Traffic sign detection on CARLA

1st Ítalo Vinícius P. Guimarães
Universidade de Brasília
Special software engineering topics
Brasília, Brazil
italo.pereira@unb.com.br

2th Guilherme Mendes Pereira
Universidade de Brasília
Special software engineering topics)
Brasília, Brazil
guilhermemendes@aluno.unb.br

***Abstract*—This article is the final project of the course Special software engineering topics, and we tried to implement traffic signs detection on CARLA using deep learning and CNN methods, we are focusing just on stop and speed traffic signs.**

***Index Terms*—component, traffic sign, CARLA, deep learning, CNN.**

I. Introduction

Nowadays the world are talking a lot about self driving cars, and this is very important point for the future because it will help us all to lower the quantity of traffic-accidents and fix the human errors, but how can we teach the car to drive better than a human ? Some new technologies are finding a way to do this, using different types of solutions, like new kinds of radar, so we can now have the information about the environment around the car. But the biggest question is, what we do with this data ? This century are marked by how fast the technologies are upgrading and just in the past fifty years we find how to teach a computer to do tasks, but nowadays we changed the manual way to a automated way to teach something. Using a lot of machine learning methods, this area includes some of the tools we gonna use in this project, like neural networks and deep learning. So that the way to convert data to learning.

II. Development of the project

The project was developed using tools that make easier the way to simulate the code and get real situations like in the real life, Obviously the way which is implemented in the simulator is very different from the real life, but makes a good job to start researching in this area of self driving cars, we gonna use CARLA simulator, deep learning methods like Yolo and optical character reader (OCR) to do, in few words, respectively, simulation on an environment of driving car, Yolo to get the traffic sign and OCR to generate the machine-encoded text

A. CARLA

1) What is CARLA: CARLA is a simulator that are made to make easier the way to control all the environment, like the people which are walking, the climate conditions, other cars in the same map, etc. But over all, CARLA is made to generate data and control the car, but why we need this ? Imagine in the real world if you implement a script to break when see a stop sign, but you need to do repetitively tests in the real life, imagine how this is boring or in other tests how is dangerous. So CARLA was made to you get the data and test the best way to apply the information of this data to control the car, so you can let the script running or the algorithm learning the best way to drive, that's the reason why CARLA is a good option to test in machine learning methods

Fig. 1. Carla running and information.

2) How it works ?: Imagine that you have a game running in your computer and you have a wireless control to play the game, CARLA works like that but the wireless control now is a server, which is implemented on python and have libraries to made the communication. In other worlds works like a client-server architecture. In the server you can control the variables of the map inclusive choose the map and you can get too the traffic violations, that's the reason why a lot of users are using CARLA to make generative adversarial networks or genetic algorithms, because, in few words, you can teach the best way to drive giving to the script the thing they "can't do".

3) How to get and chose the data ?: In CARLA all the objects that you can put in the map are named actors, so the car is an actor and the sensors too, but CARLA have a lot of sensors like RGB camera,Radar, Lidar, some sensors just exists in CARLA, like Semantic sensors. But in this project we are using RGB camera because we working with Yolo, so we can get images of each frame of CARLA.

B. Yolo

Nowadays we have a lot of deep learning methods to get data of the world, but the most famous is the Yolo acronym of You only look once, the name means a lot but this algorithm are the ideal to us because we can apply in the data of RGB

camera and get the information of the object and localization of it in the image, so the model give to us the traffic sign.

1) How it works ?: This network read the frame (image) and turn into a mosaic, separating the raw images in new small images, from that the networks make squares named bounding boxes (bbxs) of each pre-trained object and set a accuracy of it, when merge the image again they erase the bbxs of lower accuracy than a threshold and merge the same labeled bbxs using calculus. That's one of the reasons why we choose this network, because works fast and with a great accuracy. But the major reason it's because the model are already trained to recognize traffic signs.

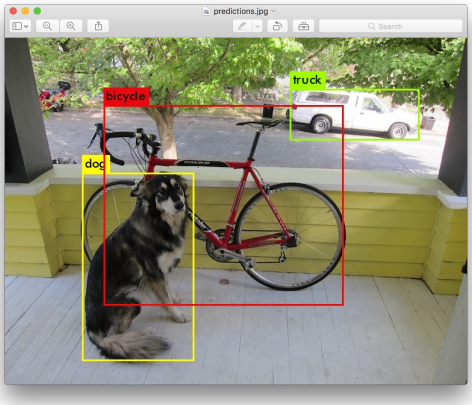

Fig. 2. Example of a Yolo network frame.

But we have some problems, first the model are recognizing the traffic signs, but we isn't labeling them like "speed sign", "attention sign" or others, the model are made just to label them as "stop sign", that's the reason why we gonna implement a pos-processing in the traffic sign, we don't no if it's the best way to implement that but we gonna do this to find if worth it, but how we gonna take the information of the traffic sign ?

C. Optical character reader or Deep learning method (trying)

Optical character reader (OCR) are made to turn raw information like images, scanned documents, or documents that we can't change to a machine-encoded text. But what this means ? Image that you have a brain and learn how to process the information of the world to a understandable information, thats the same of OCR. To computer, a image it's just only a matrix o RGB channels, composed by numbers, and our intention is to turn this matrix to usable data on CARLA.

OCR works like a pattern identifier, so it will get the image, pre-process (rotating, fixing the angles, using filters), pass of each image finding patterns of letters, matching the ones which had best accuracy of them and saving them in a writable document. But this is a very old way to do this.

But imagine now if we join AI and OCR, so the results will be very better than just use a mechanical tool, Tesseract is a open-source tool to this this exactly task, so we can be

Fig. 3. What OCR do ?

offline and this tool will still working to translate the image to text information. This model work very well on low quality images and now are maintained by users and developers but it was a tool stated by google.

We removed the option to use Vision API because we aren't getting results with our data. Other situation is imagine the car can't access internet, so using Tesseract the solution is implemented inside the car, reducing a lot of problems of connection and waiting for a response.

III. Applying Method

The proposed system comprises two main steps: detection and recognition. The detection phase explores the knowledge of the structure of the scene, that is, the size and location of the road in the frame, to determine the regions in the scene that it should Look for road signs. These regions are defined once the vanishing point (VP) of the scene and therefore the ground plane are determined. Candidate regions for road signs are then located only within these scene search regions, using a combination of MSERs and threshold, saturation and hue color value (HSV). Combining these regions through consecutive frames, Time information is used to further eliminate the detected PF regions, based on the movement of regions in relation to camera and the scene structure. After a potential traffic signal has been located by YOLO, the next stage of the algorithm tries to recognize the text within the region.

First, a rough perspective transformation is applied to the region to vertically align the characters in the text. Candidate components for text characters are then located within the region and classified into possible lines of text, before being interpreted using optical shelf character recognition (OCR) package. To improve recognition accuracy, OCR results multiple frames are combined by combining individual words between frames and using a weighted histogram results.

But, in the start of this task we was suffering a lot, mostly because we were trying just to run Tesseract with some cropped images that we already had, the first version which we were working was Tesseract 3, but after applying image_to_string() function, we don't get any result, even if we pre-processing these images, using grayscale or filter in the image.

So we found on Tesseract github that Tesseract 4 works better than Tesseract 3, so we upgraded the version of our current program. After that we get some results, but wasn't the best, because in a folder with aproximadely 210 images,

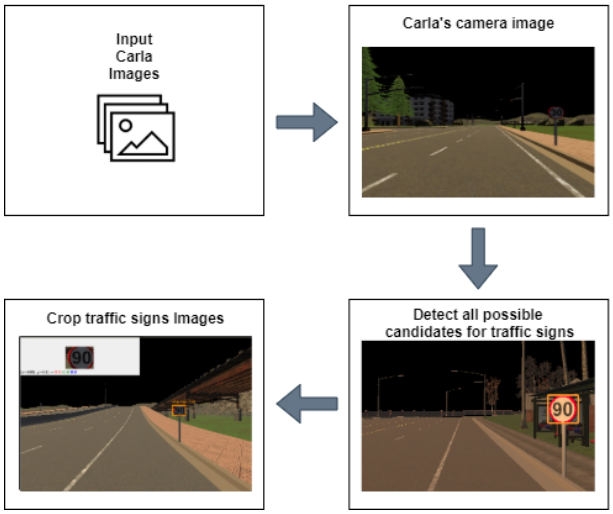

Fig. 4. Detection

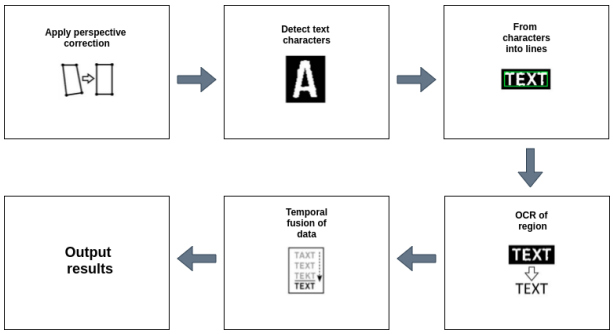

Fig. 5. Recognition

we get results of 4 or 5 images, because some of then were resulting in a string without any information. The problem wasn't the quality or the size, because some images with small size are resulting in correct string, but other bigger aren't.

Some of good cropped traffic signs that wasn't returning to us the information

Anyway, we though that the script which was getting images was generating a lot of images, so we applied these result images in a new function, this function use Tesseract to do this task (translate images to strings) and we created a dictionary to translate the results because analising the results we realized that we were getting a lot of empty strings, so using the dictionary we can analyze if the string contains specify letters or numbers,

The dictionary is signs = 'T': 'Stop', 'P': 'Stop', '3': 30, '6': 60, '9': 90 the reason why we made a dictionary using these keys it's because the char 'T' and 'P' are the most different letters to numbers, when we was using the word 'S', we saw how the Tesseract was confusing 6 or 9 with S, and war returning a wrong value to us. So we implemented like the diagram below.

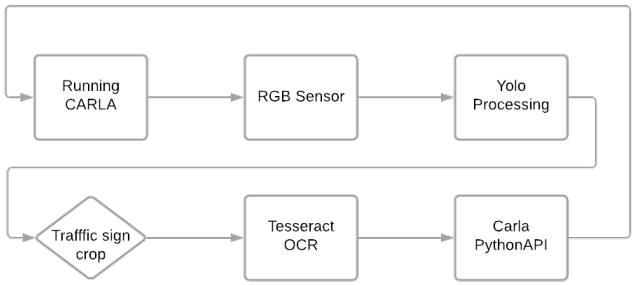

Fig. 6. Project flow

So the scripts are working in that way, we stipulate this way because we know that the crops of signs is feeding every time, when detected, the script of translation of information, so if we get a result we can return this result to our 'user', we are using like a warning, printing in the simulator the result of the detected traffic signs.

We tried to pre-process and resize the images in this version of Tesseract, but inside of it has these tools, so we didn't manipulated the image to input in the function image_to_string(), we made this option because studying about Tesseract we see that this manipulations can downgrade the dpi and resolution of image and a good number of DPI is essential to Tesseract work well. We can see how Tesseract works in the image below.

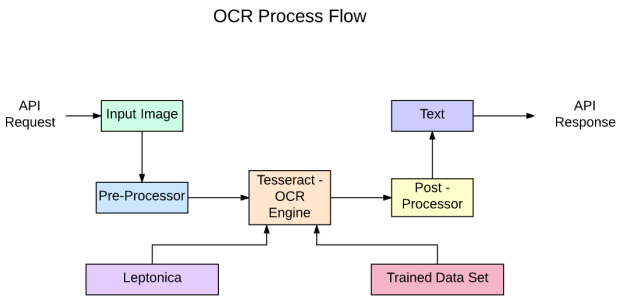

Fig. 7. Tesseract OCR flow

IV. Conclusion

The results of the working script are in the image attached below.

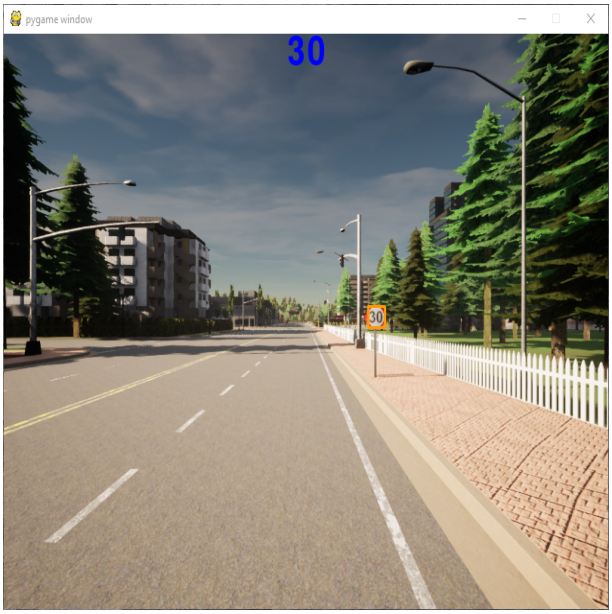

Fig. 8. 30km/h Speed Traffic Sign detected embedded on the screen

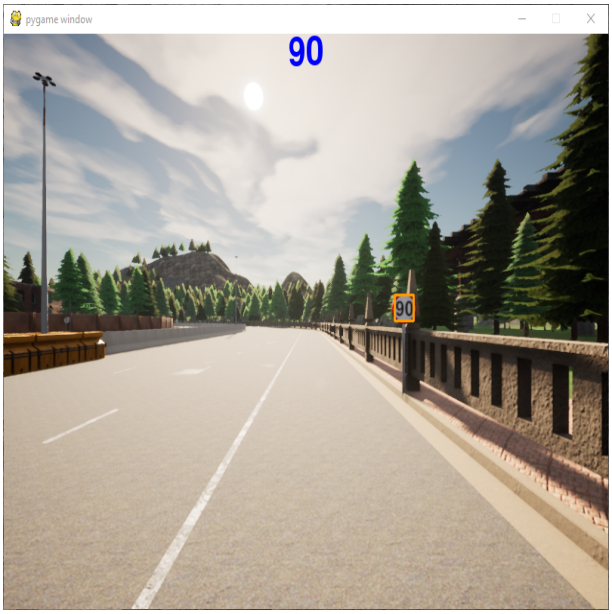

Fig. 10. 90km/h Speed Traffic Sign detected embedded on the screen

Fig. 9. Stop Traffic Sign detected embedded on the screen

I believe that we have done well with what we have accomplished, but there are still many problems that we have to solve, however the developed solution is functional, many of the problems involve hardware requirements, however, some of the problems are:

- Slow processing of yolo
- Crashes when a card is recognized
- Little variety of plates in the carla so that there is a better verification of the robustness of the model

In addition, the solution based on a manual process is obviously slower, so this model we created was made focusing on a result, obviously it was not the best or the fastest, some examples to create more robust models are:

- Train the yolo with mined data, identifying species of plates and their content, that is. it would be a model only that would do both tasks
- Use or train natural language processing models so that it can be used specifically in the carla simulator, obviously generating better results that do not require post-processing

Even with these problems we managed to comply with what we promised, that is, we were able to identify the traffic signs and we can check the information that is inserted in it, which can be used either to mine data or to actually apply in tests within CARLA.

In relation to our learning in the subject and in the project, I believe that we have grown a lot in relation to the knowledge of autonomous car features, we did not know that it was such a complex process, but we understand that it is made specifically for the safety and comfort of people. Regarding technologies, we already had a little knowledge about CARLA, but what added us most was the use of Tesseract, a tool that helped us a lot, and does about 50

I think in the most, we really enjoyed developing this project, we didn't know it would be something so complex, but it was a challenge that we chose and managed to successfully accomplish

If you want to see our model you can just access our repo using this link https://github.com/guilherme-mendes/TEES-Autonomous-vehicles

REFERENCES

[1] Martí Sánchez Juanola.Speed traffic sign detection on the CARLA simulator usingYOLO. https://repositori.upf.edu/handle/10230/42548.

[2] Joseph Redmon and Ali Farhadi. “YOLOv3: An Incremental Improvement”. In:arXiv(2018).
[3] Adrian Rosebrock.Traffic Sign Classification with Keras and Deep Learning.https://www.pyimagesearch.com/2019/11/04/traffic-sign-classification-with-keras-and-deep-learning/. Nov. 2019
[4] umtclskn,Umut Çalışkan,Carla Simulator YOLOV3 Object Detection,aug,2020,https://github.com/umtclskn/Carla-Simulator-YOLOV3-Object-Detection
[5] Jeonghun Baek et al. “What Is Wrong With Scene Text Recognition Model Comparisons?Dataset and Model Analysis”. In:International Conference on Computer Vision (ICCV).2019. published.
[6] Anthony Kay. 2007. Tesseract: an open-source optical character recognition engine. Linux J. 2007, 159 (July 2007), 2.
[7] @miscumtclskn,Filip Zelic Anuj Sable,A comprehensive guide to OCR with Tesseract, OpenCV and Python,sep,2020,https://nanonets.com/blog/ocr-with-tesseract/