International Journal of Technology and Emerging Sciences (IJTES)

www.mapscipub.com

Volume 02 || Issue 03 || July 2022 || pp. 05-09 E-ISSN: 2583-1925

Wrong Side Driving Detection

Sriram Narayana Cummaragunta [1], Srinandan K S [2], Jyoti Shetty [3]

[1,2,3]*Computer Science Engineering, RV College of Engineering, Bangalore, India*

Abstract - *Road accidents are one of the leading causes of death. Many accidents are caused due to wrong side driving. The motive of this project is to automate the difficult task of enforcing the existing traffic laws automatically. It involves continuous and real-time monitoring of traffic on a street or at an intersection. To enforce traffic laws, there are various challenges like illumination, occlusion, poor quality of CCTV footage, changing weather conditions, etc. The paper proposes a wrong side driving detection model based on Computer Vision. For object-detection, the YOLOv4 algorithm is used. The tracking of the moving objects is done using centroid tracking. Finally, the approach uses ALPR to capture the license plates of violating vehicles. The results are stored in the Firebase database for further processing and the output is obtained to an accuracy of more than 95%.*

Key Words—YOLOv4, Object Detection, Centroid Tracking, ALPR, Firebase

1. INTRODUCTION

Deep Neural Networks" refer to Artificial Neural Networks (ANN) having multiple layers. Due to its ability to handle a huge amount of data, it has become widely used by Data Scientists for all kinds of applications. Convolutional Neural Network (CNN) is an example of an ANN that has become popular in recent years. It derives its name from mathematical linear operations between matrices referred to as convolutions. [1] CNN has many layers such as the convolutional layer, the nonlinearity layer, the pooling layer and the fully-connected layer. In comparison with standard feedforward neural networks having similar sized layers, CNNs have significantly fewer connections and parameters. [2] Due to this, they are much faster and easier to train. CNN is used in a variety of applications such as image classification data sets (eg- Image Net [2]), computer vision, anomaly detection, NLP, etc.

YOLO is brief for You Only Look Once. It's a period of time seeing systems which may acknowledge multiple objects in every frame. YOLO acknowledges objects with higher precision and speed compared to alternative object detection algorithms.[3] YOLO is predicated on Convolutional Neural Networks (CNN). A CNN divides a picture into many different regions. The YOLO model then predicts the various bounding-boxes along with their possibilities for every region. YOLO views the whole image throughout coaching.

YOLO was invented by Joseph Redmon [3]. YOLO v4 was invented by 3 developers Alexey Bochkovskiy, Hong-Yuan and Chien-Yao [4]. YOLOv4's design consists of CSPDarknet53 as a backbone, abstraction pyramid pooling extra module, PANet path-aggregation neck and YOLOv3 head. [4] CSPDarknet53 may be a novel backbone that may enhance the training capability of CNN. The abstraction pyramid pooling block is side over CSPDarknet53 to extend the receptive field and strain the foremost important context options. Rather than Feature pyramid networks (FPN) for object detection employed in YOLOv3, the PANet is employed because of the technique for parameter aggregation for various detector levels. [4] YOLOv4 is double as quick as EfficientDet (competitive recognition model) with comparable performance. Additionally, AP (Average Precision) and Federal Protective Service (Frames Per Second) are magnified by a 100% and twelve-tone system compared to YOLOv3 [4]. Due to these advantages of YOLOv4 over YOLOv3, this project proposes the use of YOLOv4 for the detection of vehicles.

2. RELATED WORK

There are many related works done in this field, for varying requirements. Saidasul et al. [5] developed a real-time intelligent transportation system (ITS) to detect vehicles going on the wrong side of the road using the YOLOv3 model. Our work has been inspired by this paper, and the objective is to upgrade the algorithm to YOLOv4, while also using some of our own techniques to reduce computational costs.

Qin Zou et al. [6] propose a model that uses a hybrid deep neural-network. This model is used for lane detection. The system uses continuous frames extracted from a piece of footage taken from a vehicle driving on the road. This model combines DRNN and DCNN models. Later uses a LSTM model for mapping. The results obtained demonstrate the benefits of ConvLSTM compared to FcLSTM w.r.t sequential feature-learning as well as target information prediction with respect to lane detection. Area for research and improvement under the usage of SegNetConv network rather than UNet-Conv network is stated.

Gonçalo Monteiro et al. [7], proposed a system automatically detects drivers travelling in the wrong direction. This sets off a pre configured alarm on various highway-traffic related telematic systems. It also tracks vehicles against crowded events as well as occlusions. The model used is Optical Flow with Gaussian Mixture model and filtering. However, the authors stated that there is a possibility that a vector/vectors of flow are detected despite the absence of real motion. This is caused by unexpected vibrations or movement of the camera-pole. This could also be caused by noisy or faulty motion-flow estimation.

Zillur Rahman et al. [8] propose a system that can identify vehicles travelling in the wrong direction on a road. This system marks/demarcates them from the on road CCTV footage. YOLO object detector is used since it's highly accurate. It is also quicker compared to any other object detection algorithm. To verify their system, they captured three videos from the roads in

Chittagong, Bangladesh. The resolution of the CCTV footage was 1280 x 720 pixels. Every wrong-side car from the three videos were successfully identified and penalised. Thus, accuracy of their system, in practical scenarios, was almost 100%. One of the limitations of their system was the centroid-tracking technique that was used. The object centroids calculated from the bounding boxes must be in close proximity between neighbouring frames. Otherwise the ID number might be interchanged due to overlapping vehicles (objects).

Junli Tao et al. [9] address multi-lane roads. It uses CCTV footage from a camera to identify the currently occupied lane of a vehicle. GPS information is utilised to determine the constraints that should be enforced w.r.t directions travelled for a certain road taken into consideration. The authors adopted a multi-lane detecting monocular camera. It then analyzes the identified lanes in conjunction with GPS data in order to locate the vehicle more precisely at lane level. The system that was developed was fairly accurate for the identification of wrong side driving provided that the road marking was reasonably visible and were not occluded due to other cars present on the road or in the same lane. This system can be tweaked to take care of GPS temporal-occlusion situations. This is because the GPS data is not required for each and every frame.

A. Sentas et al. [10] address the issue of cars using emergency lanes designated for emergency vehicles like ambulances, etc. If any vehicle travels in the wrong lane, a violation is recorded by their system. Violation detection processes were carried out on the basis of various real-time image-processing algorithms. Unlike many papers, background-subtraction is not utilised in vehicle-detection algorithms. To detect a lane violation, two points are taken from the lane and an imaginary line is drawn by the program indicating the emergency lane. This is done by calculating slope and intercept and using elementary geometric techniques. Their program is resistant to conditions such as moving cameras, changing weather conditions, etc. In future the authors plan to create a model in which the user can choose what type of violations to monitor.

Y. Xing et al. [11] undertook a comprehensive analysis of various lane detection algorithms. They considered three aspects namely lane-detection techniques, integration, as well as evaluation-methods. Due to limitations inherent in camera based lane identification models, techniques to develop more robust and precise lane detection systems are pondered upon. The authors also propose a new, state of the art "computational experiment based parallel lane detection framework".

V. Nguyen et al. [12] propose an algorithm to yield information related to the lane as well as the vehicle which can be used by the proposed driver-assistance system, referred to by the authors as a "lane change assistant system (LCAS)". Many papers in the past could only detect the vehicles or the lanes separately and independently of each other. The authors, however, assert that the combination of the lane information as well as the vehicle information can be used to aid the LCAS and improve the accuracy and reliability of the proposed system. An LCAS should be able to identify frontal lanes and also discover the vehicles present around any test vehicle. A computer vision based system is proposed consisting of three cameras, two of which are under the wing mirrors, and one on the test vehicle's windscreen. The cameras' video is processed and is utilised to identify three lanes, and also detect the vehicles around it. After this, the Kalman filter is utilised to track and monitor the vehicles that are detected. And finally, the relative speed between the detected vehicle and the test vehicle is computed. Each frame takes roughly 43 ms to process. This system was tested on various Korean highways.

J. C. Nascimento et al. [13] address the issue of tracking and monitoring moving objects by using deformable models. The authors propose a new Kalman-based technique. Abrantes and Marques, in 1996, proposed a category of constrained clustering techniques in the domain of static shape estimation, which inspired the authors' work. Centroids of the moving objects are tracked and monitored by using inter-frame as well as intra-frame recursions. Centroids are calculated by computing the weighted-sums of the edge-points that correspond to the moving object's bounding box. The authors use various competitive-learning techniques inside the algorithm used for tracking objects. This leads to improved robustness w.r.t. contour sliding as well as occlusion.

J. Jin et al. [14] address the design as well as the implementation of real-time multi-object centroid-tracking for the purpose of gesture recognition. It comprises 4 stages namely "preprocessing, local intensity accumulation, object observation, and particle filter"[14]. They discuss 2 major aspects, which are the trajectory accuracy of moving objects as well as real time processing. With the help of many real world experiments, the performance of the model was evaluated and their processing speed and efficiency were compared to the algorithm itself based on its software simulations. Although their work was focussed on gesture recognition, the very same centroid tracking concepts used in [13] and [14] were taken into consideration for the design of this project's centroid tracking algorithm.

3. METHODOLOGY

This section discusses the methodology of the proposed work.

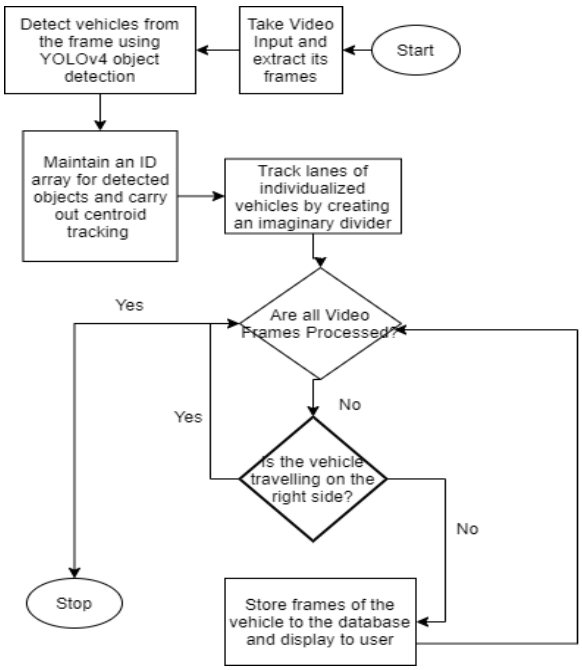

Figure 1. Flowchart representing the project

In Figure 1, one can see the basic architecture of the project. The video input is taken, frames are extracted from them, then the

frames are processed on the YOLO model. The model uses object detection to detect vehicles within each frame and form centroids for the objects detected. An imaginary divider is created from the input video file. This divider is necessary as each movement of centroids needs to be categorised using lanes. After all this processing, the output video is returned.

A. Algorithm

1. The program starts by sending an input video frame to the YOLOv4 model that was trained using the OIv6 dataset [15] on a separate Colab notebook as observed in our Git repository. One can download the obtained weights and use that in this system. This is handy to separate the training and prediction processes, since the end user is most likely to be a layperson who wants to use the system for his/her requirements and is only interested with the prediction portion of this project.
2. The program returns us bounding boxes of vehicles, once the frame is processed by the model.
3. To find out the direction of a tracked vehicle, the program computes the difference between a range of frames. It does this by first computing the object centroid from the bounding boxes using elementary geometric techniques. Suppose frame F_N has a centroid (x_N, y_N) then the program computes the pixel difference for varying values of N, during the configuration step as explained below. This step of the algorithm was inspired from [5].
4. In the first few "configuration" seconds, the direction of all vehicles is tracked. The movement of a centroid assigned an ID is tracked using the Euclidean distance algorithm and is used in the next step.
5. To determine whether the car is traveling on the right or wrong side of the road, the program creates an imaginary divider or median in the road based on the average extreme positions of vehicles moving in respective directions. Due to the jittery nature of the initial object detection algorithm, the bounding boxes' mini oscillations mean that a slope cannot be calculated easily and therefore the median calculated is assumed to be vertical. In left side driving nations, whatever vehicles moving upwards and to the right of the median are considered as violations.
6. The vehicles moving in the wrong direction are considered as violations and these violations are uploaded to the Firebase database.
7. Finally, using ALPR, the violator's license plate is found and matched with the police database and a challan is generated with the photo evidence.
8. The following dataset is used by the proposed work:

   - Object detection image dataset for training : OIv6 [15]

   Other CCTV camera videos found on YouTube are used to run validation of wrong side detection.

B. Implementation Details

Language used in the entire project is Python (3.7+). Tools used in this project include:

- Darknet : It is an open-source neural network framework. It is simple to setup and install, quick and supports GPU as well as CPU computation.
- Google Colab : This is a product from Google Research that provides a Virtual Environment for the execution of Python Code on a reasonably fast GPU to carry out ML-related tasks anywhere on the internet.
- Anvil : This is used to convert a Colab's model into a web application. This was used as the front-end of the project.
- Firebase Firestore : The cloud database used to store and send the detections to the front-end of the application.

Anvil and Colab have been used in order to use a moderately powerful GPU along with a functional frontend. This project can be implemented without these tools on a powerful machine, and the above is done for demonstration purposes only.

Although the model does not follow any specific data preprocessing steps, the dataset being used is provided by Open Images v6 (via Google), which provides annotated datasets to the specific classes needed. In this model, a "Cars" dataset was used, which was annotated by OIv6 for object detection. Collecting the dataset from Google's Open Images Dataset and then using the OIDv4 toolkit to generate the required labels is simple, efficient and elementary. The label contains the annotation labels. The labels that the toolkit gives us aren't in the prescribed YOLOv4 format. Using various helper programs, the labels were successfully converted to the required format. One has to change the label filename to utilise it along with the algorithm during the training process. [15]

The technique chosen to work with on this model is a Validation data split. To avoid re-substitution errors, the data was split into 2 different classes, namely a training dataset as well as a testing dataset. The model is built on a 70/30 split. The technique used here is referred to as the "hold-out validation technique". There may be a likelihood that a non-uniform distribution of distinct categories of data is found in training and validation dataset. To rectify this issue, the training dataset and the test dataset are created with equally distributed classes of data (referred to as stratification).

4. RESULTS AND ANALYSIS

For testing purposes, the model was verified with the help of an Anvil Frontend and a Colab backend as mentioned earlier. The input video was fed as a YouTube video link. Vehicles travelling in the wrong direction on the road were edited into the video. On clicking the submit button on the frontend, an output was displayed on the screen. Violating vehicles' pictures were also stored on the Firebase database for proof. Some of the output obtained is shown below.

Figure 2. Project running on an Anvil Frontend

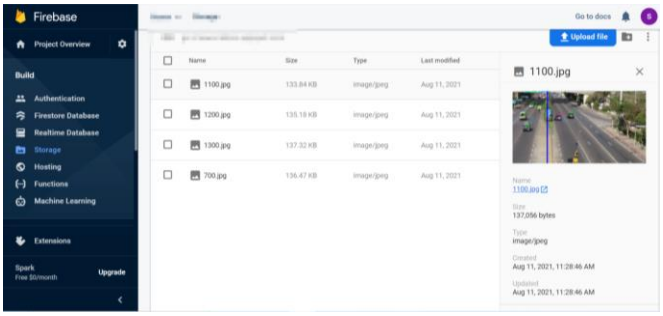

Figure 3. Picture of the violation stored in Firebase

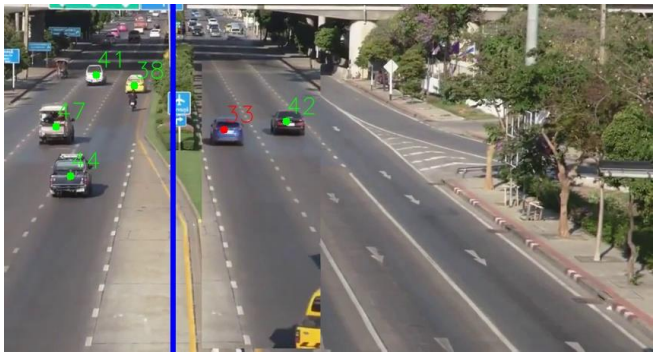

Figure 4. One of the violations stored in Firebase (red colored centroid indicates a violation and the blue line is the imaginary median automatically constructed by the program)

Figure 5. Another violation stored in Firebase

In Figure 4 and 5, it should be noted that the car whose centroid id is 42 took time to turn red from green. This is to prevent the system from penalising parked or slow moving cars.

The model trained resulted in an astonishingly high accuracy, more than 95% for the validation input provided, exceeding many expectations. However, the project still returns rare cases of false positives during certain traffic jams. Overall, one can see that the model is performing excellently and can be used with a front end.

A front end was later on added for demonstration purposes. Anvil was used for this to convert the Colab notebook to a working web-app by passing data through firebase storage. A manual video was considered for testing and a Confusion Matrix was created.

TABLE 1. CONFUSION MATRIX FOR WRONG SIDE DRIVING DETECTION

Total = 250	**Predicted Wrong Side Driving**	**Predicted Right Side Driving**
Actual Wrong Side Driving	31	1
Actual Right Side Driving	10	208

From the confusion matrix (Table 1), one can calculate attributes like Accuracy, Misclassification and Precision.

Accuracy = (TP+TN)/Total = (239)/250 = 95.6%

Misclassification= (FP+FN)/Total = 11/250 = 4.4%

Precision = TP/(TP+FP) = 40/41 = 97.5%

The model's learning curve was plotted which was fit perfectly to not overfit over the training data. The warning instructions for the training process are given by the official Darknet Git repo. [4]

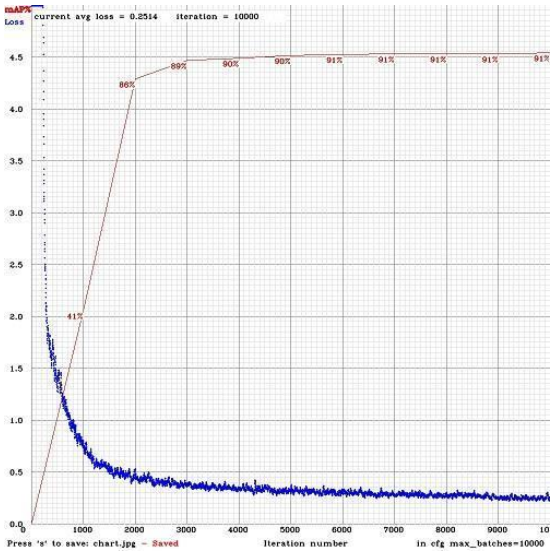

Figure 6. Learning Curve of the YOLOv4 model during the training process

Figure 2 shows the learning curve of a model which after around 2000 iterations straightens on the curve indicating its not learning at a pace worth training. This plots to an accuracy of around 0.7 to 1. Hence when the model was trained, the training was stopped sooner without overdoing it.

4. LIMITATIONS

The project has met the expectations sought out, but it certainly has faced several limitations along the way:.

1. The sensitivity of the camera input required : The input clipping that the model takes in must be from a fixed and stable CCTV. This rules out Dashboard cameras, rotating surveillance cameras, etc. Or else, the model struggles to find the divider resulting in failure of lane separation.
2. Camera orientation: As discussed in the methodology section, the CCTV camera must be straight and parallel with respect to the road since the imaginary divider does not have any slope.
3. False Positives : The model sometimes results in false positives, due to blurred footage or due to improper detections. Although it isn't very often, it does occur once in a while.
4. Speed of response from Anvil : As the project requirements don't mention a custom made front-end, Anvil was utilised to use the Colab notebook on a Webapp. And as expected, the response speed isn't as desired. However, the project's core functionality is as fast and lightweight as expected.

5. CONCLUSION AND FUTURE SCOPE

The solution proposed is completely automated and doesn't need any manual intervention. The current system is rusty as it requires a traffic police or a certain person to keep tabs on the traffic through the footage, which makes it inefficient, not so accurate, and rather time consuming. The model brings in a detection accuracy of around 95% and a real time viable lane detection accuracy.

The model has accomplished the goals sought out with great accuracy. Like most other projects, there is room for future enhancements as well. A few ideas can be explored:

- Making a custom Front-end for the project.
- Improving efficiency of the model by better training.
- Customizing the simplistic centroid tracking algorithm (which currently uses Euclidean distances).

Considering all these future enhancements left and constraints faced while implementation, this doesn't mean the goals sought out were met to the finest extent possible. We are keen on working on this project to the next level in the future by continuously fine-tuning the code.

Our code can be found at the following link:
https://github.com/sriramcu/yolov4_wrong_side_driving_detection

REFERENCES

[1] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1–6.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Commun. ACM, vol. 60, no. 6, pp. 84–90, 2017.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[4] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," arXiv [cs.CV], 2020.

[5] S. Usmankhujaev, S. Baydadaev, and K. J. Woo, "Real-time, deep learning based wrong direction detection," Appl. Sci. (Basel), vol. 10, no. 7, p. 2453, 2020.

[6] Q. Zou, H. Jiang, Q. Dai, Y. Yue, L. Chen, and Q. Wang, "Robust Lane detection from continuous driving scenes using deep neural networks," arXiv [cs.CV], 2019.

[7] G. Monteiro, M. Ribeiro, J. Marcos, and J. Batista, "Wrongway drivers detection based on optical flow," in 2007 IEEE International Conference on Image Processing, 2007, vol. 5, pp. V-141-V–144.

[8] Z. Rahman, A. M. Ami, and M. A. Ullah, "A real-time wrong-way vehicle detection based on YOLO and centroid tracking," in 2020 IEEE Region 10 Symposium (TENSYMP), 2020, pp. 916–920.

[9] Tao, J., Shin, B.-S., & Klette, R. (2013). Wrong roadway detection for multi-lane roads. In Computer Analysis of Images and Patterns (pp. 50–58). Berlin, Heidelberg: Springer Berlin Heidelberg.

[10] A. Sentas, S. Kul, and A. Sayar, "Real-time traffic rules infringing determination over the video stream: Wrong way and clearway violation detection," in 2019 International Artificial Intelligence and Data Processing Symposium (IDAP), 2019.

[11] Y. Xing et al., "Advances in vision-based Lane detection: Algorithms, integration, assessment, and perspectives on ACP-based parallel vision," IEEE/CAA j. autom. sin., vol. 5, no. 3, pp. 645–661, 2018.

[12] V. Nguyen, H. Kim, S. Jun, and K. Boo, "A study on real-time detection method of Lane and vehicle for Lane change assistant system using vision system on highway," Eng. Sci. Technol. Int. J., 2018

[13] J. C. Nascimento, A. J. Abrantes, and J. S. Marques, "An algorithm for centroid-based tracking of moving objects," in 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No.99CH36258), 1999, vol. 6, pp. 3305–3308 vol.6.

[14] J. Jin, S. Lee, B. Jeon, T. T. Nguyen, and J. W. Jeon, "Real-time multiple object centroid tracking for gesture recognition based on FPGA," in Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication - ICUIMC '13, 2013.

[15] "Open Images V6," Googleapis.com. [Online]. https://storage.googleapis.com/openimages/web/index.html. [Accessed: 26-Aug-2021].