Literature Survey - Self-driving Car Using Convolutional Neural Network and Road Lane Detector

[1]Jayant Waghmare, [2]Amir Suhail, [3]Rishab Mishra, [4]Eadhunath V, [5]K.U.Jadhav
[1]Student, [2]Student, [3]Student, [4]Student, [5]Assistant Professor
[1]Department of Computer Engineering,
[1]Sinhgad Academy of Engineering, Pune, India

Abstract : The advancement of Computer Vision these days has grown up beyond imagination. Recently, many researchers and tech companies are competing to develop self-driving car using either traditional or deep learning approach. YOLO (You Only Look Once) is one of the real-time CNN methods that aim at detecting objects from images. On the other hand, Road Lane Detector is used to detect road-track in video frames and to provide additional information that can be helpful for the decision-making process of the self-driving car.

Index Terms **- YOLO, CNN, Deep Learning, Road Lane Detector.**

I. INTRODUCTION

In India, according to a survey, more than 150,000 fatalities are caused each year in traffic accidents. That's about 400 fatalities a day and far higher than developed auto markets like the US, which in 2016 logged about 40,000. Nearly 55% of these involve 4 wheel vehicles or buses.

The idea of a self-driving car is an effort to minimize the accidents caused by careless and violent driving of 4 wheelers. The car should be such that it can recognize the distinct objects found along the roads and identify the elements of interest such as other vehicles, pedestrians, traffic signals etc. Another major challenge is building its ability to navigate properly in between the road lanes while complying with the traffic rules. In order to make a sustainable self-driving car, we need three main parts: object detection, lane detection, and controller. In this paper, we have combined these parts to help the driver with more info about the object in their driving environment and the road lane guidance. The proposed method combines YOLO for the object detection, polynomial regression with thresholding as the road lane guidance, and a controller that manages the information between those two systems. The proposed methods can be used for object detection, object position detection (left, front, or right), steering suggestion, and road lanes guidance.

There have been various different versions and improvements to the algorithms used in this proposed system and this paper covers many of the popular ones, comparing them simultaneously, clearly showing the advantages and disadvantages of each.

All survey done in this paper consists of the researches done by various analysts, their methodologies and theoretical formulae and conclusions each arrived at.

II. LITERATURE SURVEY

A. YOLO : Object Detection

For detecting simple objects from images captured by cam- era, a deep learning technique named YOLO - You Only Look Once, was proposed. YOLO was a new approach to object detection. For object detection the previous works simply used the repurposed classifiers. Here, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. From full images a single neural network predicts bounding boxes and class probabilities directly in one evaluation. The detection pipeline can be optimized directly on detection performance as the detection is single network.

At 45 frames per second the base YOLO model processes images in real-time. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. YOLO makes more localization errors but is less likely to predict false positives on background as compared to state-of-the-art detection systems. Finally, YOLO learns very general representations of objects. It performs better than other detection methods like DPM and R-CNN when it comes to generalizing from natural images to other domains like artwork.

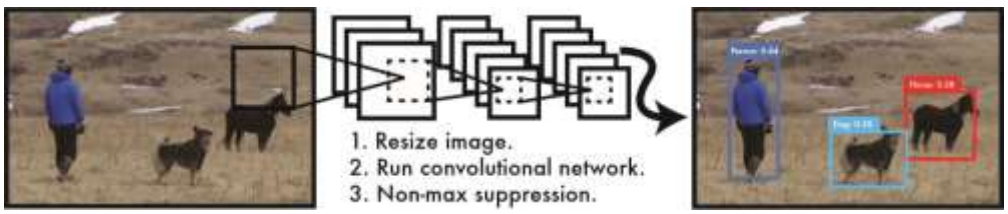

fig. 1. **YOLO detection system.** Processing images with YOLO is simple and straightforward. The system (1) resizes the input image to 448 x 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the models confidence.

Current detection systems repurpose existing classifiers to perform detection. For detection of an object, these systems rely on classifier for that particular object and then evaluate it at various locations and scales in a test image. A sliding window approach is used in systems like deformable parts models (DPM) use where the classifier is run at evenly spaced locations over the entire image. Modern approaches like R- CNN implement region proposal methods to first generate potential bounding boxes for an image and then run a classifier on these proposed boxes. After classification is done the post- processing is used to refine the bounding boxes, eliminate duplicate detections, and rescore the boxes based on other objects in the scene. As each individual component must be trained separately, these complex pipelines are slow and hard to optimize.

Firstly, YOLO as an algorithm is extremely fast. The neural network is simply run on a new image at test time to predict detections. The base network can run at 45 frames per second without batch processing on a Titan X GPU and a faster version is capable at running at more than 150 fps. Which implies that we can process streaming video in real-time with latency of 25 ms or less. Furthermore, YOLO has more than two times the mean average accuracy of other real-time systems available.

Second, YOLO considers whole image while making pre- dictions. Opposed to sliding window and region proposal- based techniques, YOLO sees the entire image during training and test time and it implicitly encodes contextual details about classes as well as their appearance. Fast R-CNN is a top detection method but it mistakes background patches in an image for objects as it can't see the larger context in the image. On the other hand YOLO makes less than half the number of background errors compared to Fast R-CNN.

Third, YOLO learns generalizable depiction of the objects in the image. When trained on natural images and tested on an artwork, the YOLO outperforms several top detection methods like DPM and R-CNN by a wide margin. When applied to new domains or unexpected inputs, YOLO is less likely to break down as it is highly generalizable.

Limitations of YOLO Detection System: YOLO implies some strong spatial constraints on bounding box predictions as each grid cell can only predicts two boxes and can only have single class. This spatial constraint limits the number of surrounding objects that our model can predict. Our model struggles with small objects that appear in groups, such as group of people. As our model learns to predict bounding boxes from data, it finds difficult to generalize to objects in new or unusual aspect ratios or configurations. Our model also uses relatively coarse features for predicting bounding boxes since our architecture has multiple down sampling layers from the input image. Finally, while training on a loss function that approximates detection performance, our loss function treats errors the same in small bounding boxes versus large bounding boxes. A small error in a large box is generally benign but a small error in a small box has a much greater effect on IOU. Our main source of error is incorrect localizations.

B. Road Lane Detection

The road lane detector consists of four main parts: warping, filtering, detecting the road lane, and de-warping. The initial step is the warping process. In this we crop the Region of Interest(ROI) and change the perspective of the image. We can use the OpenCV method for changing the perspective. *cv2.warpPerspective()* in Python. Figure 2 shows the resultant of this process. It clearly represent how the way in which the road's perspective has been changed so as to make computations easier.

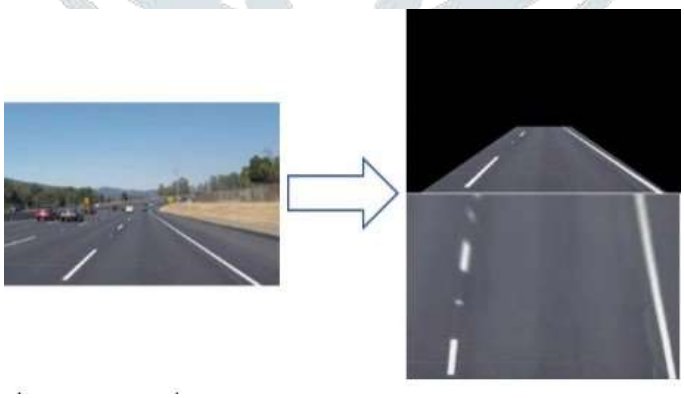

fig. 2. warping process

The next step is to filter this cropped and warped image to differentiate between the road points and the background of the image. In the filtering process, we pick the range of white and yellow color out of the images as most road lines are painted white or yellow color. After the filtering phase, we detect the road lane points by getting the non-zero readings from the previous process. First, we crop the images into 10 sub-images and divide them into left and right. This helps us in getting the peak of nonzero values of each lane (left and right). Finally, we add the area around the peak values to the list of left and right points into the collected points.

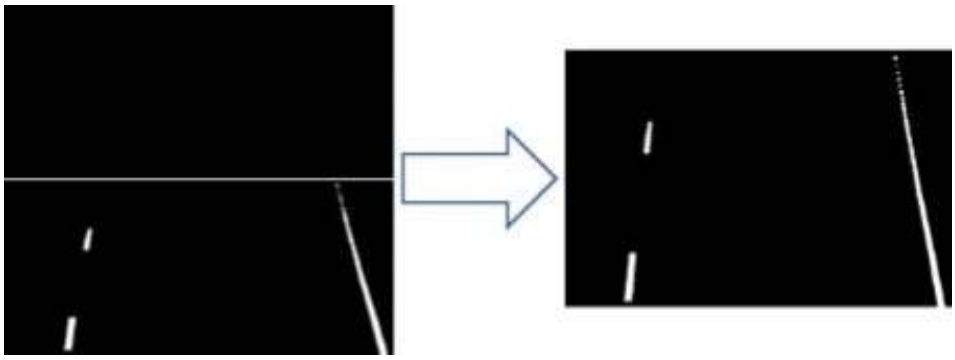

fig. 3. filtering process

The next step is obviously the road-lane detection for which the polynomial regression is used. This can be achieved easily using the Numpy function *np.polyfit()* in Python. It enables us to approximate the lane/curve by fitting lane points (y) that have been collected with coefficient points (p) into the best approximated lane points (Polyfit) by minimizing the error (E) from the fitted lanes. Hence, by getting the right and left points, we could make functions to approximate the lane structures by feeding them into the regression method to find the best fit of the road lane detection.

The final step is to now convert this processed image into a real-time projection image by reversing the warping process that was applied in step 1. This process is called as *de-warping*. This is shown in figure 4.

Limitations of Road Lane Detection : In the road lane detector, we used some filtering and warping before we detect the lane points. The image filtering limits us to accommodate the undetected road lanes and restraints us from a robust detection system. Also, sudden over lighting/illumination to recording camera/environment also greatly affects the detection of our methods even after applying normalizations, i.e. Contrast Limited Adaptive Histogram Equalization (CLAHE) and gamma correction. The road lane detector also uses the crop of left and right to find the lane points in both lanes. This approach is works well for straight and non-sharp turned roads. However, it brings out a limitation, where the sharp-turned roads could lead to inaccurate detection.

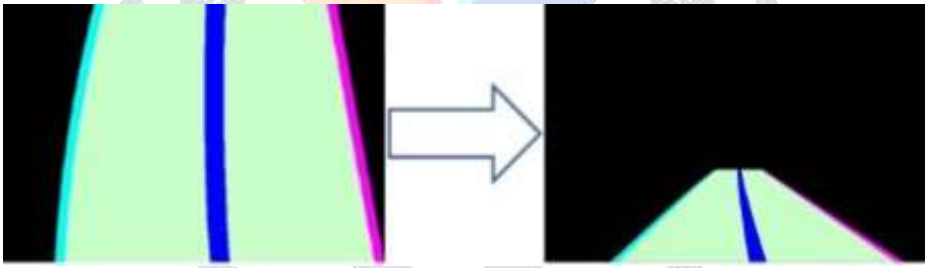

fig. 4. road lane detection and de-warping processes.

C. Controller

Prior running YOLO and road lane detector, we require the information about the general road lane width in the pre-setting configurations to modify the road width and the offset length. For our controller, we will be using the combination of our Laptop with NVIDIA 960M Graphic Processor for performing the processing of images and running the neural network and using Arduino Uno microcontroller to control the motion of the motor and wheels of our prototype car. After the controller loads the camera/video frames, it feeds them into YOLO and Road Lane detector. These 2 processes run simultaneously. YOLO provides the object detection and Road Lane detector provides the steering suggestions.

The main tasks of the controller can be described in seven stages :

- YOLO object detection function
- Road lane detector function
- Draw the detected lane
- Object Position
- Update the runtime
- Show the detection results
- Pass values to control the speed and direction of the car.

III. ACKNOWLEDGMENT

We would like to thank our Project Guide Prof. K.U.Jadhav for guiding us through the process of making a literature survey. We would also like to extend our thanks to our University for giving us a chance to work on this topic.

REFERENCES

[1] Towards Self-driving Car Using Convolutional Neural Network and Road Lane Detector.

[2] You Only Look Once: Unied, Real-Time Object Detection. Authors : Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi

[3] YOLO9000: Better, Faster, Stronger, IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Authors: J. Redmon and A. Farhadi

[4] John Canny : A Computational Approach to Edge Detection