convolution neural network (CNN). For the traffic sign recognition part, we use YOLOv5 to detect the position of the signs, then continue to use CNN to classify the signs. The dataset we built for this part is 22 classes with about 500 images each. According to the results of system testing, the recognition accuracy was 95%.

Keywords: CNN, YOLOv5, traffic sign recognition, lane detection

1. Introduction

With the strong influence of Industry 4.0, the automotive industry is experiencing a paradigm shift from conventional, human-driven vehicles into self-driving, artificial intelligence-powered vehicles. Self-driving cars will provide a safe, efficient solution that saves a lot of time and money for people. Consequently, solving problems related to self-driving cars is being invested in by most car companies around the world. Currently, there are 3 main approaches representing the top 3 automotive companies in the world [1]. The first, Tesla Inc., uses computer vision with deep learning, which combines image processing and machine learning to give each car real-time knowledge of its surroundings. Second, companies like General Motors, Mercedes Benz and Ford create high-resolution virtual 3D environments that are captured beforehand. The vehicle can then use the map and position locator to determine the corresponding position and control the movement. Otherwise, the third company, Volkswagen, does not focus on smart cars, but on creating a smart environment for the car to move on.

In this project, we choose the computer vision approach with classification and object detection algorithms. From a set of images obtained at each moment, the car needs to recognize the road, signs, and interference obstacles that appear in its surroundings. We use YOLOv5 [2] to detect and align traffic signs, use 2 models CNN in image classification [3] to classify car actions and classify traffic signs.

2. Self driving car

In this project the car will get the images from the front camera in the car and process it through a self-driving model to give signals including speed and direction. To implement this problem we divide it into two main modules for processing. In the first module we will use the input image to predict the direction the car will go. In the second module we will use the input image to identify road signs. The output of these two modules will be fed to the controller to adjust the output direction and speed in accordance with the road and traffic signs. This controller is manually built on traffic knowledge.

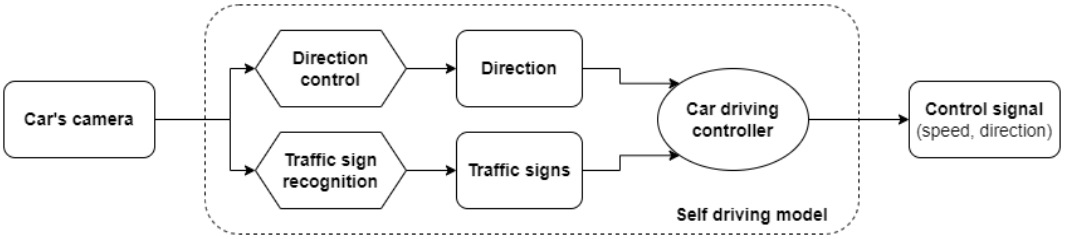

Fig. 2.1. Self-driving car processing flowchart

2.1 Follow lanes: Direction control

In this module, the input image will be fed into a classification model based on the CNN structure to give the direction. The output of this model will have three types: turn left, forward and turn right.

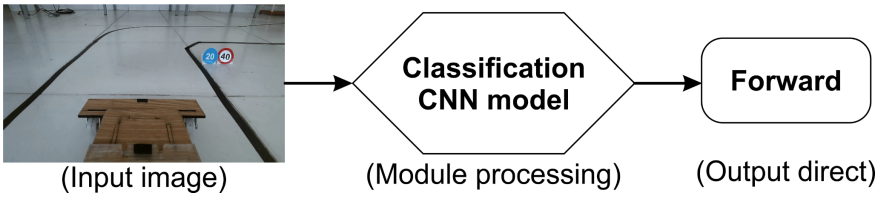

Fig. 2.2. Direction control module flowchart

2.2 Speed control: Traffic sign recognition

The task of this module is to recognize road signs that are read through the camera. The input image will be fed into the traffic sign detection model to retrieve the signs present in the image. These images will be passed through a classifier to determine the names of the signs.The input image will be fed into the traffic sign detection model to retrieve the signs present in the image. These images will be passed through a classifier to determine the names of the signs. This classifier will have 22 common types of speed and signal signs.

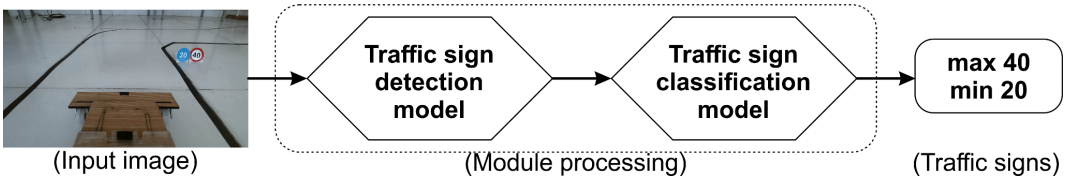

Fig. 2.3. Traffic sign recognition module flowchart

3. Hardware and data preparing

In this project we use a simple car model that can get the image from the camera and send it to the laptop and receive the control signal through bluetooth. We have two modules in this project, direction control data we get from manually driving, traffic sign recognition data we get from some dataset on the internet and from manually driving.

- **Direction control data.** We use the controller to control the car manually and record the log of control signals along with images from the camera. The logs contain 'turn left', 'turn right' and 'forward' signals. We have over 20.000 images categorized by logs. A CNN model will use these images for input and signal from logs for output.

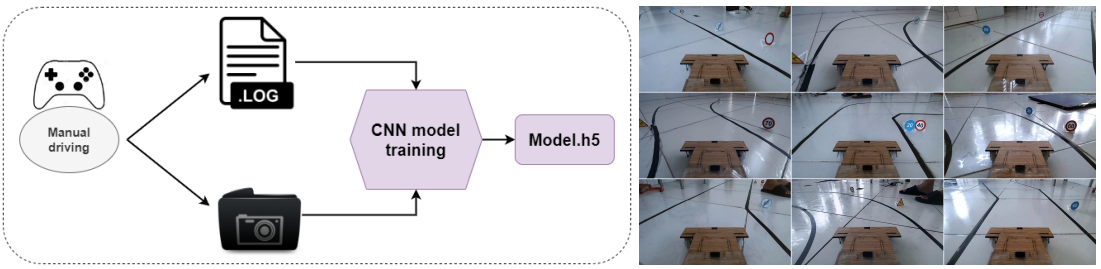

Fig. 3.1. Direction control training flowchart and sample dataset

- **Traffic sign recognition.** We used 1000 images from the car camera on the internet and 1000 images from our driving. The data will be labeled with the signs, here we use 22 types of signs. After that, the data will be divided into two groups, the first group will bring the signs to one group to identify the signs, the other group will cut the signs from the data to put in the categories. The detection model will run in reality to get more images (1000 images) for the sign classification model. Dataset will add a none signs class to avoid detection frames without signs.

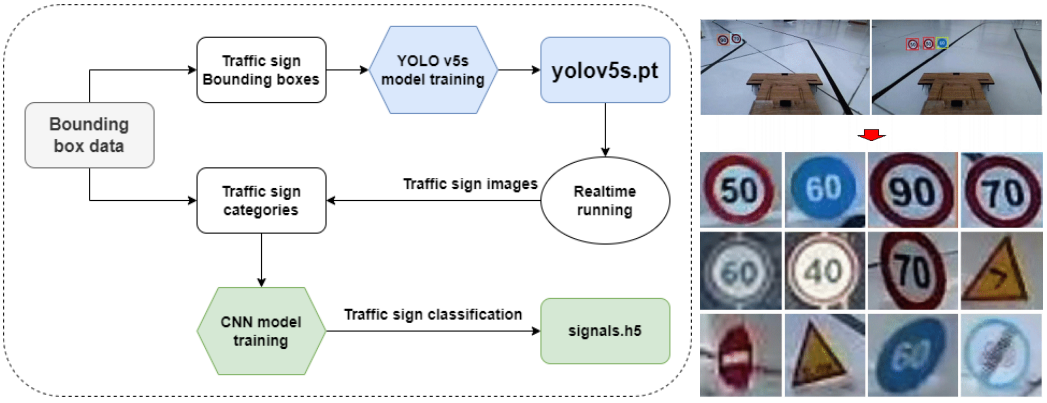

Fig. 3.2. Traffic sign recognition training flowchart and sample dataset

4. Method

This is a problem of controlling a device with feedback (feedback loop), so the processing frequency and processing speed are the deciding factors for the device to work or not. Our vehicle is 18cm wide and the road is 35cm wide, so the clearance on each side of the road will be $d = (35 - 18)/2 = 8.5cm$. With the maximum allowable deviation angle of 60 degrees, the distance for the vehicle to deviate from the road is $l = \frac{d}{sin(\alpha)} = \frac{8.5}{sin(60)} \simeq 9.8cm$. If the vehicle has a minimum speed of 20cm/s, it will take $T = \frac{l}{v} = \frac{9.8}{20} \simeq 0.5second$ to deviate from the road. Therefore, the response speed of the vehicle must be at least 3 times faster, we can calculate the minimum processing frequency of the vehicle: $F_s \geq \frac{1}{T/3} = \frac{1}{0.5/3} \simeq 6(Hz)$. To increase smooth driving, we will choose a processing frequency of 10Hz.

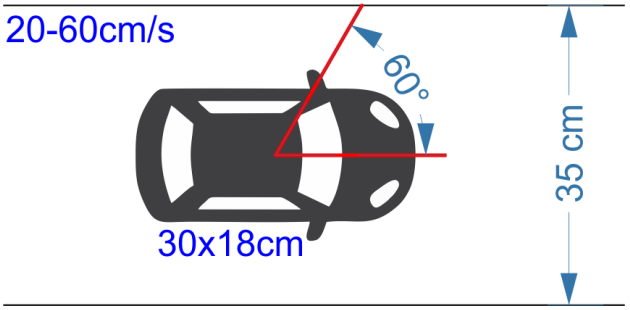

Fig. 4.1. Car and road specifications

4.1 Follow lanes: Direction control

To perform the driving direction (turn left, forward and turn right) with the input images, we use the CNN model to classify the images. In this model, we change the resolution of the image from 1920x1080px to 48x27px to help reduce the computation time of the convolution layer as well as reduce the large number of parameters in the full connection layer. With a 48x27px image, the lane information is still enough for the model to recognize which direction to go.

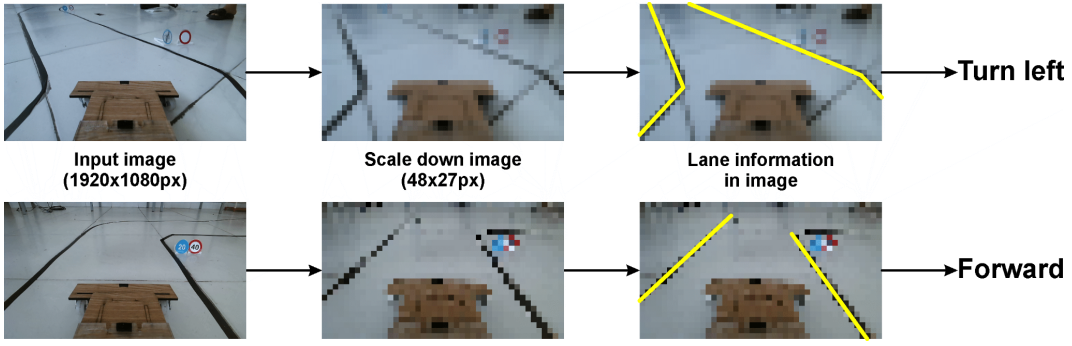

Fig. 4.2. Resolution and information contained in the image

With this basic recognition model, we have 3 convolution layers with a kernel of 3x3, the content in the image is not too much, so with these 3 layers, the features have been fully extracted. For the full connection layer we use 64 nodes to increase the logic level for predicting the 3 node actions in the last full connection layer.

Fig. 4.3. Direction control architecture

4.2 Speed control: Traffic sign recognition

This module will recognize the sign to give the appropriate speed. There is a disadvantage when using object detection to identify signs, there will be cases where the sign occupies a small percentage of the frame, so when scaled, the information it contains is not much, leading to inaccurate identification. We therefore have two stages to carry out this process. The first process will identify the area with the image and cut it out of the original image for the highest quality sign image. The second stage will use those signs with good resolution to classify - identify the information of the sign.

- **Traffic sign detection**

To get short computation time, YOLO v5s model is one of the most optimal choices when combined with CUDA cores calculation takes only 23-30ms.

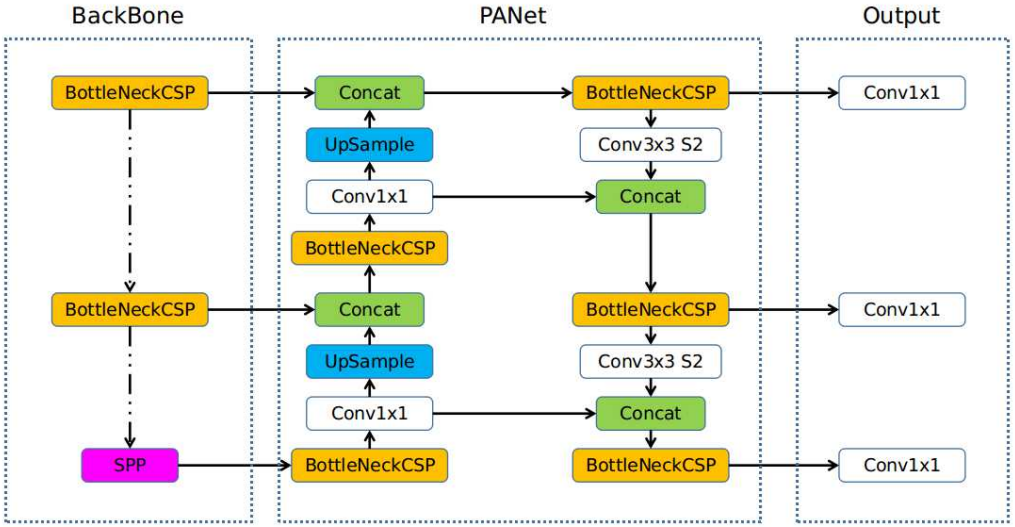

Fig. 4.4. YOLO v5 for traffic sign detection architecture

For this identity model we configure the input image to be 640x384px. In this model we have only 1 output layer representing the sign. In this class we require a confidence threshold of 60%. In more detail, in this phase we take the 1920x1080px image from the camera and scale it down to 640x384px to match our model detection. The image after scale down will be passed through the

model to recognize the traffic signs and return their location. From those locations, high-resolution images will be cropped for later use. Here we have scaled down from 1920px to 640px so the information of the sign will be increased 9 times ($1920/640 = 3; \quad 3x3 = 9$.

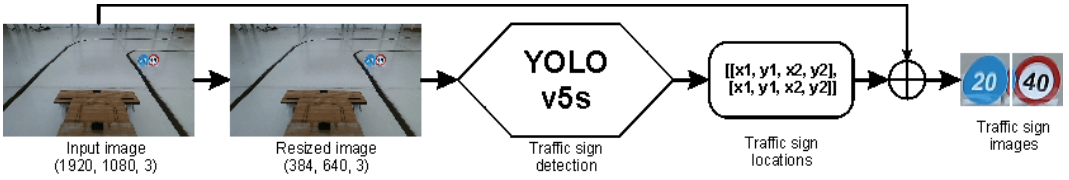

Fig. 4.5. Traffic sign detection working

- **Traffic sign classification**

At this stage, the recognition model will use the output images of the previous stage. The structure of this model is similar to the direction control model. Here the kernels of the convolution layers are 5x5 in size to increase the influence of the features in the image. The input of this model is 50x50px images and the output is 23 layers. Out of 23 output layers, there is 1 layer for identification without signs, enhancing the ability to filter noise for the detection model in the previous stage.

Fig. 4.6. Traffic sign classification architecture

After predicting the traffic signs, we will process to give maximum speed, minimum speed and stop and start signals. This data, after being calculated, will be sent to the car controller via bluetooth.

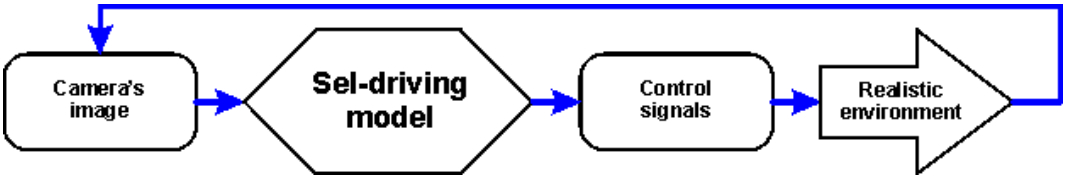

Fig. 4.7. Overview of the control process of self-driving cars.

5. Result and discussion

Taking this project out of the way we were able to make a car with a camera that could drive itself. According to this report, we have built a model that helps the car to follow on roads with a main lane (two lines on both sides) and handle some important signs with high frequency refresh up to 10Hz. The project has been successful when the car can run on its own with very low line touching ability, the car can recognize the speed signs to be able to run at the required speed on the road. We have videos testing here.

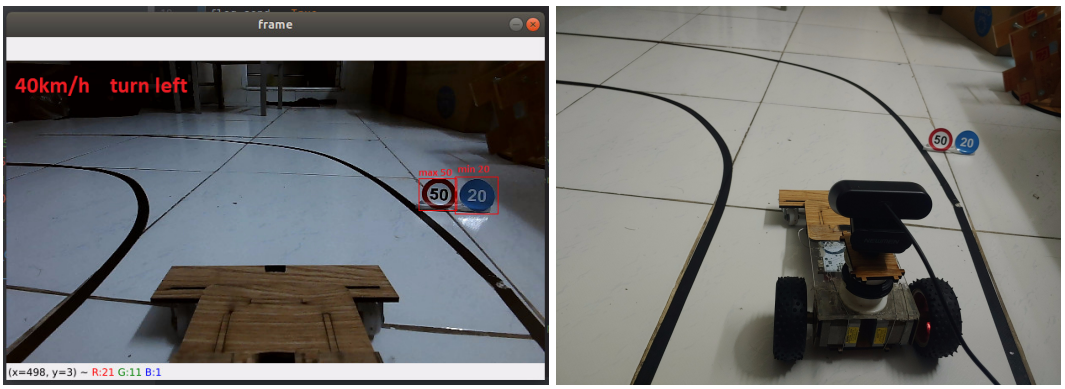

Fig. 5.1. The output of model from camera (left) and real image (right)

This is a form of feedback loop, so the requirements for processing speed as well as signal response are a hard problem. We have a car that can be controlled via Bluetooth and the signal latency is almost zero. We have a Full HD camera with 120 degree wide angle, a computer with 8 CPU cores 2.8GHz and 640 CUDA cores 1.3 GHz. Using CPU cores in combination with CUDA cores in combination with a computationally optimized model helps to increase processing speed by nearly 10 times compared to running CPU cores alone for network traffic sign detection. This helps to meet the processing frequency of the model to help the car run smoothly.

Fig. 5.2. Time sharing of our self-driving model

Besides the success when the car can run correctly with our map, there are some problems that cannot be fixed. We do not have a good camera, so the input image quality is greatly affected by the brightness of the environment when there is no good white balance, which directly affects the predictability of the model. The response speed of the camera is not high (150-200ms), causing the car's feedback signal to be slightly delayed with live. This makes the feedback loop processing ability not achieve high accuracy when the car is running at high speed.

6. Conclusion

In fact, we have many methods to put the self-driving car problem into practice such as:

- Learning the driving ability of drivers through external sensor values.
- Putting vehicle parameters into a virtual environment for the car to learn to drive and recreate the environment from the sensors so that the car can operate.

With the hardware condition as well as the ability not to allow, we have built a self-driving model based on external sensors (here is a camera) so that the car can operate on its own. This method divides work into individual jobs and then aggregates it to give control signals mechanically. It receives images from the surrounding environment to give the vehicle direction and recognizes the traffic signs to be able to run at the right speed. This is a simple method to be able to operate a vehicle with low profile and application for transportation in internal areas where it is not possible to build magnetic lines for robot AGVs.

References

[1] Bloggs, K. (2019). "Three Approaches to Solving the Autonomous Vehicle Orientation Problem" [Blog post]. Retrieved from https://connected.io/blog/

[2] Daria Snegireva, Anastasiia Perkova, "Traffic Sign Recognition Application Using Yolov5 Architecture", *2021 International Russian Automation Conference (RusAutoCon)*, 2021

[3] Farhana Sultana, A. Sufian, Paramartha Dutta, "Advancements in Image Classification using Convolutional Neural Network", *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, 2019

Appendix A. Project Plan management

Week	Times	Project phase	Detail task	Note
1	09/05 - 15/05	Idea and planing	Create timeline and proposal	Done
2	16/05 - 22/05	Collecting material	Finding dataset and sample model	Done
3	23/05 - 29/05	Training model	Process data set and training model	Done
4	30/05 - 05/06	Fine-tuning model	Retraining with another parameters	Done
5	06/06 - 12/06	Testing	Testing in real environment	Fail by noise
6	12/06 - 19/06	Change method	Find new method for direction control	Done
7	20/06 - 27/06	Training model	Preparing dataset and training	Done
8	28/06 - 03/07	Testing	Testing in real environment	Pass
9	04/07 - 10/07	Writing report	Write a detail report for project	Done
10	11/07 - 16/07	Recognition improve	Improve the accuracy of sign recognition	Done

Appendix B. Source code and data

- Source code: Github
- Dataset: Google Drive
- Another documents: Google Drive

Appendix C. Contributions

No.	Contributor	Task	Contribution rate
1	Anh Phan Ngoc (Leader)	Traffic sign recognition & control signal	35%
2	Thinh Ngo Duy	Direction control	35%
3	Truong Nguyen Nhat	Traffic sign detection	30%