IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Video-based vehicle speed estimation using speed measurement metrics

KEATTISAK SANGSUWAN[1], MONGKOL EKPANYAPONG[1]
[1]School of Engineering and Technology, Asian Institute of Technology, Khlong Nueng 12120, Thailand.
Corresponding author: Keattisak Sangsuwan (e-mail: st117815@ait.asia).

ABSTRACT Camera system is widely used as a road traffic monitoring nowadays but if using the system as a speed camera, an additional speed sensor is required. In this work, we demonstrate a novel method to estimate speed of vehicle in traffic video without using additional sensor. We implement two speed measurement models including measuring vehicle moving distance in a given unit time and measuring vehicle traveling time in a given unit distance. To get parameters of the models, we define four virtual intrusion lines on the road in camera view. Then, YOLOv3, DeepSORT, GoodFeatureToTrack ,and Lucas-Kanade pyramidal optical flow algorithm are implemented together for vehicle detection and tracking while the target vehicle moving on the road. From the tracking data, pixel displacement between two consecutive frames while the vehicle crossing each line is measured as Crossing distance. And number of frame that the vehicle consumes while moving from the first line to the other lines is measured as Traveling time. These two parameters at each intrusion line are used as speed measurement metrics. Solution of the metrics are solved by using tracking data of 20 vehicles at 9 difference ground truth speed measured by a laser speed gun. Then, the metrics are used to estimate speed of 813 vehicles. Our best accuracy is with MAE of 3.38 and RMSE of 4.69 km/h when comparing to their ground truth speed. The same dataset are tested on a Multilayer Perceptron Neural Network model. It can reach accuracy with MAE of 3.07 km/h (RMSE 3.98 km/h).

INDEX TERMS Vehicle speed estimation; DeepSORT; YOLOv3; GoodFeatureToTrack; Lucas-Kanade optical flow; Vehicle tracking

I. INTRODUCTION

VEHICLE speed measurement can be categorized into two types including intrusive and non-intrusive technologies [1], [2]. For intrusive technologies, the speed sensors are embedded under the road surface. These sensors, for examples, are Inductive loop detectors, Magnetic detector, Piezo-electric, Weight-In-Motion. Installation and maintain of these sensors are costly and difficult. For non-intrusive technologies, the speed sensors can be divided into overhead mounted and side-fired sensors, for example, infrared speed sensor, microwave radar speed sensor, and laser speed sensor. These advanced sensors are costly but easy to install, access, and maintain [1]–[3]. However, both intrusive and non-intrusive technologies require a camera for applying as an automatic road speed limit violation system (called speed camera system) because the camera has to be used for taking a photo of the speed violation vehicle. On the other hand, many cameras (without speed sensor) are widespread installed on highways, Toll ways, urban road, or intersections just only for traffic monitoring or traffic management purpose [4]. Therefore, it would be a great advantage if these cameras can be integrated with speed measurement function without additional speed sensor.

Recent advanced computer vision technologies introduce numerous studies and researches in vision-based vehicle speed measurement. The main effort is trying to adapt traffic monitoring camera to be used as a speed camera without using additional speed sensor. With many proposed solving techniques over this problem, the solution can be broken down into three main groups [5] including, camera settings and calibration techniques, vehicle detection and tracking technique, and the technique of ground truth speed generation for accuracy evaluation. Camera settings and calibration techniques concern complicated work of getting parameters for transferring between 2D camera image plane (x, y) and 3D real-world coordinate system (X_w, Y_w, Z_w) [6]–[9]. Vehicle detection and tracking technique can be done by various approaches including background/foreground sub-

traction [10]–[12], feature-based detection [6], [7], [13], [14], machine learning-based detection [15]–[20]. Ground truth speed generation is the work to generate data for evaluating the proposed speed measurement model [16]. The data can be generated by their existing vehicle speed measurement system both intrusive or non-intrusive technologies. In summary, making camera as a speed sensor is a complicated task and these make vision-based vehicle speed measurement system still being a challenge problem to be solved for a great potential advantage.

In this work, we focus on estimating speed of moving vehicle in video frames received from a fixed Field of View (FoV) traffic camera system. The goal is to provide a method which is easy to integrate speed measurement function into the system without using additional speed sensor.

Our proposed process starts by applying a Deeplearning-based computer vision technology, YOLOv3 [21], and Simple Online Real-time Tracking with Deep association metric algorithm (DeepSORT) [22] for vehicle detection and tracking while GoodFeatureToTrack [23] and pyramidal Lucas-Kanade optical flow [24] algorithms are supported to increase tracking precision by tracking the vehicle at wheel area. These algorithms together with four virtual intrusion lines on the videos frame are used to generate speed measurement metrics and then can be used to estimate speed of the other vehicle in the same FoV camera. The main contributions of this work are:

1) We show that YOLOv3 with DeepSORT and feature tracking at wheel area perform better vehicle trajectory than tracking vehicle using YOLOv3 detection box.

2) We introduce two speed measurement metrics with simple machine learning method techniques to obtain higher accuracy.

3) We show that the metrics for speed measurement can reduce error from fault detection.

4) We propose an input metric for vehicle speed estimation by using a simple machine learning model.

5) We analyze the impact and errors arising from these two speed measurement metrics and on changing of video frame rate.

The remainder of this paper is organized as follows. Section II presents some related works about vehicle speed estimation with difference computation technique. Section III is our proposed method for vehicle speed estimation. Section IV presents the experimental results. Finally, in Section V is the conclusion of this work.

II. RELATED WORKS

Vision-based vehicle speed measurement is work to analyze a series of consecutive traffic video frame. Vehicle must be detected first and keep tracking in following frame to get moving speed in pixel per second. And then, the speed on image is converted to real-world speed (km/h) by using pixel-to-meter relationship (using camera perspective or geometric transformation). In this section, we discuss the existing vehicle detection and tracking techniques.

A. BACKGROUND-FOREGROUND BASED

Vehicle detection using background-foreground technique is a computer vision technology to detect vehicle in traffic video frame by using frame difference. The technique is the subtraction between reference frame and current frame. The reference frame with no vehicle is considered as a background image while another frame with a vehicle is considered as the current frame. Subtraction between these two frames results in a foreground image showing location of the detected vehicle (as a group of difference pixels). Then, a series of the foreground image can be generated by subtracting the background with each traffic video frame. Later, vehicle moving distance between frames (or pixel displacement) can be calculated by subtracting (x, y) values among the foreground image. Where, x and y values are horizontal and vertical pixel location which represents geometric center of the detected vehicle (called centroid). Finally, vehicle moving distance in pixel unit comparing to video frame rate in second can be converted to km/h by using pixel-to-meter ratio. In study of Genyuan *et al.* [11], they used simple machine learning algorithm to model background (Fig.1-a top) and detecting vehicle on their traffic video. The centroid was used for vehicle tracking (Fig.1-a bottom). Finally, speed of the moving vehicle was estimated by using pixel-to-meter ratio referred to the road mark. The best accuracy was with the side view traffic video (comparing to top and intersection view). In [12], a near-top side view camera was used with background-foreground vehicle detection. By tracking the centroid, the speed estimation accuracy was 94% with in +3/-2 km/h of 1,870 vehicles speed in range 0 to 79 km/h. However, detecting and tracking vehicle using this technique is sensitive to background, light intensity, vehicle occlusion, and the other moving object in the frame.

B. FEATURE BASED

Small point on the detected vehicle is hi-lighted for some studies. In [6], after vehicle was detected by background-foreground technique, SURF [25] and FLANN [26] algorithms were implemented for vehicle feature detection and tracking. Although, they claimed that it was robust method when facing light change condition but they still had problem with vehicle occlusion. In [7], the moving vehicle was detected by background-foreground technique then some features on the vehicle license plate was selected and tracked by using GoodFeatureToTrack [23], pyramidal Lucas-Kanade [24], and SIFT [27] algorithm (Fig.1-b). Later, tracking distance in pixel unit was transformed to meter by using camera perspective transformation. And finally, vehicle moving speed was estimated based on video frame rate. The error value was reported at -0.5 km/h comparing with ground truth speed detected from inductive sensor in range between 10 to 69 km/h. However, detecting and tracking vehicle on license plate has some limitations, for example, the camera has to be installed in a near FoV with high resolution for good license plate detection, height of license plate from ground level on the vehicle is sensitive to estimated speed, the license plate

has to be in a good condition.

Another interesting feature detection was proposed in [28]. Instead of detecting feature of the vehicle, they detected vehicle movement pattern vector in the traffic video. They systematically placed four intrusion lines on a single lane road. And then, the passing vehicle was detected by optical flow algorithm. The passing frame number on each line was recorded as the vector. Finally, the probability distribution function model was applied to the vector for estimating vehicle speed. The maximum error 4% was reported comparing to ground truth speed measured by GPS. However, a single lane road is required and only a narrow speed range between 72 to 94 km/h was reported in the study.

C. MACHINE LEARNING BASED

A recent computer vision technology on machine learning such as Deeplearning-based object detection algorithm [29] is also easily applied for vehicle detection [5]. The algorithm can be fed by a traffic video frame and, if there is only one vehicle on the frame, the output will be a rectangle box showing location of the detected vehicle on the image plan. The box represents by four values including top-left horizontal and vertical location (x, y) and size of the box $(width, height)$ in pixel unit.

Deeplearning-based object detection algorithm can be categorized into two types: Two-stage and One-stage. Two-stage type will find regions of possible object at the first stage and then perform object classification on the second stage while these two stages are combined together for One-stage type. Among various proposed algorithms over these two types, an One-stage type such RetinaNet [30] showed outstanding performance as illustrated in [21]. But when testing with many traffic videos for vehicle detection, YOLOv3 showed a lowest Floating Point Operation Per second (FLOPs) resulting in faster frame per second as illustrated in [31]. However, Two-stage algorithm was applied in [16] for vehicle detection. Faster-RNN [32] was used to detect vehicle in traffic video. Tracking position of the 2D rectangle box given by the detecter was not precise enough for speed estimation because it had variation frame by frame (example in violet dot tracking line in Fig.1-c). Then, the box was transformed to 3D bounding box and mapping to actual 3D box of vehicle in database. By this way, size of the detected vehicle was estimated precisely and tracked with association of Kalman filter. Finally, moving distance in pixel unit with traveling time were used to estimate speed referring to pixel-to-meter ratio. Although, the study archived a good mean speed measurement error at 1.10 km/h but they used a complicated work processes and vehicle 3D model metadata were required for 2D to 3D bounding box mapping.

Another work in [33], they used Kalman-based tracker called Simple Online and Realtime Tracking (SORT) [34] and DeepSORT [22] to track directly center of bottom of the detected box given by Mask-CRNN [35]. Finally, vehicle speed in pixel unit was transformed to real world unit in km/h by using camera perspective transformation. The error was reported at 15.35 km/h (9.54 mph). One-stage algorithm was in study of D. Bell *et al.* [17]. They used YOLOv2 [36] for vehicle detection. A bottom center of the detected box was tracked by SORT [34] (violet dot line in Fig.1-c). Then, vehicle moving speed was estimated by using camera perspective transformation. Their error was 2.25 km/h (0.25 m/s) but the samples were a few vehicle in urban traffic at low speed in range 1.6 to 15.8 km/h (0.452 to 4.393 m/s). Héctor *et al.* [18] used One-stage YOLOv3 algorithm to detect passing vehicle on side road and Kalman filter was used for tracking center of the detected box. Speed of the vehicle was estimated by linear regression models and machine learning models. Their best mean absolute error was 1.695 km/h. However, this work was performed on side road which had high pixel-to-meter ratio.

Fully apply of Deeplearning algorithm for vehicle speed estimation was presented in [19], [20]. One-stage object detection algorithm YOLOv5 [37] was used to generate a series of vehicle detection box from side road video dataset. And another Deeplearning model was used to estimate speed of vehicle after trained. The average error was reported at 2.76 km/h and 4.08 km/h respectively.

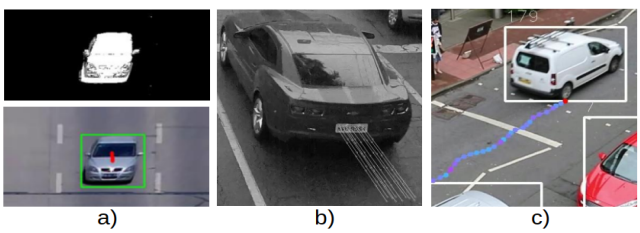

FIGURE 1. Vehicle detection and tracking methods. From left, a) Background-foreground based [11] (Top is vehicle detection, bottom is centroid tracking), b) feature based [7] (License plate tracking), and c) Machine learning based [17] (Box tracking).

D. PROBLEMS AND SOLUTION

As our analysis, background-foreground vehicle detection technique faces with unstable of background-foreground, changing of light intensity, the detector detects other moving object or shadow in the frame. These problems affect directly to speed estimation accuracy. We think these problems shall be solved by using Deeplearning-based vehicle detection algorithm. The algorithm has a good performance in vehicle detection under difference situation such as night time, rainy, or cloudy [38]. In addition, because the algorithm detects vehicle by using the whole body of vehicle, then, this is easily for human to identify characteristics or classify type of the target vehicle for further purpose. Among various proposed Deeplearning-based vehicle detection algorithms, we select YOLOv3 as our vehicle detector with the following reasons:

1) YOLOv3 facilitated with multi-scale detection. This is very helpful for us because our target vehicle is moving away from camera during detect and tracking period.

2) YOLOv3 has a fast processing speed. This is potentially allowed us for real-time processing.

3) YOLOv3 is well tested for vehicle detection in many studies.

4) YOLOv3 is user friendly [39] and its accuracy is acceptable in our long tracking period.

Although there are many variant versions of YOLO such v4, v5, R, x, PP-YOLO, v6, v7, v8. These difference version were proposed with some improvement features for difference kind of target application [40]–[42]. For example, YOLOv7 is the most accurate for tiny object detection due to higher input resolution [43]. But, here, we do not need to detect a small vehicle.

Another problem on vision-based vehicle speed estimation is tracking accuracy. As in [17], tracking vehicle on the detected box given by the detector provides unstable tracking line. Then, we propose to detect and track feature on the vehicle as [7]. We select DeepSORT for being vehicle identification and use pyramidal Lucas-Kanade optical flow algorithm for feature detection and tracking. In addition, we improve the accuracy by applying vehicle moving pattern vector borrowing from [28].

III. PROPOSED METHOD

A. VEHICLE SPEED MEASUREMENT IN TRAFFIC VIDEO

In general, to get speed of moving vehicle, we can simply measure in two models. The first model is measuring the moving distance in a given unit of time. The second model is measuring traveling time in a given unit of distance. And then, speed in m/s or km/h can be calculated. Here, we are going to do both measurement models by using a traffic camera. But to use a camera as a vehicle speed sensor, we have to note that the camera is a discrete sensor with a constant sampling rate at frame per second (fps). Therefore, the camera can only get a series of temporal movement location of vehicle on road surface in video frame. It is rather hard to detect exactly at a specific given point on the road. Hence, the speed estimated by the camera is a discrete value.

For the first speed measurement model, measuring moving distance of target vehicle in a given unit of time, we use camera frame rate as a given unit of time and measuring moving distance of the vehicle between two consecutive frames. In this case, we have to note that error of distance measurement is higher at the farther point comparing to the error measuring at the closest point from camera. Hence, measuring distance is better to be done at the closer point for better accuracy.

If we consider position of the moving target vehicle between any two consecutive frames, moving distance of the vehicle between these two frames called pixel displacement (Δd) is represented by :

$$\Delta d = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \tag{1}$$

Where, (x_i, y_i) is horizontal and vertical pixel position of the target vehicle on frame i and (x_{i+1}, y_{i+1}) is the horizontal and vertical position of the target vehicle on the later frame $i + 1$. Speed of the vehicle (v) can be calculated by dividing moving distance with traveling time which is camera frame rate (fps) as:

$$v = \frac{\Delta d}{fps} \tag{2}$$

Unit of Δd is in pixel and fps is in second then v is in pixel per second. If Δd can be transferred to the distance in real world unit, for example, using pixel-to-meter ratio. Then, (2) can be used to estimate speed of moving vehicle between any two consecutive frames in m/s or km/h.

For the second speed measurement model, we measure vehicle traveling time in a given unit of distance. If we define a unit of distance by using two virtual intrusion lines on the road surface in video frame, then, we can measure traveling time of the target vehicle between these two lines by counting usage frame (n) and multiply by video frame rate (fps). As mentioned earlier, camera is a discrete sensor then this traveling time is a discrete number given by $n \times$ fps where n = 1, 2, 3, 4,..., N and n is the usage frame which starts counting after the vehicle crosses the first virtual intrusion line until the vehicle crosses the second line. Then, speed of the vehicle can be calculated by:

$$v = \frac{L}{n \times fps} \tag{3}$$

Where L is a given unit of distance defined by those two virtual intrusion lines. If L is in pixel, v will be in pixel per second. If L can be defined in meter then v could be in km/h.

B. VEHICLE MOVING PATTERN AND SPEED ESTIMATION EQUATIONS

In this analysis phase, two speed measurement models in section A are implemented. We create some traffic videos by setting up a traffic camera at the height approximately 7.3m on a sky crossing bridge over Phaholyothin road, Pathumthani, Thailand as shown in Fig.2 (Google map coordinate: 14.07500439231682, 100.617625303785l3). A new iPad Pro 2012 is settled up for capturing traffic videos. Its back camera is configured to Full HD resolution 1920x1080 pixels and frame rate is 60 fps. Laser speed gun LTI 20/20 TruSpeed with accuracy +/- 2 km/h is used to measure ground truth speed of the target vehicle in the range between 40 to 60 meters from the camera. Voice reading is used to record speed value from the speed gun into the video during capturing period.

Based on the video frame, four virtual intrusion lines *Start line0, line1, line2* and *line3* are defined at y equals to 490, 310, 226, 164 respectively where image pixel coordinate x=0 and y=0 is at the top left corner of the image as shown in Fig.3. Distance among the lines are known based on road mark sign by using data from Thailand Department of Highways (DOH). The white dash line length (in the green box on Fig.3) is 3 meters and the gap between each white dash lines is 9 meters. However, these information are only for reference, we do not use them in this study. Our target vehicles are the vehicle which passing in Region of Interest

FIGURE 2. Traffic video recording and vehicle speed measurement on the skywalk crossing bridge over Phahonyothin road at the height 7.3m over the road. (Image on the road by Google street view).

(ROI) as the green box shown in Fig.3 because speed limit violation always occurs on this most right lane.

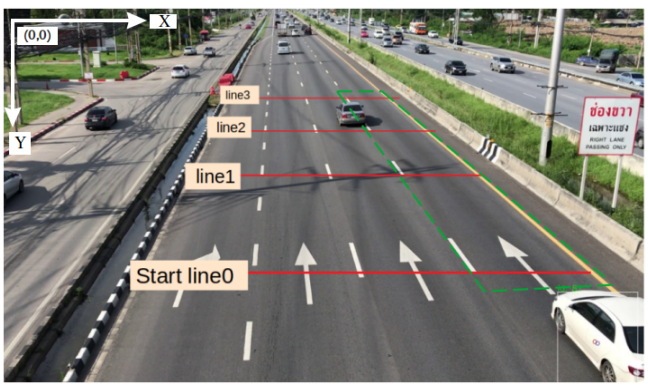

FIGURE 3. Four virtual intrusion lines (Red lines) and ROI location (Green dash box) are defined on the video frame size 1920x1080.

From Fig.3, we now can implement our two speed measurement models (2) and (3). By using four virtual intrusion lines, we can track the vehicle and then can get Δd together with number of usage frame n at each virtual intrusion line as shown on Fig.4.

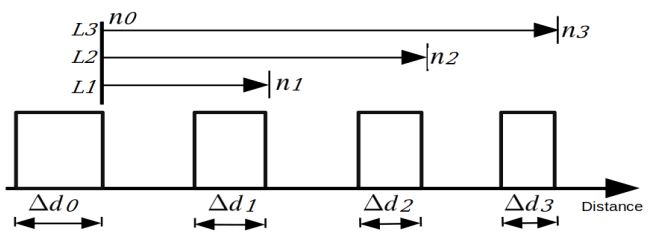

FIGURE 4. Vehicle moving pattern at four intrusion lines.

At this stage, two sets of temporal vehicle movement pattern can be introduced. First, a set of selected Δd called "Crossing distance". It is the pixel displacement between two consecutive frames before and after the vehicle crossing each virtual intrusion line at speed v. Crossing distance represents by:

$$\Delta d_v = [\Delta d0_v, \Delta d1_v, \Delta d2_v, \Delta d3_v] \tag{4}$$

Where Δd_v is Crossing distance set at vehicle moving speed v km/h. $\Delta d0_v$, $\Delta d1_v$, $\Delta d2_v$, and $\Delta d3_v$ are the pixel displacement at *line0, line1, line2,* and *line3* respectively.

The second set of the temporal vehicle movement pattern is a set of usage frame *n* called "Frame counter number". It is the traveling time of the target vehicle which starts counting immediately after the vehicle crossing *line0* until it crosses the other lines at moving speed v. Frame counter number represents by :

$$n_v = [n0_v, n1_v, n2_v, n3_v] \tag{5}$$

Where n_v is Frame counter number set at vehicle moving speed v km/h. $n1_v$ is the usage frame counting after the vehicle crossing *line0* until the vehicle crossing *line1*. $n2_v$ is the usage frame from *line0* to *line2*. And $n3_v$ is the usage frame from *line0* to *line3*, respectively. $n0_v$ is the starting frame and it always equals to one.

In our assumption, if the target vehicle moves with a constant speed v_c km/h in ROI and we know the relationship between image and real-world coordinate, then, speed of the target vehicle can be calculated with (2) by using any Δd in (4) or calculated with (3) by using $n1$, $n2$ or $n3$ in (5). All calculated speeds v have to equals to v_c. Therefore, the speed can be calculated easily if we track the target vehicle precisely to generate Crossing distance and Frame counter number.

In this study, we implement a Deeplearning-based object detection, YOLOv3, to detect vehicle position in the video frame. The detected vehicles are passed as a box with (*x, y, width, height*) to DeepSORT for getting identification number (ID). And then, DeepSORT tracks the vehicle throughout our ROI. However, the box given by YOLOv3 is not accurate enough to track position of the vehicle, then, GoodFeatureToTrack and pyramidal Lucas-Kanade optical flow algorithms are implemented to get better tracking accuracy on the target vehicle. At this stage, we manually point GoodFeatureToTrack algorithm to wheel area of the vehicle because the wheel is not too high from the road surface. And this technique helps to reduce speed estimation error caused by height of tracking point on the vehicle in 3D world coordinate. After GoodFeatureToTrack algorithm selects some points on the wheel area then pyramidal Lucas-Kanade optical flow algorithm is used for tracking these all selected points. Centroid among the selected points is used to be a reference for generating vehicle tracking data. Fig.5 shows 6 points selected by GoodFeatureToTrack (All red dots) and their centroid (Blue dot) is calculated for being the reference and used for generate tracking data. Fig.6 shows the better tracking precision given by GoodFeatureToTrack and Lucas-Kanade optical flow in pyramids algorithm (wheel tracking in blue dot line) comparing to tracking the box given by YOLOv3 (box tracking in green dot line).

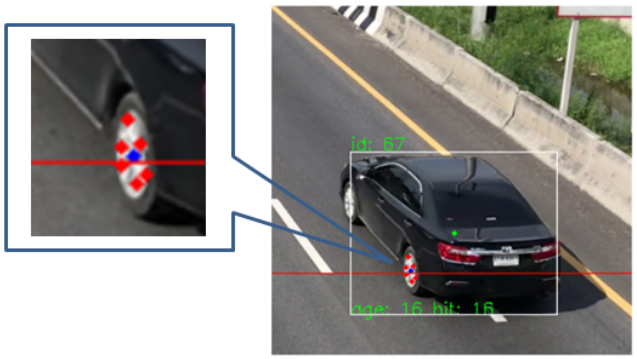

FIGURE 5. **Six points of Feature selection at the wheel of the vehicle (Red dots) and their centroid (Blue dot) for being a reference to generate tracking data.**

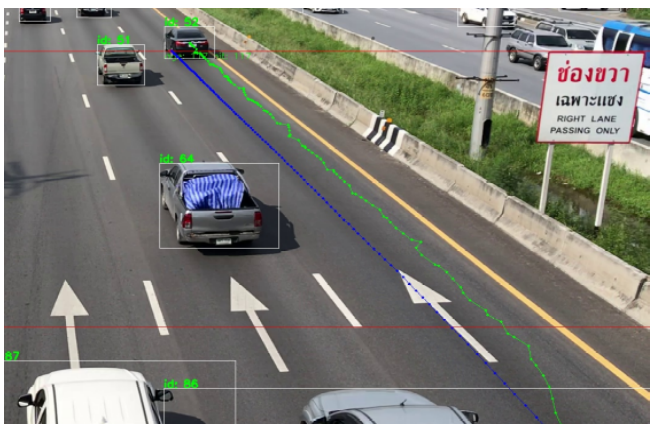

FIGURE 6. **Lower precision from YOLOv3 tracking (Green dot line) comparing to wheel tracking (Blue dot line).**

In this step, we analysis 187 traffic videos recorded by the method as mentioned previously. We have one target vehicle in one video file, each video is processed manually to get tracking data. The process consists of three manual steps as following:

1. The videos are processed by Python code with implementation of YOLOv3 for vehicle detection and DeepSORT for getting Identification number (ID). Result from the code are images of all vehicle entering ROI with its ID.
2. Virtual inspection is performed on each video file for mapping each target vehicle on the video to its entering ROI image. Here, we know ID and speed of each target vehicle. These information are entered in to a CSV (Comma Separate Values) file including video file name.
3. The videos are processed again following information in the CSV file with two additional algorithms, GoodFeatureToTrack and pyramidal Lucas-Kanade optical flow. Here, when the target vehicle entering the frame, the code will pause and allow us to point GoodFeatureToTrack to wheel area of the target vehicle. After that the code uses pyramidal Lucas-Kanade optical flow to track the selected feature until the end of ROI (Blue dot line in Fig.6).

Finally, from the third step, we get 135 tracking data from 135 target vehicles. The data include pixel position of the centroid and frame number. However, some tracking data are not valid because the tracking point disappears during tracking period, lane changing during tracking, and no feature selected by GoodFeatureToTrack algorithm.

Fig.7, from left to right, shows tracking data of 9 target vehicles at ground truth speed 72, 77, 82, 87, 92, 97, 102, 108, and 112 km/h respectively. In the Figure, we can see a linearity decreasing of traveling time (frameCount) when the target vehicle moving with higher speed (Black arrow). In contrast, Δd (pixel displacement) increases by increasing of speed (Blue arrow).

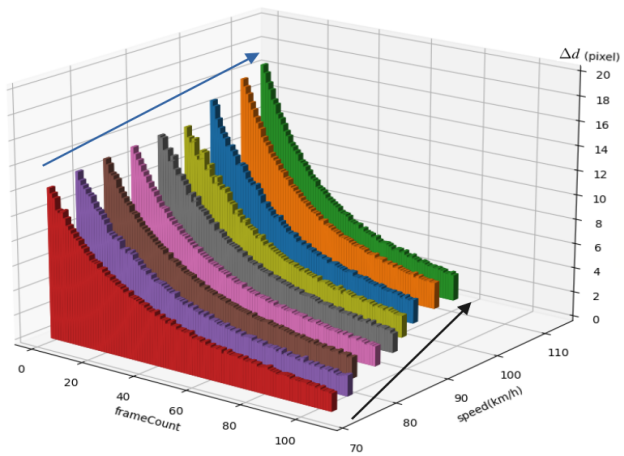

FIGURE 7. **Increasing of vehicle speed causes decreasing of traveling time (frameCount) in ROI (Black arrow) while pixel displacement increases by increasing of speed (Blue arrow).**

By referring tracking data in Fig.7 to (4), we can plot Crossing distance set as in Fig.8. The plot represents linear relationship between vehicle speed and Crossing distance at each virtual intrusion line. Increasing of speed causes increasing of Crossing distance.

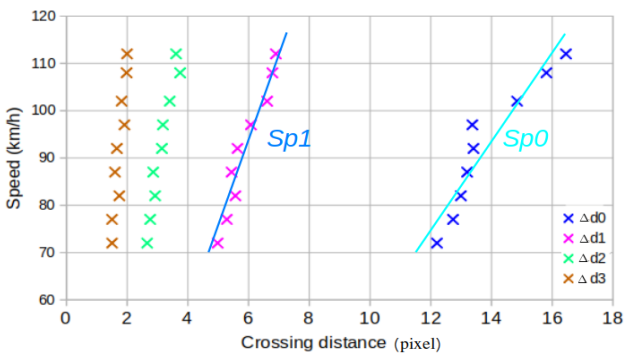

FIGURE 8. **Relationship between Crossing distance ($\Delta d0$, $\Delta d1$, $\Delta d2$, and $\Delta d3$) and vehicle moving speed.**

From Crossing distance in Fig.8, we can get two most linear equations of speed and Crossing distance as:

$$Sp_0 = (slope_0 \times d_0) + B_0 \quad (6)$$

and,

$$Sp_1 = (slope_1 \times d_1) + B_1 \quad (7)$$

Where, $slope_0$, B_0 and $slope_1$, B_1 are a constant value which can be calculated from linear curve fitting method.

In the similar way, by referring tracking data in Fig.7 to (5), we can plot Frame counter number set as in Fig.9. The plot represents linear relationship between vehicle speed and Frame counter number at each virtual intrusion line. Increasing of speed causes decreasing of frame count.

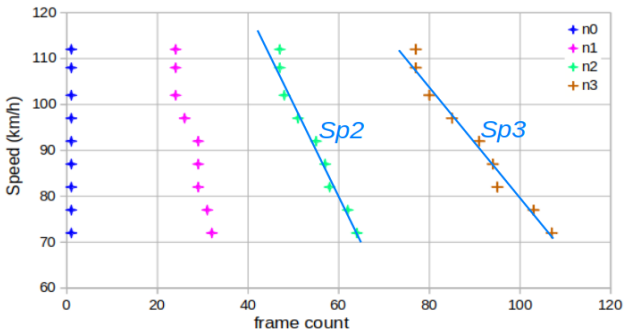

FIGURE 9. Relationship between Frame counter number (n) at each virtual intrusion line ($n0 = 1, n1, n2$, and $n3$) and vehicle moving speed.

From Frame counter number in Fig.9, we can get two most linear equations of speed and Frame counter number as:

$$Sp_2 = (slope_2 \times n_2) + B_2 \quad (8)$$

and,

$$Sp_3 = (slope_3 \times n_3) + B_3 \quad (9)$$

Where, $slope_2$, B_2 and $slope_3$, B_3 are a constant value which can be calculated by using linear curve fitting method.

IV. EXPERIMENTS

A. DATASET

The similar process in analysis phase is used for creating a new dataset but iPad camera, in analysis phase, is replaced by iPhone 13 Mini. The back camera of iPhone 13 Mini is configured to Full HD resolution 1920x1080 pixels at 120 fps. The videos are captured in the afternoon of December 1^{st}, 2022. The captured time is between 11:00 to 16:00 Local time. The weather conditions during the capturing day are clear with very bright sun light. All target vehicles are on the most right lane in the video frame. We capture as a short video file with a few target vehicles for easy processing in the next step.

From 138 video files, with totally 898 target vehicles, the steps to precess the video file are same as in our analysis phase. But one video file here can consist of more than one target vehicle. Then, frame number when the target vehicle entering the frame has to be added to the CSV file for being reference. Information in the CSV file include, video file name, entering frame number, vehicle ID, and ground truth measured speed. The total 898 vehicles have minimum speed at 44 km/h and maximum speed at 127 km/h. There are 70% of the target vehicle moving at speed between 70 to 90 km/h.

The whole dataset here will be made available online for research purpose and shall be considered as one of our work contribution. (The link will be provided at the publication time.)

B. EXPERIMENTAL SETTING

In our analysis phase, wheel of the target vehicle was selected manually but this is rather hard for the new dataset. Then, we train one YOLOv3 model especially for wheel detection in our case. Training data are taken from all vehicles entering ROI in the most right lane in the analysis phase. Another change here is we track directly to the selected points. Because tracking by using centroid among the selected points was not stable. Here, only two tracking points are selected among many good features given by GoodFeatureToTrack algorithm. Flow diagram of our proposed system illustrates in Fig.10.

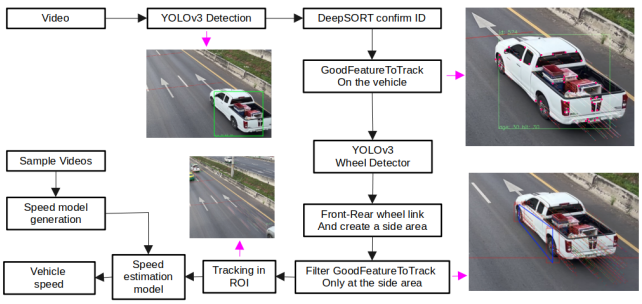

FIGURE 10. Flow diagram of our system and outputs.

From the flow diagram, first, four visual intrusion lines including *line0, line1, line2* and *line3* are defined on the video frame at y = 655, 447, 351, and 278 respectively (frame resolution is 1920x1080, the top left corner is x=0 and y=0 as in Fig.3). Second, after entering the frame, all vehicles are detected by YOLOv3 and receive ID from DeepSORT in a few frame later. Then, we filter all vehicle by mapping their ID to the ID in the CSV file and the target vehicle should appear at the corresponding frame number. Third, GoodFeatureToTrack is applied to the target vehicle for getting tracking points (Big red dots over the white vehicle in Fig.10 or more clear in Fig.11). We keep tracking every points on the target vehicle until the vehicle enters wheel detection area (at y=750). At this point, YOLOv3 for wheel detection is applied to the vehicle for identify location of two wheels (Two red rectangle at the wheel in Fig.11). Then, a blue rectangle is created to link between the box of two wheels (Fig.11). Later, two tracking points which are nearest to horizontal half of the blue rectangle will be selected. From now, only these two selected points are tracked (Fig.12) until the end of ROI. And their pixel positions in each frame are recorded together with its frame number as a tracking data. Fig.12 shows two tracking points are tracked in ROI.

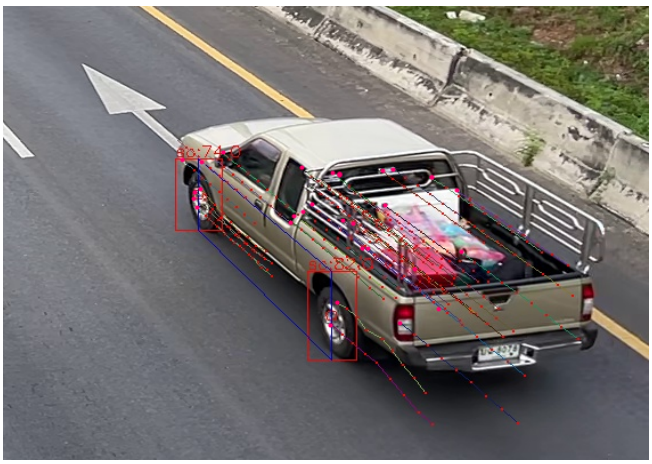

FIGURE 11. Wheel detection (Red boxes) and area selection (Blue box) at the side of target vehicle after the vehicle crossing wheel detection point at y=750.

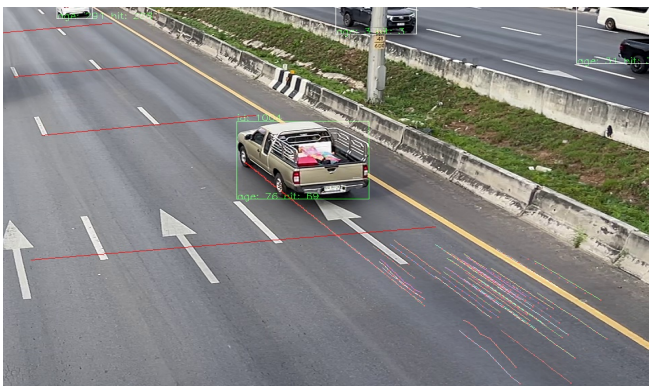

FIGURE 12. Only two selected tracking points at the wheel area are continuously tracked while the target vehicle is moving in ROI.

From 898 tracking data of 898 target vehicles, we use vanishing point technique to remove some invalid data including, tracking data while the target vehicle is moving out from the most right lane, tracking data which lose of tracking point and can not recover by DeepSORT, and the tracking point is not on the target vehicle. Finally, we have only 833 valid tracking data. Speed distribution of the valid tracking data are illustrated on Fig.13.

For video processing and computation in the analysis phase, we work on the work station CPU: Intel® Core i7-8750H 2.2GHz with memory 32 GB and GPU: nVidia GeForce GTX 1060 Mobile. The operating system was Ubuntu 18.04; processing software was coded by using Python 3.6.9 with supported libraries from OpenCV 4.2.0, CUDA 10.0 CuDNN 7.6, and Tensorflow 1.13.1 for video processing. But this experiment is performed on the difference hardware configuration. It is on CPU: Intel® Core i7-9700K 3.6GHz with memory 16 GB and GPU: nVidia GeForce GTX 1080Ti. The operating sysem is Ubuntu 18.04; Processing software is Python3 3.6.9, CUDA 11.0, CuDNN 8.0, Tensorflow 2.6.0 for implementation of Multilayer Perceptron (MLP) testing on the same hardware.

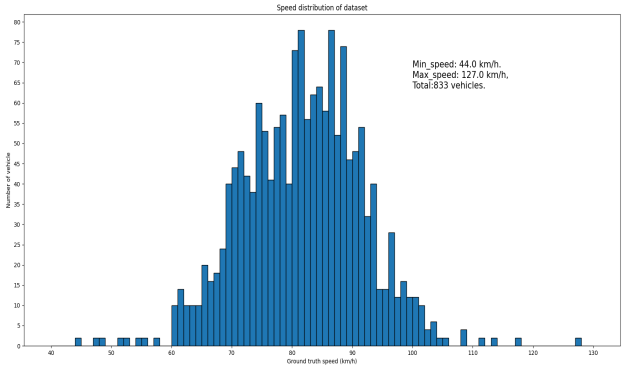

FIGURE 13. Number of vehicle at each ground truth speed in the dataset after filtering invalid data.

C. EVALUATION

Statistical evaluation Mean Absolute Error (MAE) is adopted as evaluation tool. It is the summation of each absolute error from individual estimated speed then divided by total number of vehicles which is defined as:

$$MAE = \frac{1}{M}\sum_{m=1}^{m=M} \mid v_m^g - v_m^e \mid \tag{10}$$

Where M is total number of vehicle to be measured, v_m^g is the ground truth speed of vehicle m, and v_m^e is the estimated speed of vehicle m. Root-Mean-Square Error (RMSE) is used to represent the sensitivity of variation to large errors. It is the square root of the mean of squared residual defined as:

$$RMSE = \sqrt{\frac{\sum_{m=1}^{m=M} (v_m^g - v_m^e)^2}{M}} \tag{11}$$

D. SPEED ESTIMATION RESULT

Results from the experiment are speed estimation by using our metrics model as an input of the linear equations. Neural Network estimation is a parallel test for comparing accuracy but number of test data is lower than linear estimation equation method.

1) Linear equations speed estimation

Here, we test vehicle tracking data with 3 levels of frame rate including 120 fps, 60 fps, and 30 fps. From the original frame rate at 120 fps, tracking data of 20 vehicles are used to find *slope* and *B* of each linear equation (6), (7), (8), and (9). Five tracking data from each vehicle speed 79, 86, 93, and 100 km/h are processed by Framedrop (1:1). Then, they are filtered by using (4) and (5) to get Crossing distance set and Frame count number st at each virtual intrusion line. Later, linear curve fitting is applied to get *slope* and *B*. Finally, the equations are used to estimate speed of 813 vehicles. The work flow is illustrated on Fig.14.

The same process is repeated but Framedrop will be 2:1 and 4:1 for dropping tracking data down to 60 fps and 30 fps,

respectively. Table 1 shows all linear equations (6), (7), (8), and (9) with their $Slope$ and B at the difference fps received from 20 samples.

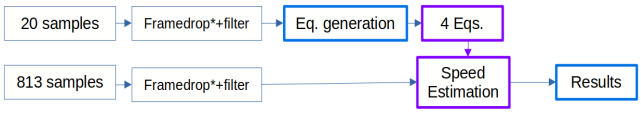

FIGURE 14. Work flow for the experiment, Framedrop is 1:1, 2:1, and 4:1.

TABLE 1. Constant value $Slope$ and B of each linear equations received from 20 samples.

Speed Eq.	fps	Slope	B	R^2
Sp0	120	9.4513	3.5	0.9937
	60	4.7482	3.5	0.9872
	30	2.3535	3.5	0.9809
Sp1	120	22.9307	3.0	0.9948
	60	11.5854	3.0	0.9958
	30	5.7692	3.0	0.9951
Sp2	120	-0.6711	160.0	0.9970
	60	-1.3284	160.0	0.9971
	30	-2.5964	160.0	0.9960
Sp3	120	-0.4194	163.0	0.9973
	60	-0.8317	163.0	0.9974
	30	-1.6343	163.0	0.9962

Table 2, 3, and 4 show speed estimation result by using (6), (7), (8) , and (9) referring to calculated values in Table 1. Based on MAE and RMSE value, the best accuracy of speed estimation using Crossing distance (6) and (7) can be founded on Table 4 at 30 fps with MAE 5.92 km/h, RMSE 12.42 km/h and 4.37, 7.65 km/h respectively. Crossing distance equations give lower accuracy when fps is higher to be 60 and 120 fps as shown on Table 3 and Table 2.

TABLE 2. Estimated speed statistical evaluation of 813 vehicles by using Sp_0, Sp_1, Sp_2, and Sp_3 at 120fps.

Speed Eq.	MAE(km/h)	RMSE(km/h)
Sp_0	10.98	33.63
Sp_1	7.37	21.98
Sp_2	3.38	4.69
Sp_3	3.51	6.72

TABLE 3. Estimated speed statistical evaluation of 813 vehicles by using Sp_0, Sp_1, Sp_2, and Sp_3 at 60fps.

Speed Eq.	MAE(km/h)	RMSE(km/h)
Sp_0	6.74	17.56
Sp_1	5.03	12.23
Sp_2	3.80	5.59
Sp_3	4.03	7.70

The better speed estimation accuracy can be founded on Table 2 by using Frame counter number equations (8) and (9). The accuracy is at MAE 3.38 km/h, RMSE 4.69 km/h from (8) and it is a bit lower by using (9) which has MAE 3.51 km/h, RMSE 6.72 km/h. However, these are still better than using Crossing distance equations at all video frame rates. Frame counter number equations are opposite with Crossing distance equations. For Frame counter number equations, the accuracy increases at higher videos frame rate as shown on Table 3 and Table 4.

TABLE 4. Estimated speed statistical evaluation of 813 vehicles by using Sp_0, Sp_1, Sp_2, and Sp_3 at 30fps.

Speed Eq.	MAE(km/h)	RMSE(km/h)
Sp_0	5.92	12.42
Sp_1	4.37	7.65
Sp_2	5.30	8.05
Sp_3	5.58	11.17

2) Neural Network speed estimation

We test if a fully connected neural network can get the same accuracy for vehicle speed estimation after trained by some of our data. We create 5 hidden layers of 20 Dense nodes. Activation function is the combination of tanh and linear. Dataset using here are the same 813 tracking data but we take only Crossing distance data at 30 fps and Frame counter number at 120 fps because they can provide best accuracy. Then output of the model is a speed of target vehicle which is a function of *d0, d1* of 30 fps and *n2, n3* of 120 fps as presented on Fig. 15.

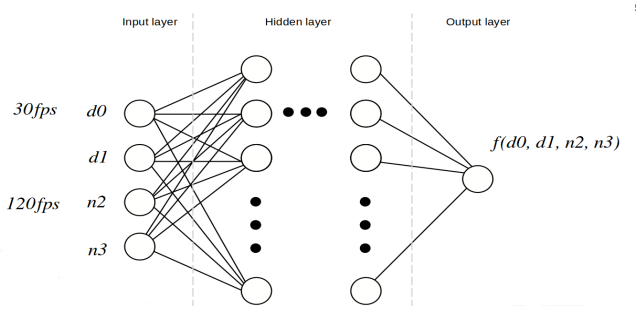

FIGURE 15. Inputs and output of Fully connected neural network model for the experiment.

The tracking data is separated into train data and test data. There are about 70% of the dataset (570 vehicles) used for training and the rest 30% (243 vehicles) are used for speed estimation test. Fig.16 shows speed distribution of the train data and Fig.17 shows speed distribution of the test data.

Mean Squared Error (MSE) is configured as the loss for training our model. Fig. 17 is the comparison between loss of training data and testing data. Our best result is at Epoch 141,197 with MAE 3.07 km/h and RMSE at 3.98 km/h.

To compare accuracy of speed estimation using neural network with linear equations, the same test data (243 vehicles) are statically summarized as in Table 5. This shows machine learning model provides better performance than linear equations when using our metrics as the input.

3) Expected result

In comparison to other research, many difference techniques are applied on video-based speed measurement as shown in

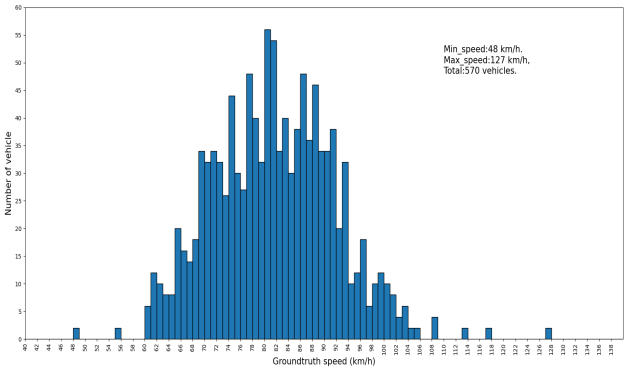

FIGURE 16. Speed distribution of 570 vehicles from training dataset.

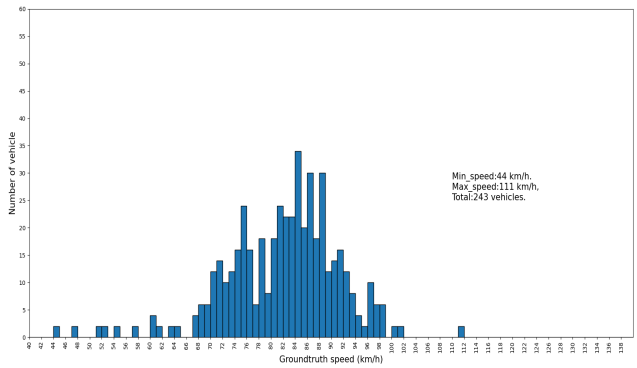

FIGURE 17. Speed distribution of 243 vehicles from test dataset.

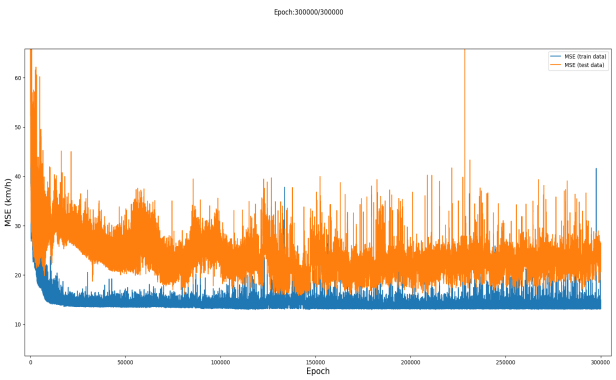

FIGURE 18. Loss of training data and test data during training upto 300000 epoch.

TABLE 5. Estimated speed statistical evaluation of 243 vehicles by using Sp_0, Sp_1, Sp_2, and Sp_3 at 120 fps.

Speed Eq.	MAE(km/h)	RMSE(km/h)
Sp_0	5.09	9.83
Sp_1	3.81	4.94
Sp_2	3.75	5.56
Sp_3	3.60	5.39

[5]. Here we compare some similar techniques for accuracy comparison based on their report as shown in Table 6.

The first four studies are hi pixel-to-meter ratio method. In [18], their accuracy can reach to MAE 0.95 km/h by using linear estimation method while Multilayer Perceptron can do at 3.17 km/h. Our Multilayer Perceptron shows better accuracy with our input metric. We can reach lower MAE at 3.07 km/h and this is also agreed by its RMSE at 3.98 km/h. Our linear model has higher error because we have lower pixel-to-meter ratio at farther FoV. Another study in [19], their peed estimation by using 1D-CNN model can do better than our at RMSE 2.76 km/h but this value is not too far from our at 3.98 km/h although they are working with hi pixel-to-meter ratio. Our Multilayer Perceptron are also a little bit better of accuracy comparing to RNN speed estimation model [20] at RMSE 4.08 km/h. But we have to re-mind that in [20], they are working on side-road FoV with a hi pixel-to-meter ratio video frame.

Luvizon *et al.* [7] using KLT tracking on license plate with top-down near FoV. Their reported errors of speed estimation were in between -4.68 km/h and +6.00 km/h but most of the vehicle had low moving speed between 10 to 59 km/h.

Javadi *et al.* [28] reported their speed estimation average error 1.77% only at vehicle moving speed at 72, 73.8, 91.08, and 94.32 km/h.

D. Bell *et al.* [17] used SORT to track YOLOv2 detection box. Their approach presented promising performance of the computation processes with average RMSE at 2.25 km/h (0.625 m/s). However, their experiment was performed in urban road traffic under non-free-flowing condition with a few samples of vehicle.

Amit Kumar *et al.* [33], similar FoV and method with ours, used Mask-RCNN for vehicle detection and tracking by SORT in the NVIDIA AI City challenge 2018. Their speed estimation RMSE was reported at 15.35 km/h (9.54 mph) which lower than ours.

Dong *et al.* [44] can reach better MAE at 2.71 km/h than ours by using 3D ConvNets. But their RMSE shows variation of large error at 14.62 km/h almost similar to [33]. This illustrates clearly the better performance of our input metrics of Multilayer Perceptron. However, we can not reach to accuracy of Sochor *et al.* [16] at MAE 1.10 km/h. But their method requires 3D model metadata of vehicles while we use only simple metric model from only 20 vehicles. The comparison results are illustrated on Table 6 and Fig.19 compares the difference of FoV between each study.

V. CONCLUSION

A simple video-based vehicle speed measurement method is presented in this study. The main motivation is to integrate a road traffic camera system with an automatic vehicle speed limit enforcement function.

Two speed measurement models are considered here, in our study, including measuring moving distance of the vehicle in a given unit of time (Crossing distance) and measuring vehicle traveling time in a given unit of distance (Frame counter number). By implementing YOLOv3, DeepSORT, GoodFeatureToTrack, and pyramidal Lucas-Kanade optical flow algorithm for vehicle detection and tracking, tracking

data of 20 vehicles at ground truth speed 72, 77, 82, 87, 92, 97, 102, 108, and 112 km/h are used to create speed estimation metrics and used to estimate speed of 813 vehicles. The result shows the best speed estimation accuracy at MAE 3.38 km/h RMSE 4.69 km/h. This best accuracy can be received from the relationship between Frame counter number and ground truth speed at the third virtual intrusion line (line2) using video frame rate at 120 fps. And the accuracy is getting lower to be MAE 3.80 km/h RMSE 5.59 km/h and MAE 5.30 km/h RMSE 8.05 km/h when the frame rate is dropped down to 60 and 30 fps respectively. While our Multilayer Perceptron model provides the best accuracy at MAE 3.07 km/h RMSE 3.98 km/h. The model consists of 20 Denses 5 hidden layers and it is trained with 70% of dataset. The rest 30% is used to evaluate accuracy of the model.

Our proposed speed measurement metrics has a good accuracy comparing to the previous study and the metrics is simple to calculate with only 20 samples at wide range speed. The metrics can also apply as an input of Multilayer Perceptron for speed estimation.However, our method also has some limitations as following. The first one is failure of vehicle detection and wheel tracking because of the occlusion from the vehicles in the next lane. The worst case happens when there is a big vehicle such as truck or bus in the next lane. It will cover all camera view. The second problem is even though our approach does not mainly rely on the detector. Missing vehicle detection is allowed on some frames. But if the missing happens during the target vehicle crossing the intrusion line, this will result in large error of speed estimation. For future work, we plan to improve speed measurement of dataset with more accurate ground truth speed.

REFERENCES

[1] Tom V. Mathew, eds., *Automated Traffic Measurement.* Bombay, India, Indian Institute of Technology Bombay, 2023, Accessed on: Sep. 11, 2023. [Online]. Available: https://www.civil.iitb.ac.in/tvm/nptel/524_AutoMer/web/web.html#x1-150002.2

[2] Paraskevi Michalaki *et al.*, "A Sensor-based System for Monitoring Hard-shoulder Incursions: Review of Technologies and Selection Criteria," *ICTTE*, vol. 81, no. 02019, Oct. 2016, doi: 10.1051.matecconf/20168102019.

[3] Traffic Monitoring Guide eBook. US Department of Transportation Federal Highways Administration Washington DC, 2014. [Online]. Available: https://www.fhwa.dot.gov/policyinformation/tmguide/tmg_2013/traffic-monitoring-theory.cfm. Accessed on: Sep 11, 2023.

[4] Heba A. Kurdi, "Review of Closed Circuit Television (CCTV) Techniques for Vehicles Traffic Management," *IJCSIT*, vol. 6, no. 2, pp. 199–206, Apr, 2014. Accessed on: Sep. 14, 2023, doi: 10.5121.ijcsit.2014.6216.

[5] David F. Llorca, Antonio H. Martínez, Iván G. Daza, "Vision-based vehicle speed estimation for ITS: A survey," *IET Intelligent Transport Systems*, vol. 15, issue 8, pp. 987–1005, Aug. 2021, doi: 10.1049/itr2.12079.

[6] Agustín Gabriel Yabo *et al.*, "Vehicle classification and speed estimation using computer vision techniques," in *AADECA 2016*, Buenos Aires, Argentina, 2016.

[7] Diogo C. Luvizon *et al.*, "A Video-Based System for Vehicle Speed Measurement in Urban Roadways," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, issue. 6, pp. 1393–1404, Sep. 2017, doi: 10.1109./TITS.2016.2606369.

[8] Chen Wang and Aibek Musaev, "Preliminary Research on Vehicle Speed Detection using Traffic Cameras," in *2019 IEEE Int. Conf. on Big Data (Big Data)*, Los Angeles, CA., USA., 2019, pp. 3820-3823.

[9] Wan-Ping Wu *et al.*, "Design and Implementation of Vehicle Speed Estimation Using Road Marking-based Perspective Transformation," in *2021 IEEE 93rd Vehicular Technology Conf. (VTC2021-Spring)*, Helsinki, Finland, 2021. pp. 1–5.

[10] Shalaka S. Wardha *et al.*, "Development of automated technique for vehicle speed estimation and tracking in video stream," in *2017 2nd IEEE RTEICT*, Bangalore, India, 2017, pp. 940–944, doi: 10.1109/RTEICT.2017.8256736.

[11] Genyuan Cheng *et al.*, "Real-Time Detection of Vehicle Speed Based on Video Image," in *2020 12th Int. Conf. on Measuring Technology and Mechatronics Automation*, Phuket, Thailand. Feb. 2020, pp. 313–317, doi: 10.1109/ICMTMA50254.2020.00076.

[12] B. Krishnakumar *et al.*, "Detection of Vehicle Speeding Violation using Video Processing Techniques," in *2022 ICCCI*, Coimbatore, India, 2022, pp. 01–07, doi: 10.1109/ICCCI54379.2022.9740909.

[13] Wencheng Wu *et al.*, "Vehicle speed estimation using a monocular camera," *Proc. of SPIE Video Surveillance and Transportation Imaging Applications*, vol. 9407, Mar. 2015, doi: 10.1117/12.2083394.

[14] A. El Bouziady *et al.*, "Vehicle speed estimation using extracted SURF features from stereo images," in *Int. Conf. on Intelligent Systems and Computer Vision*, Fez, Morocco, 2018, pp. 1–6, doi: 10.1109/ISACV.2018.8354040.

[15] Chenghuan Liu *et al.*, "A Vision-Based Pipeline for Vehicle Counting, Speed Estimation, and Classification," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 12, pp. 7547–7560, Dec, 2021, doi: 10.1109/TITS.2020.3004066.

[16] Jakub Sochor, Roman Juránek, Adam 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, Aug. 2017, doi: 10.1016/j.cviu.2017.05.015.

[17] D. Bell, W. Xiao, P. James, "Accurate vehicle speed estimation from monocular camera footage," in *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences 2020 XXIV ISPRS Congress (2020 edition)*, Nice, France, 2020, vol. V-2-2020, pp. 419–426.

[18] Héctor Rodríguez-Rangel *et al.*, "Analysis of Statistical and Artificial Intelligence Algorithms for Real-Time Speed Estimation Based on Vehicle Detection with YOLO," *Applied Sciences*, vol. 12, Issue 6, Mar. 2022, doi: 10.3390/app12062907.

[19] A. Cvijetić, S. Djukanović and A. Peruničić, "Deep learning-based vehicle speed estimation using the YOLO detector and 1D-CNN," in *27th Int. Conf. on Information Technology*, Zabljak, Montenegro, 2023, doi: 10.1109/IT57431.2023.10078518.

[20] A. Peruničić, S. Djukanović and A. Cvijetić, "Vision-based Vehicle Speed Estimation Using the YOLO Detector and RNN," in *27th Int. Conf. on Information Technology*, Zabljak, Montenegro, 2023, doi: 10.1109/IT57431.2023.10078639.

[21] Joseph Redmon and Ali Farhadi, "YOLOv3: An Incremental Improvement," 2018, arXiv:1804.02767.

[22] Nicolai Wojke, Alex Bewley, Dietrich Paulus, "Simple online and realtime tracking with a deep association metric," in *IEEE ICIP*, Beijing, China, Sep. 2017, doi:10.1109/ICIP.2017.8296962.

[23] Jianbo Shi and Carlo Tomasi, "Good Features to Track," in *Proc. of Computer Vision and Pattern Recognition*, Seattle, WA, USA, 1994, pp. 593-600, doi: 10.1109/CVPR.1994.323794.

[24] Jean-Yves Bouguet, "Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm," *Microprocessor Research Labs, Intel Corporation Microprocessor Research Labs*, USA, Rep. 2000.

[25] Herbert Bay, Tinne Tuytelaars, Luc Van Gool, "SURF: Speeded Up Robust Features," in *European Conf. on Computer Vision*, Berlin, Germany, 2006, pp. 404–417.

[26] Marius Muja, and David G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *Int. Conf. on Computer Vision Theory and Applications*, Lisboa, Portugal, 2009, pp. 331–340.

[27] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int. Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[28] Saleh Javadi, Mattias Dahl, Mats I. Pettersson, "Vehicle speed measurement model for video-based systems," *Computers&Electrical Engineering*, vol. 76, pp. 238–248, Jun. 2019, doi:10.1016/j.compeleceng.2019.04.001.

[29] Licheng Jiao *et al.*, "A Survey of Deep Learning-Based Object Detection," *IEEE Access*, vol. 7, pp. 128837–128868, 2019, doi: 10.1109/ACCESS.2019.2939201.

[30] Tsung-Yi Lin *et al.*, "Focal Loss for Dense Object Detection," 2017, arXiv:1708.02002.

[31] Jiyang Xie *et al.*, "Deep Learning-Based Computer Vision for Surveillance in ITS: Evaluation of State-of-the-Art Methods," *IEEE Trans. on*

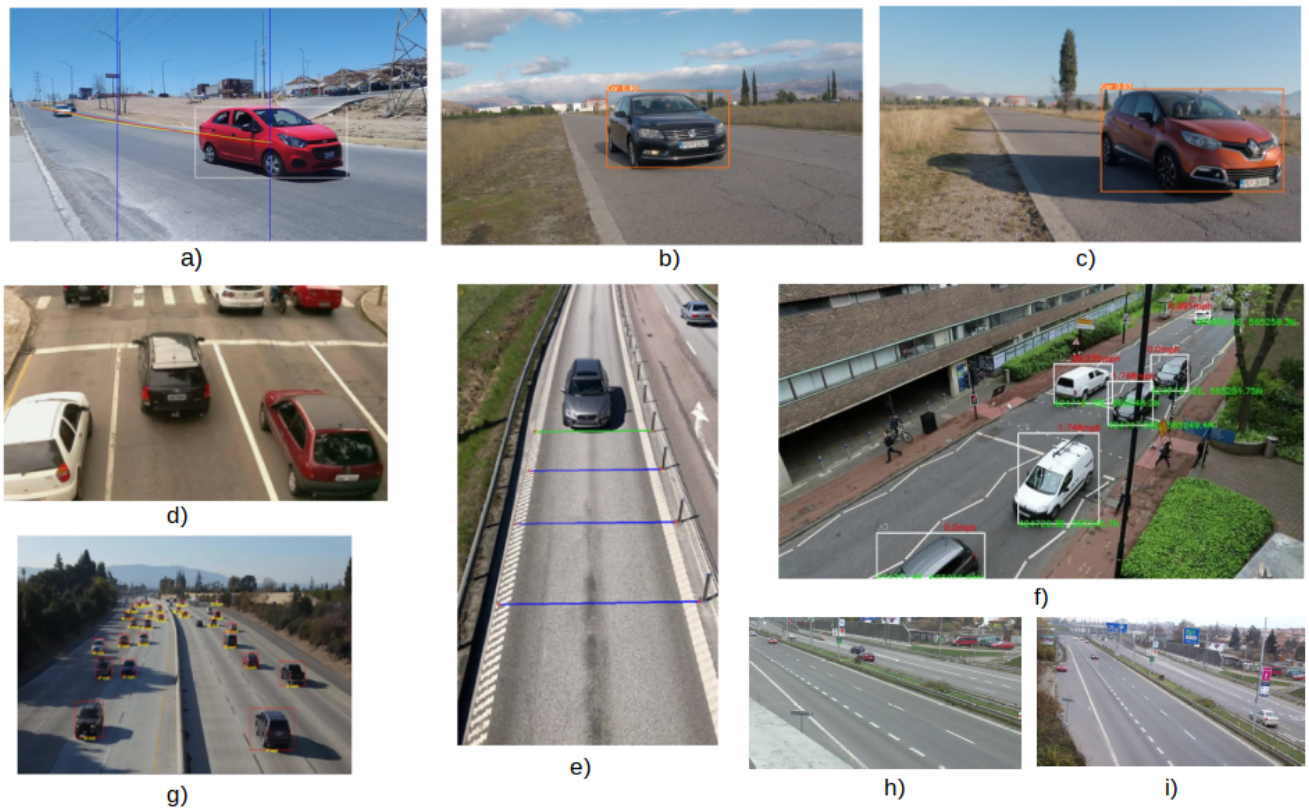

FIGURE 19. **FoV comparison among each study, a) [18], b) [19], c) [20], d) [7], e) [28], f) [17], g) [33], h) [44], i) [16].**

TABLE 6. **Comparison of speed estimation between start-of-the-art works and our proposed method.**

Author	Method		Estimation algorithm	MAE	RMSE	Conditions
	Detection	Tracking				
Héct. [18]	YOLOv3	Kalman filter	Linear regression	0.95 km/h	-	Zoom-side road at one vehicle.
Héct. [18]	YOLOv3	Kalman filter	Multilayer Perceptron	3.17 km/h	-	Zoom-side road at one vehicle.
Cvij. [19]	YOLOv5	-	1D-CNN	-	2.76 km/h	Zoom-side road at one vehicle.
Peru. [20]	YOLOv5	-	RNN	-	4.08 km/h	Zoom-side road at one vehicle.
Luvi. [7]	Background-Foreground	KLT	Linear model	-0.50 km/h	-	Top-down near FoV, Low speed 10 to 69 km/h.
Java. [28]	Moving pattern vectors	Intrusion lines	PDF of moving pattern vectors	1.77%		Top-down near Fov.
Bell. [17]	YOLOv2	SORT	Camera perspective	1.85 km/h	2.25 km/h	Top-down side near FoV, Low speed 1.6 to 15.8 km/h.
Amit. [33]	Mask-RCNN	SORT, DeepSORT	-	-	15.35 km/h	Similar to ours
Dong. [44]	3D ConvNets	Optical flow	3D ConvNets	2.71 km/h	14.62 km/h	Similar to ours
Soch. [16]	Faster-RCNN	Kalman filter	Linear model	1.10 km/h	-	Similar to ours, 3D model metadata of vehicles
Our	YOLOv3	DeepSORT	Linear model	3.38 km/h	4.69 km/h	Far FoV.
Our	YOLOv3	DeepSORT	Multilyer Perceptron	3.07 km/h	3.98 km/h	Far FoV.

Vehicular Technology, vol. 70, no. 4. pp. 3027–3042, Apr. 2021, doi: 10.1109/TVT.2021.3065250.

[32] Shaoqing Ren *et al.*, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," 2015, arXiv:1506.01497.

[33] Amit Kumar *et al.*, "A Semi-Automatic 2D solution for Vehicle Speed Estimation from Monocular Videos," in *2018 IEEE/CVF Conf. CVPRW*, Salt Lake City, UT, USA, Jun. 2018, doi: 10.1109/CVPRW.2018.00026.

[34] Alex Bewley *et al.*, "Simple online and realtime tracking," in *IEEE Int. Conf. on Image Processing*, Phoenix, AZ, USA, Sep. 2016, doi: 10.1109/ICIP.2016.7533003.

[35] Kaiming He *et al.*, "Mask R-CNN," 2017, arXiv:1703.06870.

[36] Joseph Redmon *et al.*, "You Only Look Once: Unified, Real-Time Object Detection," in *IEEE Conf. on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.

[37] Glenn Jocher, "YOLOv5 by Ultralytics," [Online]. Available: https://github.com/ultralytics/yolov5/releases.

[38] Kwang-Ju Kim *et al.*, "Multi-Scale Detector for Accurate Vehicle Detection in Traffic Surveillance Data," *EEE Access*, vol. 7, pp. 78311-78319, Jun. 2019, doi: 10.1109/ACCESS.2019.2922479.

[39] A. Horzyk and E. Ergün, "YOLOv3 Precision Improvement by the Weighted Centers of Confidence Selection, " in *Int. Joint Conf. on Neural Networks*, Glasgow, UK, 2020, pp. 1-8, doi: 10.1109/IJCNN48605.2020.9206848.

[40] M. A. Bin Zuraimi and F. H. Kamaru Zaman, "Vehicle Detection and Tracking using YOLO and DeepSORT," in *IEEE Symp. on Computer Applications & Industrial Electronics*, Penang, Malaysia, May. 2021, pp.

23–29, doi: 10.1109/ISCAIE51753.2021.9431784.

[41] Tausif Diwan *et al.*, "Object detection using YOLO: challenges, architectural successors, datasets and applications," *Multimedia Tools and Applications*, vol. 82, pp. 9243–9275, Aug. 2022, doi: 10.1007/s11042-022-13644-y

[42] U. Sirisha *et al.*, "Statistical Analysis of Design Aspects of Various YOLO-Based Deep Learning Models for Object Detection," *Int. Journal of Computational Intelligence Systems*, vol. 16, no. 126, Aug. 2023, doi: 10.1007/s44196-023-00302-w.

[43] Chien-Yao Wang *et al.*, "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors," in *2023 IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, Vancouver, BC, Canada, 2023, pp. 7464-7475, doi: 10.1109/CVPR52729.2023.00721.

[44] Huanan Dong, Ming Wen, and Zhouwang Yang, "Vehicle Speed Estimation Based on 3D ConvNets and Non-Local Blocks," *Future Internet in Big Data and Augmented Intelligence*, vol. 11, no. 6, pp. 78311–78319, May. 2019, doi: 10.3390/fi11060123.

KEATTISAK SANGSUWAN received the B.Eng and M.Eng degree in electronics engineering from King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand, in 2006, M.B.A from Sukhothai Thammathirat Open University Thailand, in 2009, and M.Sc. degree in Aeronautical and Space Systems (Embedded Systems) from Institut Supérieur de l'Aéronautique et de l'Espace, Toulouse, France, in 2014. He is currently pursuing Ph.D. degree in microelectronics and embedded systems with School of Engineering and Technology, Asian Institute of Technology, Thailand. His research interests include image processing, machine learning, and embedded systems.

MONGKOL EKPANYAPONG received the B.Eng. degree in computer engineering from Chulalongkorn Univerisity, Bangkok, Thailand, in 1997, the M.Eng. degree in computer science from the Asian Institute of Technology, Thailand, in 2000, and the M.Sc. and Ph.D. degrees in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2003 and 2006, respectively. From 1997 to 1998, he was a System Engineer with United Communication Network, Thailand. From 2006 to 2009, he was a Senior Computer Architect with Core 2 Architecture Design Team, Intel Corporation, USA. He joined the School of Engineering and Technology, Asian Institute of Technology, in 2009, where he is currently an Associate Professor. His research interests include VLSI design, physical design automation, microarchitecture, compiler, and embedded systems.