Applied Computer Vision on Advanced Driving Systems

Abraham Rodriguez Zepeda, Mechatronic Engineer[1]. Alicia María Reyes Duke[2], Master's degree in engineering. and Rigoberto Castro Castro, Master's degree in engineering[3].

[1, 2, 3] Universidad Tecnológica Centroamericana (UNITEC), Honduras, abrahamrodz17@unitec.edu, alicia.reyes@unitec.edu.hn, rigocastro@unitec.edu,

Abstract*– As driving systems, and autonomous vehicles are developing, computer vision plays a vital role on the perception of these systems, this allows vehicles to detect the drivable area, or the area between lanes, and detect objects, with the purpose of reacting to its environment, generating a certain level of autonomy. In this paper, an algorithm based on computer vision techniques such as sliding windows, color thresholding, edge detection, and perspective transformation are used for lane detection, once the lanes are detected the left and right curves can be fitted with polynomial regression, obtaining the drivable area, which is the area between the detected curves. For object detection, a pretrained version of YOLOv5 is implemented. The algorithm was implemented using a custom dataset generated on various sectors of San Pedro Sula, Honduras, showing promising results on different scenarios. For a dataset of 1705 processed images, the lane detection accuracy of the algorithm under ideal conditions is 94%.

Keywords—ADAS Systems, Autonomous Vehicles, Lane Detection, YOLOv5.

I. Introduction

Nowadays, autonomous vehicles and ADAS systems are a highly active research area, it's currently in search of techniques that develop vehicle autonomy, driving assistance, collision avoidance, and computing power reduction. For safer self-driving vehicles, one of the problems that have yet to be solved completely is lane detection. Since methods for this task must work in real-time (+30 FPS), they not only have to be effective (i.e., have high accuracy) but they also must be efficient (i.e., fast) [1]. In vehicle perception, lane and object detection are the main problems to be solved. Object detection refers to identifying the location and size of objects of interest. Both static objects, from traffic lights and signs to road crossings, and dynamic objects such as other vehicles, pedestrians or cyclists are of concern to autonomous driving systems [2] In this paper, the general problem is the vision system for autonomous vehicles and ADAS systems; the relevance of this paper is based on the actual tendencies of the search on reducing, and optimizing the use of specialized sensors on autonomous vehicles, by depending on camera focused systems, using computer vision techniques, and neural networks, this greatly simplifies the systems of the autonomous vehicle in general, which also induces the reduction of computing power. This paper uses a dataset generated in San Pedro Sula, Honduras; it's worth mentioning that, to the WHO, Honduras does not have standards and laws for vehicular safety [3]. This paper presents an algorithm based on traditional computer vision techniques for lane detection and the implementation of YOLOv5 for object detection.

Digital Object Identifier (DOI):
http://dx.doi.org/10.18687/LACCEI2022.1.1.222
ISBN: 978-628-95207-0-5 **ISSN:** 2414-6390

II. Context

A. Related Work

There has been a numerous studies and works regarding computer vision on diverse areas, such as Industrial applications, Security, and Agricultural applications for recognition such as [4]. Computer vision studies are numerous on ADAS systems, and autonomous vehicles for lane detection, vehicle detection, etc. There is a difference between various research, that is the usage of traditional computer vision techniques, and the usage of neural networks for lane detection. Rezwanul Haque et al. [5]. proposed an algorithm, based on computer vision techniques, using HLS thresholding for color characteristics, and gradient thresholding for edge detection, and sliding windows, tested with the KITTI dataset, the system got good results, getting an accuracy of 84%. However, it was tested using a small quantity of images, consisting in 289 images.

K. C. Bhupathi and H. Ferdowsi [6]. proposed an algorithm, similar to [5] by using computer vision techniques such as HSV thresholding for white color extraction, and a canny detector for edge detection but it applied cache for lane detection, this causes that the initial position of the sliding window considers the previous frame initial positions, this approach increases the confidence of the windows starting point for lane detection. However, the algorithm, was tested indoors, and under ideal conditions, but it considered sharp curves. It got high results with an accuracy of 96.26%.

Rohit Gandikota [7]. studied various computer vision techniques on both indoors and outdoors, for outdoors vehicles, a lane detection technique based on canny edge detection and Hough transforms, was applied, also a vehicle detection based on HOG features, this study is a good example of all the necessities of a self-driving car, however for lane detection a Hough transform technique is sensible to noise and does not perform well under various conditions.

M. Gluhaković, M. Herceg, M. Popovic and J. Kovačević [8]. proposed a method for vehicle detection for autonomous vehicles based on ROS and YOLOv2, trained on a custom dataset with data gathered from known datasets such as KITTI, ImageNet, PASCAL VOC, and frames obtained from a camera, this study tested the algorithm on different weather conditions, and on Carla Simulator, getting an average of

59.48 IoU accuracy. It concluded by mentioning that using YOLOv3 would be more accurate results, at the cost of higher computing power.

B. *Software and Tools*

Python 3.7, OpenCV, Numpy, Matplotlib, and Threading were the libraries used for the lane detection algorithm with an Intel i5 8300h CPU, developed on Visual Studio Code. Carla Simulator was used for testing the algorithm running on real time, with an Nvidia GTX 1050 4Gb GPU. Google Colab with GPU acceleration was used to implement YOLOv5 to the dataset.

III. METHODOLOGY

For the realization of this project, an incremental methodology approach was applied, divided on three increments or stages:

- First Increment: Lane Detection
- Second Increment: Object Detection
- Third Increment: Integration

Each increment is divided in various phases as shown on Fig. 1.

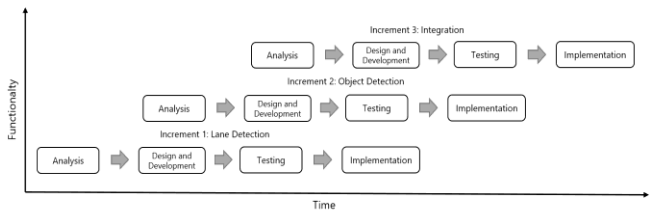


Fig. 1 Incremental Methodology.

A. *Dataset*

To apply the algorithm on San Pedro Sula, it was necessary to generate a dataset, this was accomplished using a smartphone camera recording at 1080p at 30 fps, mounted at the center of a car dashboard, 5 videos were generated, accumulating a total of 11.30 minutes or 20,340 total frames. In order to reduce computing power, the resolution was down scaled to 480p, this was accomplished by using Python and OpenCV. The algorithm was implemented on a sample of 5072 frames or 25% of the total population, for the validation of the results, 1705 frames or 8% of the total population were analyzed. In Fig. 2. the population is shown.

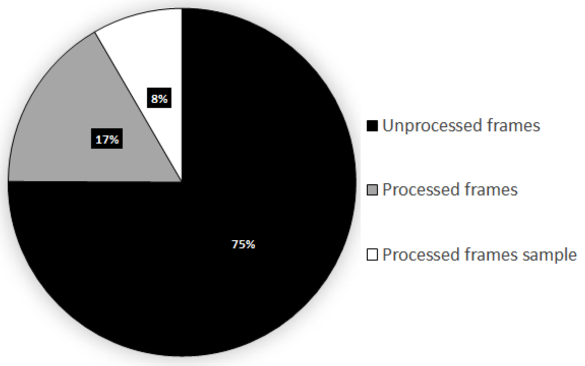


Fig. 2 Dataset.

B. *Increment I: Lane Detection*

To perform lane detection, an algorithm based on computer vision techniques is applied, this process is shown in Fig. 3.

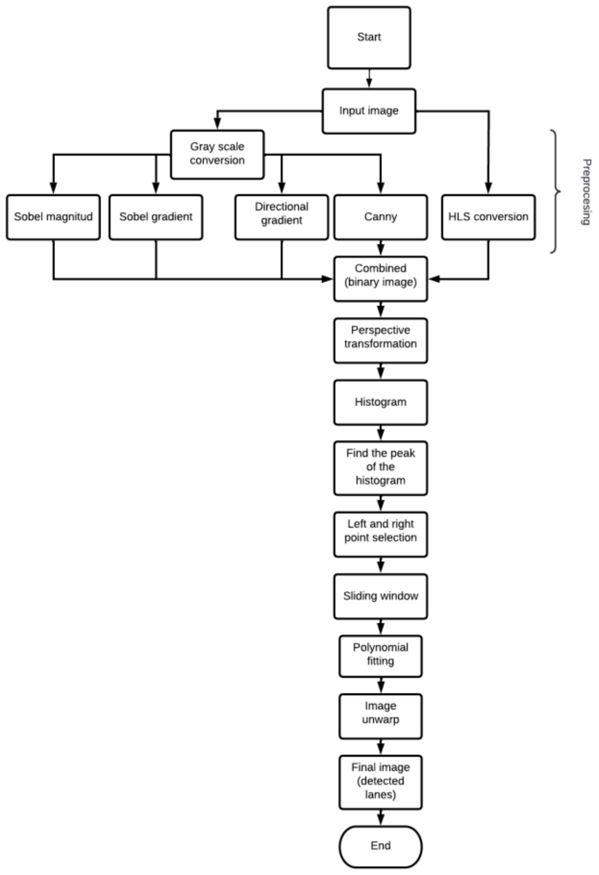


Fig. 3 Increment I

1) Preprocessing:

After the dataset is defined, the images get preprocessed, grayscale conversion is applied to reduce computational complexity, then border detection is applied, both Sobel magnitude, Sobel gradient in the x-direction, the Directional gradient, and a Canny detector are used. Also, the input images get converted to HLS (Hue, Lightness, and Saturation) color space, where the S channel, was thresholded to emphasize the colors of the lanes. The result of the preprocessing is a binary image that consists of the combinations of the methods described before.

2)Perspective Transformation:

Once a binary image is generated. It's necessary to focus on the ROI (Region of interest) of the image. In this case, the ROI is the area that contains the lanes. This makes it easier to analyse the lanes contained in the image. By applying a linear transformation called perspective transformation, with points marking the ROI area, the result is a bird's eye view of the ROI. This is achieved by applying equation (1).

$$\begin{pmatrix} x_1' \\ x_2' \\ x_3' \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \quad (1)$$

3)Histogram:

A histogram is applied to find the highest frequency or peaks of white pixels generated from the binary image, which ideally are the left and right lanes, this can be found by applying the argument of the maxima of the histogram. After the peaks are found, points on the left and right are selected, as starting points for sliding window search.

4)Sliding Window Search:

Once the peaks of the histogram are found, this serves as starting position for the initial window on the sliding window search, the next window is estimated by calculating the mean of the white pixels inside the window area within an established margin, this process gives the position of each window, which depends on the number of established windows. By having various points that correspond to both left and right lanes, polynomial fitting can be applied to estimate the lane curves, and finally obtaining the area between the curves or the drivable area. In this case a polynomial of second order is obtained by using equation (2)

$$y = a + bx + cx_2 + dx_3 + \cdots \quad (2)$$

5)Final Image:

On the final image, the lane area is detected and shown in green. This result and all steps mentioned are shown in Fig. 4.

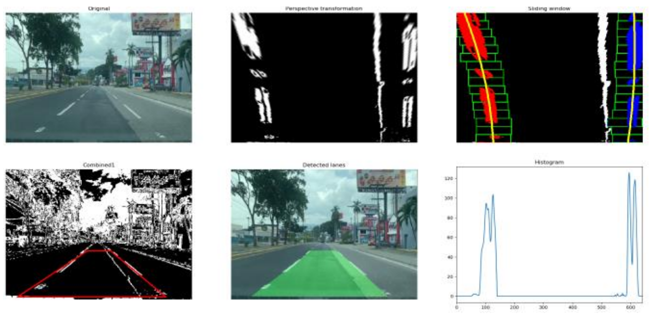
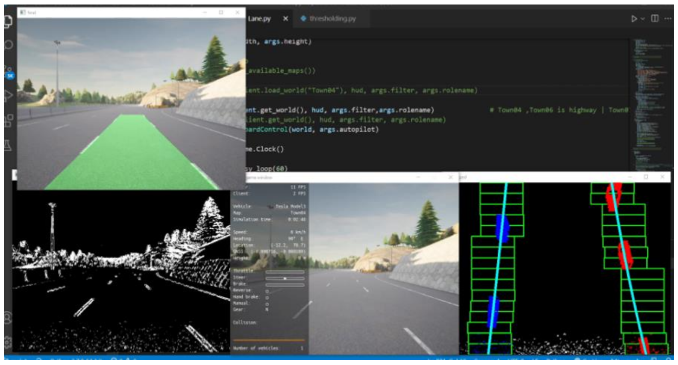


Fig. 4 Increment I: Complete Process

6)Testing:

Using the generated dataset, the algorithm was tested on random frame samples from each video, this allowed for corrections and improvements. Carla Simulator was used to test the lane detection algorithm running on real time, on a local machine, see Fig. 5. However, these results were not quantified due to the low hardware available, getting an average of 8 FPS, causing frame drops, making it unreliable for results calculation.



```
dth, args.height)

_available_maps())

ient.load_world("Town04"), hud, args.filter, args.rolename)

ent.get_world(), hud, args.filter,args.rolename)          # Town04 ,Town06 is highway | Town0
lient.get_world(), hud, args.filter, args.rolename)
oardControl(world, args.autopilot)

me.Clock()

y loop(60)
```

Fig. 5 Testing on Carla Simulator

C. *Increment II: Object Detection*

This increment consists of the application of the YOLOv5 neural network, for object detection, this neural network was pre-trained, using the COCO dataset. Using Google Colab and GPU acceleration, the neural network was cloned from the YOLOv5 Github repository, and applied to each video, getting, as a result, a video in .mp4 format. This process is shown in Fig. 6.

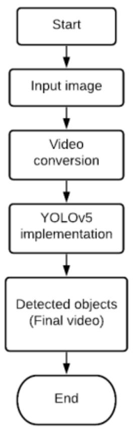


Fig. 6 Increment II: Object Detection Process

D. *Increment III: Integration*

This increment consists of the integration of the first and second increments, to accomplish this, a similar process to the second increment was applied, in this case, the dataset is already processed by the lane detection algorithm, this is applied to every video, getting the result in .mp4 format. This process is shown on Fig. 7. The final result is shown on the Fig. 8.

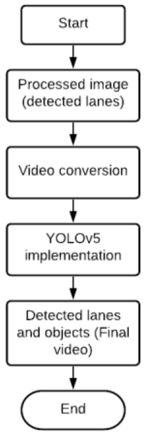

Start
Processed image (detected lanes)
Video conversion
YOLOv5 implementation
Detected lanes and objects (Final video)
End

Fig. 7 Increment III: Integration Process

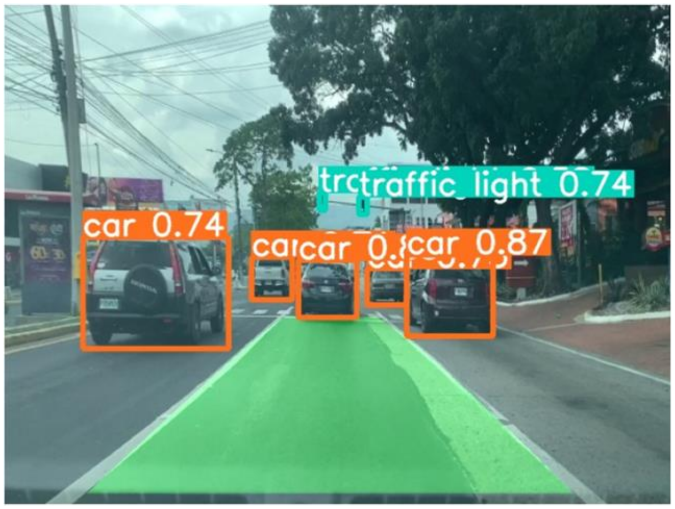


Fig. 8 Final Result

IV. Results

As the algorithm was applied on different videos under different conditions, all videos are used to test the accuracy of the lane detection algorithm, on Table. I. The quantity of analyzed frames are shown. This quantity corresponds a sample of 8% of the total population, as shown in Fig. 2.

TABLE I
ANALYZED FRAMES

Frame #					
Video 1	***Video 2***	***Video 3***	***Video 4***	***Video 5***	***Video 6***
252	252	301	300	300	300

A. *Considerations*

This algorithm was tested on a dataset generated during daytime, also due to the nature of roads, it's common to find various conditions of roads, that why the conditions are classified as ideal and nonideal based on the assumptions shown on Table II. Every video is classified based on the conditions stated on Table II, shown in Table. III.

TABLE II
ROAD CONDITIONS

Conditions	
Ideal	***Nonideal***
Visible lanes Lack of potholes Lack of trash Lack of road cracks	Nonvisible lanes Presence of potholes Presence of trash Visible road cracks

TABLE III
DATASET CONDITIONS

Video #	***Conditions***
1	Ideal
2	Ideal
3	Nonideal
4	Ideal
5	Nonideal
6	Nonideal

As mentioned on the methodology, a pre-trained model of YOLOv5 was implemented, due to that, it lacks labeling, making it difficult to obtain results without training the model with a custom dataset. Because of this, results for object detection are neglected. Also, half of the videos, lack from objects of interest, only videos 2, 4 and, 5 count with various objects.

B. *Result Analysis*

The results obtained by the lane detection algorithm show that, under ideal conditions it behaves quite well, some samples of both ideal and nonideal conditions are shown on Fig. 9. and Fig. 10.

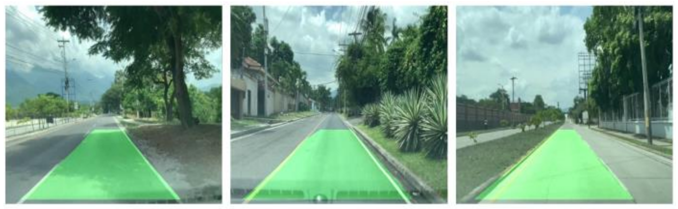


Fig. 9 Ideal Conditions Samples

Notice that under nonideal conditions, there is noise caused by cracks on the roads, and some frames lack a side of lanes. In some frames the algorithm showed promise under nonideal conditions as shown on Fig.11. Notice that one lane is replaced by a crack on the right side of the road, and the left lane is discolored, also there is presence of sand.

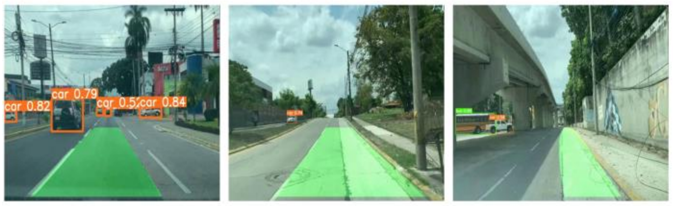


Fig. 10 Nonideal Conditions Samples

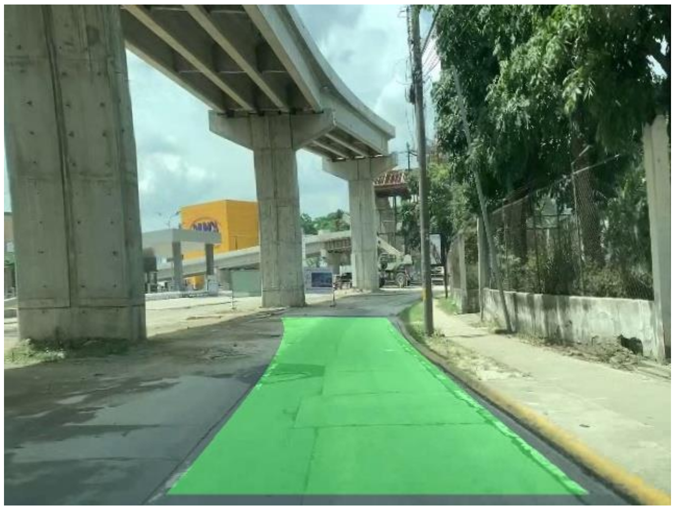


Fig. 11 Nonideal Conditions Sample of Successful Detection

In some special cases as on Fig. 12. there are road markings e.g., arrows and letters. Under these special cases it's expected to fail due to the similarities between lanes and markings, being the same color, and border presence. These cases cause drops in the algorithm precision.

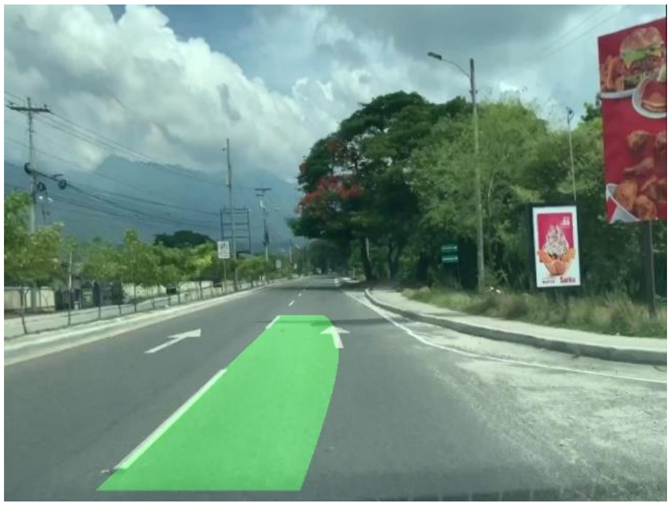


Fig. 12 Special Cases

In order to calculate the precision, equation (3) is used

$$Precision = \frac{TP}{TP + FP} \quad (3)$$

Where TP is the sum of true positives, and FP the sum of false positive. On Table IV. Is shown the precision for each video.

TABLE IV
DATASET CONDITIONS

Precision %					
Video 1	***Video 2***	***Video 3***	***Video 4***	***Video 5***	***Video 6***
96	96	49	89	56	68

By taking the mean of both ideal and nonideal conditions, results on the global precision of the algorithm.

TABLE V
PRECISION MEAN

Precision mean %		
Ideal	***Nonideal***	***Global***
94	58	76

Using the python time package, the execution time for a dataset of 300 frames equivalent to 10 seconds of video recorded at 30 frames per second, was obtained for the lane detection algorithm, being on average 19.268 frames per second or 15.70 seconds in total, this result was obtained by using CPU multithreading. As reading and writing frames is I/O bound, multithreading is ideal for these type of situations.

V. LIMITATIONS

By using second degree polynomials, on extreme curves, the algorithm will not perform or detect the lanes, though by making the degree greater it would be at the cost of computational resources.
The classical computer vision techniques such as Canny detectors and HLS thresholding are sensible to brightness, lightning, contrasts, glare, etc. The studied dataset did not contain extreme lightning conditions.
The proposed approach solves lane detection and implements a neural network for object detection, but these processes are done sequentially, on real life situations this approach is inefficient, this can be solved by using parallel computing or by computing both processes concurrently using different dedicated hardware e.g., an FPGA for lane detection and a GPU for object detection.

VI. CONCLUSIONS

On this paper an algorithm based on computer vision techniques for lane detection and an implementation of YOLOv5, is applied on a custom dataset generated on San Pedro Sula, Honduras; obtaining a high precision under ideal conditions. By analyzing the algorithm in a 10 second video or 300 frame set, the execution time was 15.70 seconds. This shows that this algorithm based on traditional computer vision techniques can be implemented successfully under similar conditions. As object detection was implemented, in order to analyze results, it's necessary to train the model using a custom dataset containing objects of interests, this allows for future work and quantification of the behavior of the model in this context of autonomous vehicles, and advanced driving systems.

References

[1] Lucas Tabelini and Rodrigo Berriel and Thiago M. Paixao and Claudine Badue and Alberto F. De Souza and Thiago Oliveira-Santos "Poly-LaneNet: Lane Estimation via Deep Polynomial Regression". 2020, arXiv:2004.

[2] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A Survey of Autonomous Driving: Common Practices and Emerging Technologies," in IEEE Access, vol. 8, pp. 58443-58469, 2020, doi: 10.1109/ACCESS.2020.2983149.

[3] Global status report on road safety 2018. Geneva: World Health Organization; 2018. Licence: CC BY-NC-SA 3.0 IGO.

[4] M. S. Fuentes, N. A. L. Zelaya, and J. L. O. Avila, "Coffee Fruit Recognition Using Artificial Vision and neural NETWORKS," 2020 5th International Conference on Control and Robotics Engineering (ICCRE), 2020, pp. 224-228, doi: 10.1109/ICCRE49379.2020.9096441.

[5] Haque, Md & Islam, Md & Alam, Kazi & Iqbal, Hasib & Shaik,Md. (2019). A Computer Vision based Lane Detection Approach. International Journal of Image, Graphics and Signal Processing. 11. 27-34. 10.5815/ijigsp.2019.03.04.

[6] K. C. Bhupathi and H. Ferdowsi, "An Augmented Sliding Window Technique to Improve Detection of Curved Lanes in Autonomous Vehicles," 2020 IEEE International Conference on Electro Information Technology (EIT), 2020, pp. 522-527, doi: 10.1109/EIT48999.2020.9208278.

[7] Rohit Gandikota, ``Computer Vision for Autonomous Vehicles``. 2018, arXiv:1812.02542

[8] M. Gluhaković, M. Herceg, M. Popovic and J. Kovačević, "Vehicle Detection in the Autonomous Vehicle Environment for Potential Collision Warning," 2020 Zooming Innovation in Consumer Technologies Conference (ZINC), 2020, pp. 178-183, doi: 10.1109/ZINC50678.2020.9161791.