Vehicle Speed Prediction Using YOLOv4 and XGBoost Regression

By

Yafet Jaya Kusumo
Institut Sains dan Teknologi Terpadu Surabaya
Email: mailto:yafetkusumo@gmail.com,

Evan Kusuma Susantoz
Institut Sains dan Teknologi Terpadu Surabaya
Email: mailto:evanks@stts.edu,

Yosi Kristian
Institut Sains dan Teknologi Terpadu Surabaya
Email: mailto:yosi@stts.edu

Abstract

There are many ways to detect vehicles' speed these days, which can be categorized into two approaches: a non-computer-vision-based and a computer-vision-based. In this paper, we propose a computer-vision-based approach using YOLOv4 and XGBoost Regression. To predict vehicles' speed efficiently, we use YOLOv4 for vehicle detection and XGBoost regression for speed prediction. In order to get the best speed prediction, we build our dataset by recording the local traffic and measuring their speed using a speed gun. From those traffic videos, we detect vehicles by using YOLOv4 to generate its bounding boxes. From the bounding boxes, we can extract its coordinates relative to the screen, the distance and the angle between the two centroids, and the time it takes from point A to point B. This information will be our features, and the speed from the speed gun will serve as the target to train our XGBoost regression model. In this paper, we conduct several experiments using various features to get the best model. Our experiments conclude that our speed prediction approach using YOLOv4 and XGBoost regression has a very high performance regarding to the ground truth with an MAE of just 2 km/h.

Index Terms – Automatic Vehicle Speed Detection, Computer Vision, Machine Learning, YOLO.

I. Introduction

There are already many approaches to measure a vehicle's speed, either using computer-vision-based approaches or more traditional approaches using devices, such as inductive loop detectors, radars detectors, and so on [1]–[3]. In this paper, we focused on the computer-vision-based approaches because it is simpler to deploy and more cost-efficient [1]–[6]. The only challenge that computer-vision-based approaches have is providing reliable and accurate results, as accurate as traditional approaches. In this paper, we used speed gun results as the ground truth following [4]–[6].

However, in the computer-vision-based approach, earlier researchers need to calibrate the camera first, either with the help of manual work or automatically done. For example, He and Yung [7] use a vehicle's speed estimation based on a calibration pattern formed by lane markings on the road [8]. The authors use a rectified image in further processing to deal with perspective projection. To obtain the locations of the vehicles within the ground plane, shadows cast by rear bumpers are used. The vehicles and shadows are detected by background subtraction and binary block matching.

Cathey and Dailey [9] use a method based on detection of the vanishing point which is in the direction of vehicles movement. To obtain this vanishing point, detected line markings are used and their intersection in the least square manner. The scale (pixels/meters ratio) for the camera is computed from average line marking stripe length and known stripe length in the real world. Finally, the authors used cross correlation to compute the number of pixels which vehicles passed between consecutive frames.

Grammatikopoulos et al. [10] use the assumption that the camera is only tilted along the x-axis; thus, they assume that the second vanishing point (horizontal and perpendicular to the first one) is in infinity. The first vanishing point is detected as the intersection of the line markings with least squares adjustment. The vehicles are detected by background subtraction and tracked by normalized cross-correlation.

You et al. [11] use detection of the vanishing point in the direction of vehicles' movements from lane markings and vanishing point perpendicular to road plane from detected poles and pedestrians. The authors obtain the scale from a known height of the camera above the road or known dimensions on the road.

The following researchers calibrate the camera automatically by using the vehicle's movement. Dubska et al. [5] published a speed measurement system using a calibration method by detection of two vanishing points [6]. Similarly to Dusbka et al. [5], [6], Sochor et al. [4] use the detection of two vanishing points to calibrate the camera, infer the scene scale by using 3D models bounding box of frequently passing cars and aligning it with the real observe cars, and measure the speed of passing cars by detecting the vehicles using Faster-RCNN [12] and tracking the vehicles by a combination of background subtraction and Kalman filter [13] assisted by the detector. In simply, they call the method as Edgelets + BBScale + Reg.

Schoepflin et al. [14] use an activity map (by detecting the vehicles as the moving foreground) to obtain lane boundaries and the intersection of the boundaries treat as the first vanishing point in the direction of the vehicle motion. The second vanishing point is detected as the intersection of lines formed by the bottom edges of the vehicles. One known length (manually measured and entered per camera) in the image is used for scale inference.

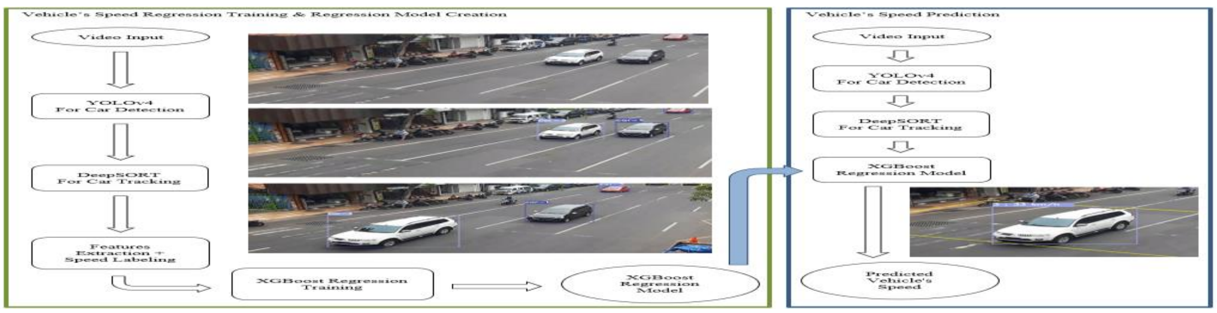
Vehicle's Speed Regression Training & Regression Model Creation
Video Input
YOLOv4 For Car Detection
DeepSORT For Car Tracking
Features Extraction + Speed Labeling
XGBoost Regression Training
XGBoost Regression Model
Vehicle's Speed Prediction
Video Input
YOLOv4 For Car Detection
DeepSORT For Car Tracking
XGBoost Regression Model
Predicted Vehicle's Speed

Fig. 1: *The workflow of our proposed work. It contains two parts: the first part is to train the machine and to create the regression model, and the second part is to predict the vehicle's speed using the regression model. On the first part, the features extraction include: 1. The start and the end bounding boxes coordinates, 2. The distance between two centroid of the start and the end bounding boxes, and 3. The angle between two centroid of the start and the end bounding boxes.*

Filipiak et al. [15] use sequences of detected license plates of vehicles for finding intrinsic and extrinsic camera parameters by evolutionary algorithm. The method was evaluated on a dataset captured by zoomed surveillance cameras with a small field of view on the road.

In this paper, we propose a new approach to predict the vehicles' speed by using a combination of YOLOv4 [16] and trained regression model via XGBoost Regression [17]. In this paper, our method only requires a comprehensive dataset which we extract to get several key features. We combine the features with the speed label from the ground truth. After that, we train the machine to learn from those parameters, and finally, predict the vehicles' speed.

The key contributions of this paper are:

1) We introduce a new way of detecting vehicles' speed without the need for camera calibration.
2) Our experiments show that the most influential feature that help to determine predicting the vehicle's speed more accurately is the distance between two centroids of each bounding box.
3) Our experiments show that our approach can predict vehicles' speed with a very small MAE of 2 km/h.

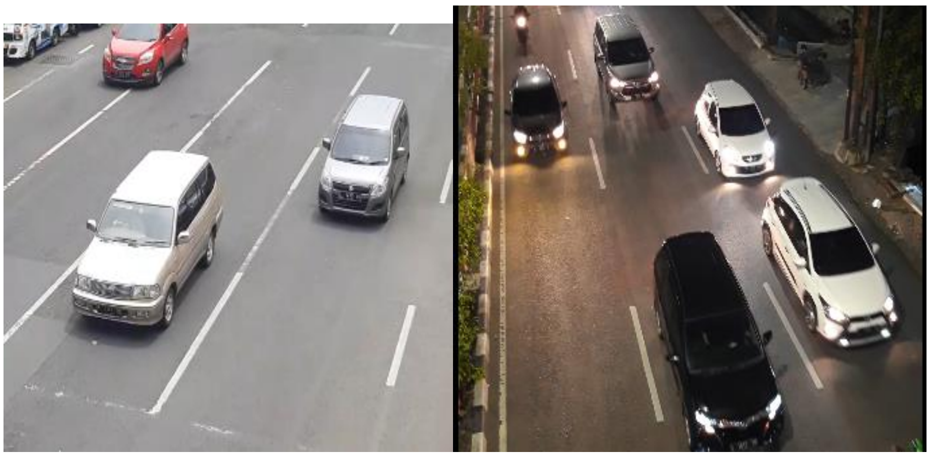

Fig. 2: *Recorded local traffic during daytime and nighttime in two different angles of the camera.*

Table I: Recorded local traffic dataset of cars with labeled ground truth speed.

No.	**# Cars w/ Ground Truth**	**Day / Night**	**Left / Right Angle**	**Resolution**	**Duration**
1	12	Day	Left	1920x1080	00:02:05
2	12	Day	Left	720x1280	00:01:06
3	18	Night	Left	1920x1080	00:00:50
4	3	Night	Left	720x1280	00:00:16
5	42	Night	Left	1920x1080	00:01:26
6	31	Night	Left	720x1280	00:01:23
7	31	Night	Left	1920x1080	00:01:27
8	19	Night	Left	720x1280	00:01:25
9	24	Night	Left	1920x1080	00:01:59
10	74	Day	Right	1920x1080	00:04:19
11	12	Day	Right	1920x1080	00:00:50
12	56	Day	Right	1920x1080	00:03:26
13	124	Day	Right	1920x1080	00:07:28
14	79	Day	Right	1920x1080	00:04:40
15	111	Day	Right	1920x1080	00:06:41
Total	648				00:39:21

II. Related Work Methods

The first task of detecting the vehicle's speed is to detect the vehicle itself. I have read the paper by Sochor et al. [4], and they used Faster-RCNN to detect the vehicle. Based on the paper by [12], Faster-RCNN is quite fast to detect an object which is about 200 ms per image. However, at the time of this research was being initiated, there was a faster and more accurate object detection method: YOLOv4. YOLOv4 was proposed by Bochkovskiy et al. [16], and based on the experiment, the result yields 33 FPS with the image size of 608 and AP of 43.5%, while Faster-RCNN yields 9.4 FPS and AP of 39.8%, on the same dataset MS COCO (test-dev 2017) and with the same Pascal GPU. Another research by Kim et al. [18] concluded that YOLOv4 also performed better than Faster-RCNN on the same dataset and GPU (GeForce RTX 2080Ti), with mAP of 98.19 and 93.40 and FPS of 82.1 and 36.32, respectively.

The second task is to find a good method to track a detected vehicle. This is a very important task because when predicting the speed, we want to give relevant parameters to the regression, for example, giving the coordinates of the vehicle moving from Point A and the coordinates of the vehicle moving to Point B to the regression, and those coordinates have to be from the same vehicle. Thus, the speed can be predicted properly for that vehicle. We found that DeepSORT has the lowest possible of ID Switches when tracking a detected object, and based on the experiment, it reduces the number of ID Switches by 45% from traditional SORT tracking method [19].

Lastly, we need to find a good regression method to yield a good model so that it can predict the vehicle's speed faster and more accurate. Chen et al. [17] proposed a regression method called XGBoost, which performed much faster than other major tree boosting systems, such as: pGBRT, Spark MLLib, H20, scikit-learn, and R GBM. When processing a classification case on Higgs-1M data, XGBoost runs 0.6841 sec per tree with the Test AUC score of 0.8304, which is better than scikit-learn which runs at 28.51 sec per tree and 0.8302 for the AUC score and R GBM which runs at 1.032 sec per tree and 0.6224 for the AUC score.

III. The Proposed Work

In this paper, we use YOLOv4 [16] for the vehicle detection and use DeepSORT [19] to track the vehicle's movement on the screen. We use the information retrieved from those two techniques and extract some features to train the machine using XGBoost Regression [17]. We also experiment on several features combinations, as the key input parameters, to train the machine to accurately predict the vehicle's speed. After we get the regression model, we use it to test and calculate the MAE of the predicted speed to the ground truth. Fig. 1 shows the workflow diagram of our proposed work.

A. Dataset

We gather our own dataset by recording a local city traffic in Surabaya, Indonesia. There are a total of 15 videos recorded in two different angles of the camera with a total duration of 39 minutes in 1920x1080 and 720x1280 resolutions which consist of 648 vehicles with the labeled ground truth speed, please refer to Table I. We also recorded the traffic during daytime and nighttime, please refer to Fig. 2.

On each passing vehicle that is set to be part of the ground truth dataset, we use a speed gun to clock the vehicle one time on a certain area only, please refer to Fig. 3. In other words, we don't track the speed of the vehicle throughout the appearance on the screen time from the beginning to the end. Therefore, on this experiment, we assume that the vehicle is moving at constant speed.

There is also a location of recording where it is close to a traffic light, so there is a condition where the vehicles are slowing down and stopping at the red light. For this condition, we create an augmented data for these vehicles to indicate there is no movement or little movement based on the threshold, please refer to Fig. 4. This will be explained more on section III.C Feature Extraction.

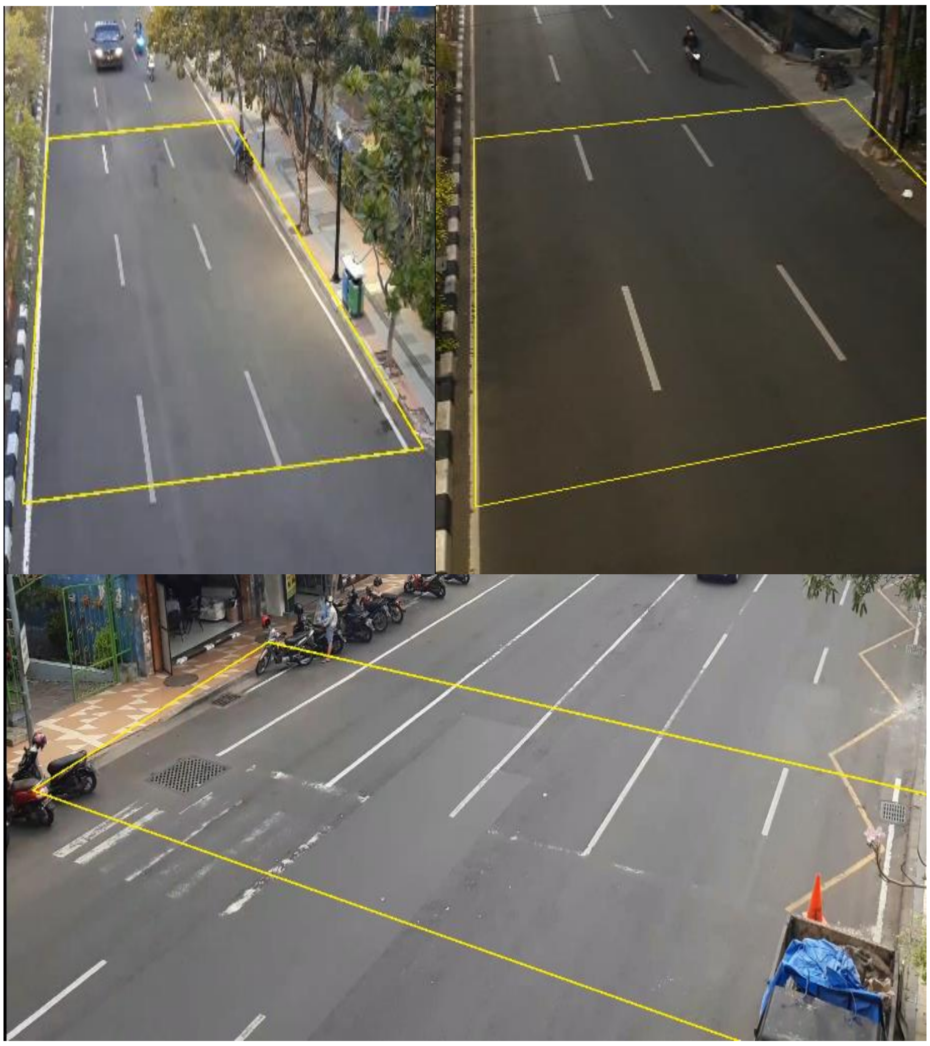

Fig. 3: *The yellow box indicates an area where we use the speed gun to clock the speed for each passing car.*

B. Vehicle Detection and Tracking

We use YOLOv4 [16] for the vehicle detection, due it is performance which is also reliable on the real-time detection. In this paper, the YOLOv4 detector has been pre-trained with MS COCO dataset to identify 80 classes. However, for this experiment, we modify the detector to only focus on the car class. So, the detector should only detect cars.

We fetch the recorded traffic videos to the YOLOv4 detector. The detector draws a bounding box on each detected car, so it gives us the information about the coordinates of each bounding box as seen on Fig. 5.

Once the cars are detected, we use DeepSORT [19] Tracker to track the movement of those cars. The tracker assigns an ID to each detected car as the car keeps moving along the road on the screen, as seen on Fig. 5.

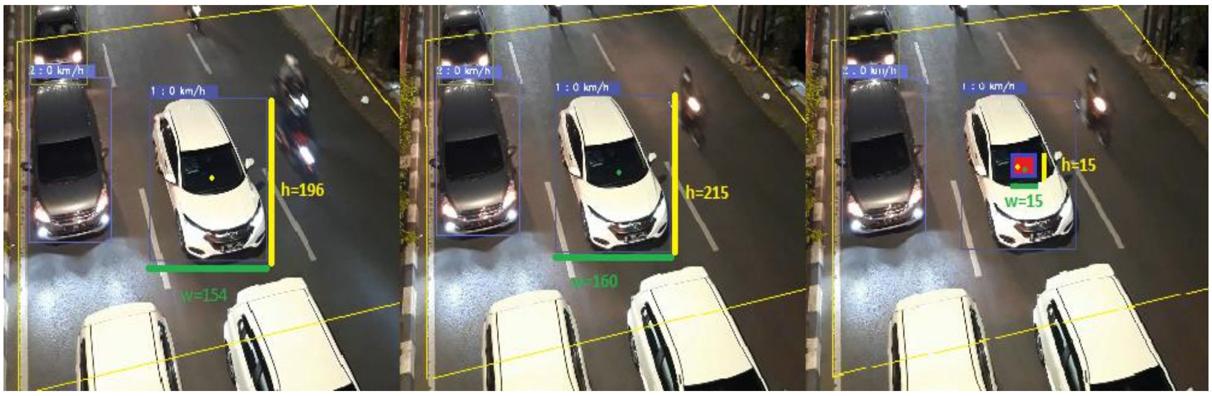

Fig. 4: *On certain occasion, although the car is not moving such as stopping at the red light, the bounding box drawn to the car may keep changing on its different frame. An augmented data is created to handle this scenario. On the left image at nth frame, we set the bounding box as the start bounding box with its coordinates, size and centroid (mark as yellow dot). On the middle image at (n+10)th frame, we set the bounding box as the end bounding box with its different coordinates, size and centroid (mark as green dot). We then define a threshold area of 15x15 pixels in the center of a bounding box (right image). As long as the start centroid and the end centroid are within the area of threshold, we define the car as not moving, and thus, we label the speed as 0 km/h.*

C. Feature Extraction

As part of the pre-processing, we extract some features from the information we have received from the vehicle detection and tracking. These features are going to be used as the key input parameters to train the machine later on.

First, we have received the coordinates of the bounding boxes and its centroids from the detection and tracking. For this feature extraction, we collect the start bounding boxes and the end bounding boxes of the same car's ID. We define the interval between the start and the end bounding box to be 10 frames or more apart. So, when a car is detected in the beginning, the bounding box coordinate will be used as the start bounding box feature, and 10 frames or more later, the same detected car's bounding box coordinates will be used as the end bounding box feature. We keep doing this repetitively at interval for each detected car. We only capture the start and the end bounding boxes coordinates as long as the coordinates are inside the speed captured area, which is the area when we did the speed clocking area for the ground truth, please refer to Fig. 6. We use these start and end bounding box coordinates as the first features.

Before we move on to the next feature extraction, we also create an augmented data to indicate that the car is not moving. We define that the car is not moving if between the start bounding box centroid and the end bounding box centroid, the distance is not more than 15 pixels away. If that's the case, we label the car's speed to be 0 km/h. This augmented data is to handle the situation when the car is slowing down and then finally stops at the red light, as seen in Fig. 4.

Next, for each captured start and end bounding box coordinates, we extract the distance from that information. The distance is calculated from the centroid of the start bounding box **(Xs,Ys)** to the centroid of the end bounding box (Xe, Ye), by using Euclidean distance (Fig. 7). We use the distance as the second feature.

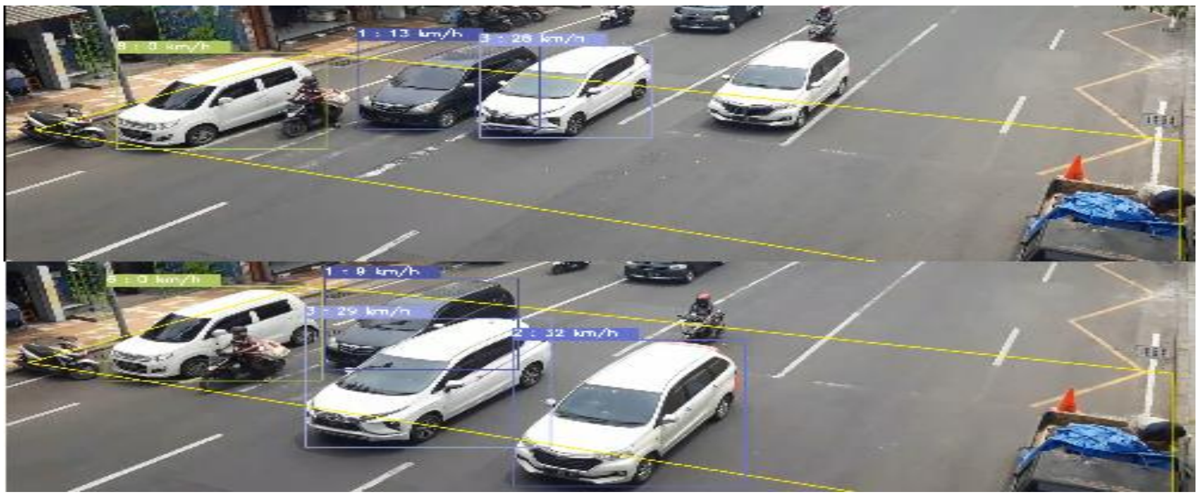

Fig. 5: *A car is detected using YOLOv4, and when entering the ground truth area and if the centroid of the car is inside the area, the start bounding box is drawn as seen on the top image. DeepSORT tracks the movement of the detected car as seen on the bottom image.*

After that, again for each capture start and end bounding box coordinates, we calculate the angle (in rad unit) of the movement between these two bounding boxes. The angle is calculated from the direction of the start bounding box centroid to the end bounding box centroid by using arctangent inverse trigonometric function (Please refer to Fig. 8). We use the angle as the last feature, and to make the machine be able to adapt to various camera's point of view.

Table II: *Feature combinations of regression training.*

Experiment No.	Start BB Feature	End BB Feature	Distance Feature	Angle Feature
1	Yes	Yes	Yes	Yes
2	Yes	Yes	No	No
3	No	No	Yes	Yes
4	Yes	Yes	Yes	No
5	Yes	Yes	No	Yes

D. XGBoost Training

After all of the pre-processing is done, we then ready to machine using XGBoost [17] regression. We experimented with the training by doing features combinations as shown in Table II.

After training the machine with each combination, we get the regression model that we can use to test for the vehicle's speed prediction. So, we fetch the recorded traffic videos again to YOLOv4 detector and DeepSORT tracker, and we add the regression model to predict the speed of the vehicle, please refer to diagram workflow in Fig. 1.

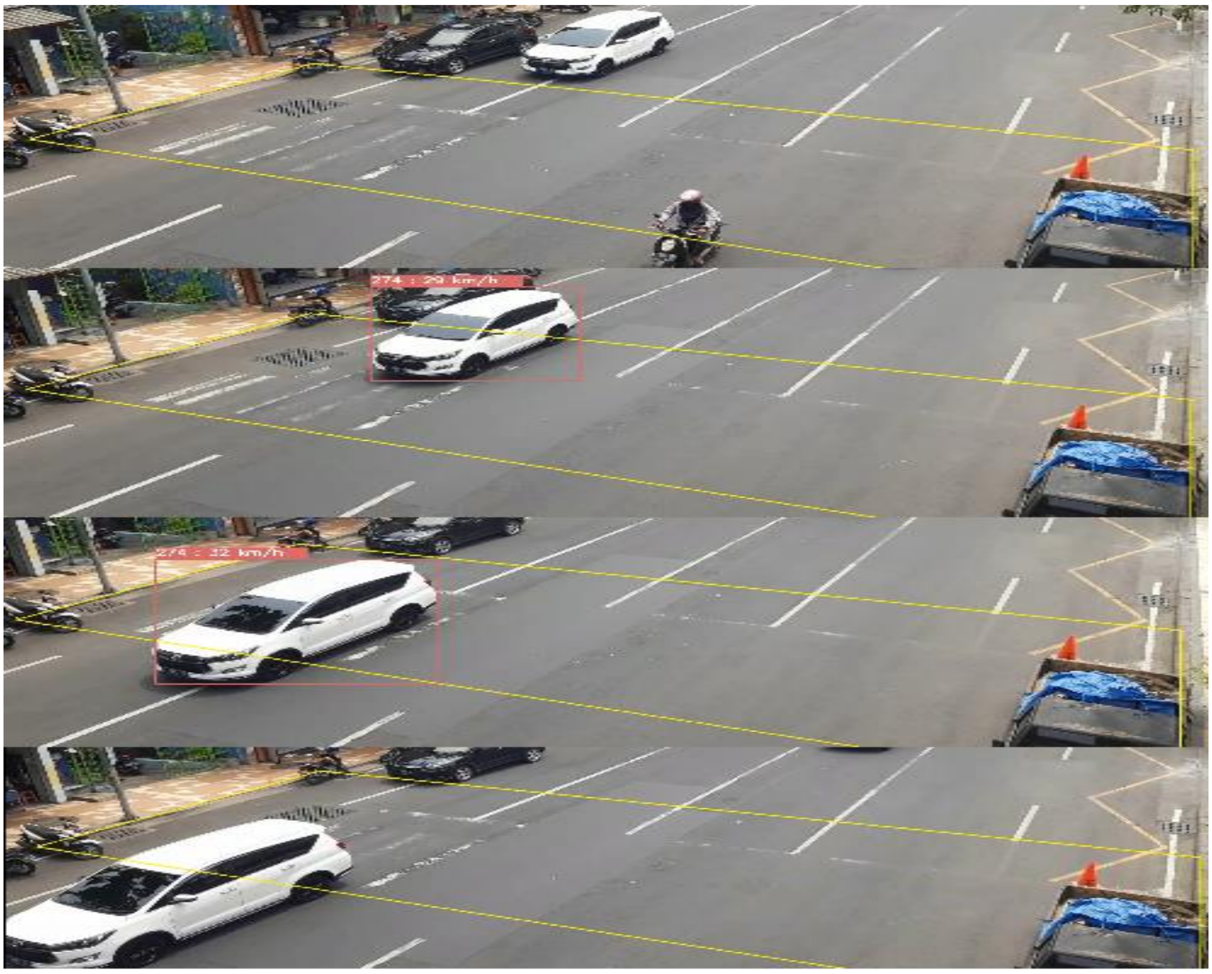

Fig. 6: *The bounding boxes is only drawn per frame when the centroid of the car is inside the ground truth area marked with yellow box.*

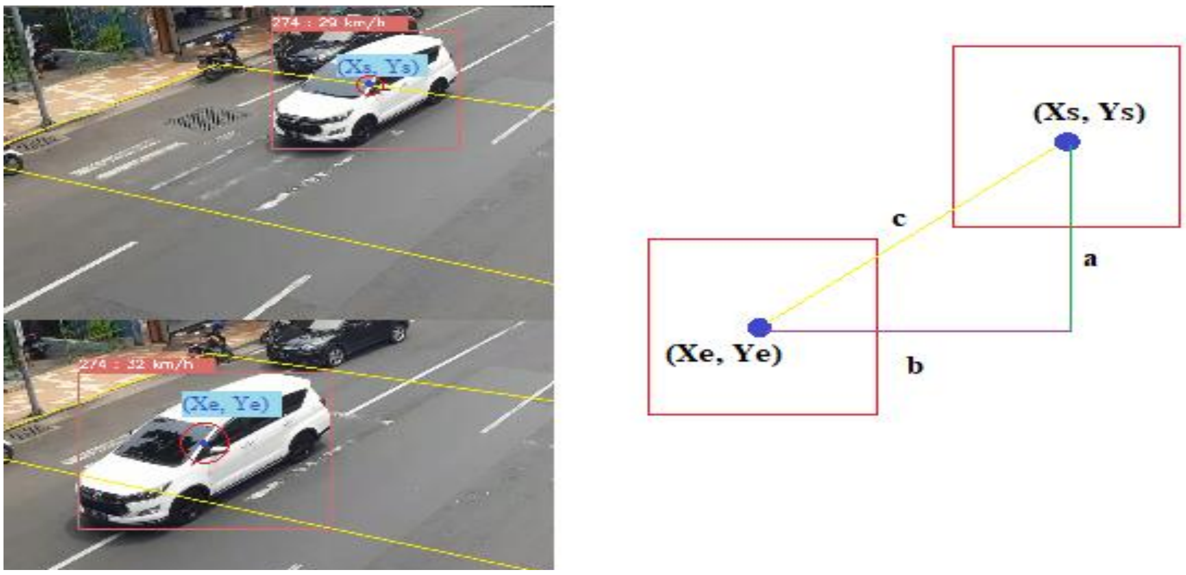

Fig. 7: *The distance feature is extracted by using the Euclidean distance to find* ***c****, as* ***a*** *and* ***b*** *are known from the start centroid coordinate which is* ***(Xs, Ys)*** *and the end centroid coordinate which is* ***(Xe, Ye)****.*

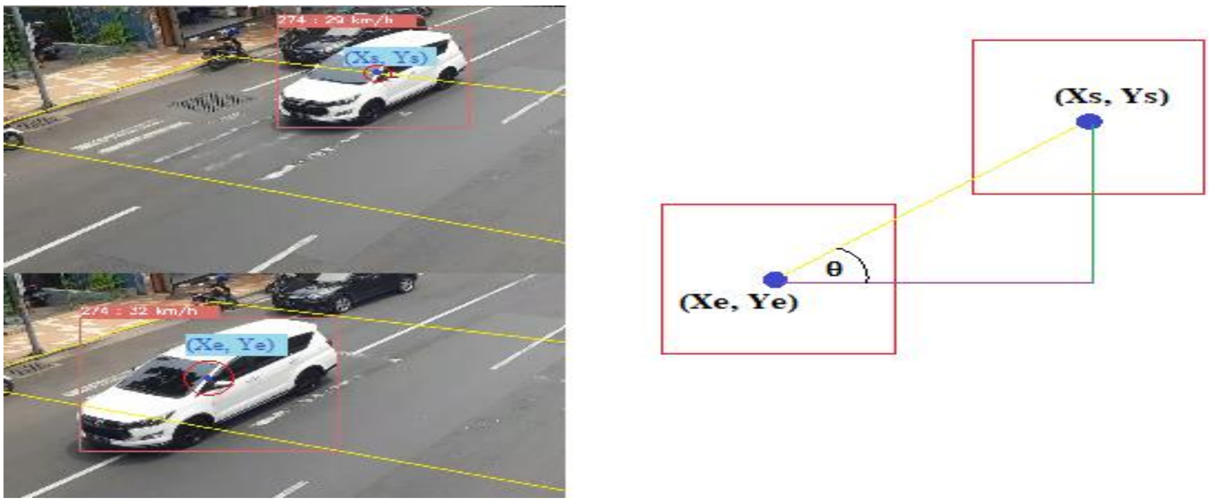

Fig. 8: *The angle feature is extracted by using arctangent function between the start centroid coordinate which is **(Xs, Ys)** and the end centroid coordinate which is **(Xe, Ye)**.*

Experiment & Result

In this paper, we perform five sets of experiment, please refer to Table II. From these experiments, we found that the model which has been trained by using all features yields the best result with MAE of 2.090 km/h. However, if we only use the start and the end bounding box coordinates as the features, the result yields the worst MAE of 3.271 km/h. All of the other experiment's result is shown in Table III.

Based on the results, the distance feature is the most influential feature in term of helping the prediction to be more accurate. The regression models which have been trained without the distance feature have bigger MAE.

IV. Conclusion

In this paper, we have described a new approach to detect the vehicle's speed without calibrating the camera by using machine learning and regression. We use a combination of YOLOv4 detector, DeepSORT tracker, and XGBoost regression. All of these methods are high performance methods [16], [17], [19] which can be run at real time and can be helpful in real-world.

Table III: *Experiments' MAE result.*

Regression Model	**MAE (km/h)**
All Features	**2.090**
Start and End BB + Distance Features	2.117
Angle + Distance Features	2.884
Start and End BB + Angle Features	3.029
Start and End BB Feature	3.271

Based on the experiment's result, the distance between two centroids of the start and the end bounding boxes is the most influential feature to help predicting the vehicle more accurately. However, we still need to combine it with all of the other features, such as the coordinates of the start and the end bounding boxes and the angle of the two centroids, to yield even more accurate result with MAE of 2.090 km/h.

Our suggestion for future research is to train the machine with more camera's point of view variations. We also can try to prepare more dataset which contains a dynamic vehicle's speed (not a constant speed).

References

[1] M. A. Adnan, N. Sulaiman, N. I. Zainuddin, and T. B. H. T. Besar, “Vehicle Speed Measurement Technique Using Various Speed Detection Instrumentation,” *IEEE Business Engineering and Industrial Applications Colloquium (BEIAC)*, pp. 668–672, 2013, doi: 10.1109/BEIAC.2013.6560214.

[2] W. Czajewski and M. Iwanowski, “Vision-Based Vehicle Speed Measurement Method,” in *Computer Vision and Graphics*, 2010, pp. 308–315.

[3] N. Kassem, A. E. Kosba, and M. Youssef, “RF-Based Vehicle Detection and Speed Estimation,” in *2012 IEEE 75th Vehicular Technology Conference (VTC Spring)*, May 2012, pp. 1–5. doi: 10.1109/VETECS.2012.6240184.

[4] J. Sochor, R. Juránek, and A. Herout, “Traffic Surveillance Camera Calibration by 3D Model Bounding Box Alignment for Accurate Vehicle Speed Measurement,” *Computer Vision and Image Understanding*, vol. 161, pp. 87–98, Feb. 2017, doi: 10.1016/j.cviu.2017.05.015.

[5] M. Dubska, A. Herout, and J. Sochor, “Automatic Camera Calibration for Traffic Understanding,” in *Proceedings of the British Machine Vision Conference 2014*, 2014, pp. 42.1-42.12. doi: 10.5244/C.28.42.

[6] M. Dubska, A. Herout, R. Juranek, and J. Sochor, “Fully Automatic Roadside Camera Calibration for Traffic Surveillance,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 3, pp. 1162–1171, Jun. 2015, doi: 10.1109/TITS.2014.2352854.

[7] X. C. He and N. C. Yung, “A Novel Algorithm for Estimating Vehicle Speed from Two Consecutive Images,” in *2007 IEEE Workshop on Applications of Computer Vision (WAC V '07)*, Feb. 2007, pp. 12–12. doi: 10.1109/WACV.2007.7.

[8] X. C. He and N. C. Yung, “New method for overcoming ill-conditioning in vanishing-point-based camera calibration,” *Optical Engineering*, vol. 46, no. 3, p. 037202, Mar. 2007, doi: 10.1117/1.2714991.

[9] F. W. Cathey and D. J. Dailey, “A novel technique to dynamically measure vehicle speed using uncalibrated roadway cameras,” in *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.*, Jun. 2005, pp. 777–782. doi: 10.1109/IVS.2005.1505199.

[10] L. Grammatikopoulos, G. Karras, and E. Petsa, “Automatic Estimation Of Vehicle Speed From Uncalibrated Video Sequences,” *Proceedings of International Symposium on Modern Technologies, Education and Profeesional Practice in Geodesy and Related Fields*, pp. 332–338, 2005.

[11] X. You and Y. Zheng, “An accurate and practical calibration method for roadside camera using two vanishing points,” *Neurocomputing*, vol. 204, pp. 222–230, Sep. 2016, doi: 10.1016/j.neucom.2015.09.132.

[12] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017, doi: 10.1109/TPAMI.2016.2577031.

[13] R. E. Kalman, “A New Approach to Linear Filtering and Prediction Problems,” *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, Mar. 1960, doi:

10.1115/1.3662552.

[14] T. N. Schoepflin and D. J. Dailey, “Dynamic Camera Calibration of Roadside Traffic Management Cameras for Vehicle Speed Estimation,” in *Proceedings. The IEEE 5th International Conference on Intelligent Transportation Systems*, 2002, vol. 2002-Janua, no. 2, pp. 25–30. doi: 10.1109/ITSC.2002.1041183.

[15] P. Filipiak, B. Golenko, and C. Dolega, “NSGA-II Based Auto-Calibration of Automatic Number Plate Recognition Camera for Vehicle Speed Measurement,” in *Applications of Evolutionary Computation*, 2016, pp. 803–818.

[16] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection,” Apr. 2020, [Online]. Available: http://arxiv.org/abs/2004.10934

[17] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2016, vol. 13-17-August-2016, pp. 785–794. doi: 10.1145/2939672.2939785.

[18] J. A. Kim, J. Y. Sung, and S. H. Park, “Comparison of Faster-RCNN, YOLO, and SSD for Real-Time Vehicle Type Recognition,” Nov. 2020. doi: 10.1109/ICCE-Asia49877.2020.9277040.

[19] N. Wojke, A. Bewley, and D. Paulus, “Simple online and realtime tracking with a deep association metric,” in *2017 IEEE International Conference on Image Processing (ICIP)*, Sep. 2017, pp. 3645–3649. doi: 10.1109/ICIP.2017.8296962.