Proposing an algorithm using YOLOv4 and DeepSORT for tracking vehicle speed on highway

Phat Nguyen Huu, Manh Bui Duy
School of Electronics and Telecommunications, Hanoi University of Sceience and Technology (HUST), Vietnam

Article Info

Article historys:

Received Oct 10, 2021
Revised
Accepted

Keywords:

YOLOv4
DeepSORT
Object tracking
Tracking vehicle
Highways

ABSTRACT

Currently, expressways are increasingly developed and expanded. Several highways of Vietnam allow vehicles to travel up to 120 kilometers per hour helping to transport goods quickly and bring a lot of socio-economic benefits. Vehicle monitoring plays an important role in reducing traffic accidents helping to handle violations.The paper proposes a model to identify and monitor car speed on highways. The proposal method uses YOLOv4 combining with DeepSORT for vehicle identification and tracking. We then calculate the speed of car based on video recording and sending back from highway. The execution context is highway where vehicles move very fast. The results show that system meets set requirements with over 90% accuracy and execution times for up to 70 frames per second that is suitable for real systems.

Corresponding Author:

Phat Nguyen Huu,
School of Electronics and Telecommunications,
Hanoi University of Sceience and Technology (HUST),
01 Daicoviet Road, Hai Ba Trung district, Hanoi, Vietnam.
Email: phat.nguyenhuu@hust.edu.vn

1. INTRODUCTION

Currently, expressways are increasingly developed and expanded. Several highways of Vietnam allow vehicles to travel up to 120 kilometers per hour helping to transport goods quickly and bring a lot of socio-economic benefits. Vehicle monitoring plays an important role in reducing traffic accidents helping to handle violations. However, investment in researching an automatic monitoring system on highways is facing many difficulties. As a result, surveillance systems are performed by manual monitoring from video capturing and sending back on highway. Therefore, we will study the car speed detection and monitoring algorithm on highway that makes monitoring work easier in the paper.

The objective of the paper is to build a model to recognize and monitor vehicle speed on highways with video input recording on highway. The scope of model is a highway with only vehicles moving at fast speed. In proposal model, we change two blocks to determine correct velocity comparing to traditional model as follows:

- First, we use YOLOv4 to combine DeepSORT to identify and track vehicles. This is high-precision identification algorithm for context of express highways where objects move fast and need real-time processing.
- Secondly, we convert from pixel to distance (meter) and use the recycle track to solve the same vehicle problems that appear in consecutive frames.

The rest of the paper includes five parts and is organized as follows. Section 1 presents an overview of estimating the distance to the camera system. Section 2 presents several related works. Section 3 presents the proposal system. Section 4 will evaluate the proposal model and analyze the results. In final section, we give conclusions and future research directions.

2. RELATED WORK

On highways, vehicles need to be tracked and speed calculated [1-3]. There are several problems for existing systems [4-10].

1. Many objects to be tracked [5-12]:

Object tracking (OT) is the problem of tracking one or more moving objects of video. It is a higher level problem than object detection when object being processed is not simply single image. OT can be divided into two main approaches, namely single object tracking (SOT) and multiple object tracking (MOT).

SOT focuses on tracking a single object throughout video. To know which object to track, we need to have a bounding box. The details of this method are mentioned in [7-9]. MOT is geared towards more extensible applications. In MOT method, they try to simultaneously detect and track all objects including new objects appearing in video. Therefore, MOT is often more difficult problems than SOT and receives a lot of attention from researchers [10-12].

Besides, the methods of solving this class are also divided and popularized, namely online tracking and offline tracking [4, 13-17].

In online tracking, this method only uses current and previous frames for tracking while processing video. The treatment may reduce accuracy of algorithm. However, it reflects how the problem is handled in practice when “online” is essential. Several effective approaches have been proposed including sort [14] and DeepSORT [17]. In offline tracking, these methods usually use entire frame of video. Therefore, it often achieves much higher accuracy than online tracking. However, its computational cost is much higher [13, 17].

While monitoring vehicle speed, vehicles must be tracked from frame to others since object detection may not require continuous tracking. The Kalman filter [15] has been a reliable tool for solving object tracking tasks in many fields. The authors [4] has shown that simple tracker can be better than complex ones when objects are reliably detected.

2. Calculating velocity with multiple objects

Many studies have been performed for field of vehicle speed detection with different methods [18-20]. The authors [20] proposes an approach involving vehicle position comparison between current and previous frame to predict traffic speed from digital video capturing by fixing camera. Camera calibration is performed by applying geometric equations. The system is designed to be scalable to other application domains and has an average error of ±7 kilometers per hour for detecting vehicle speeds. The authors [18] uses motion parameters of image for conjunction with projection information between ground and image planes to obtain various metrics including vehicle speed. The method uses real-time monitoring techniques.

The authors [19] presented an approach based on detecting moving target of video by mapping relationship between pixel and actual distances. In the algorithm, three-frame bakground difference are used to extract features from moving vehicles.

In general, above existing methods have low accuracy and do not meet actual requirements in highways of Vietnam. In our knowledge, the proposal system with speed calculation and usage tracking using YOLOv4 combining with DeepSORT has not been mentioned.

3. PROPOSED SYSTEM

3.1. Overview of proposal system

Figure 1 is an overview of proposal system. Video through the pre-processor block obtains images that are input into recognition module. Id will be assigned to track object through tracking block after predicting label and location of object. Vehicles after identifying and assigning Id will be included in velocity block to calculate speed. Details of implementation blocks are as follows.

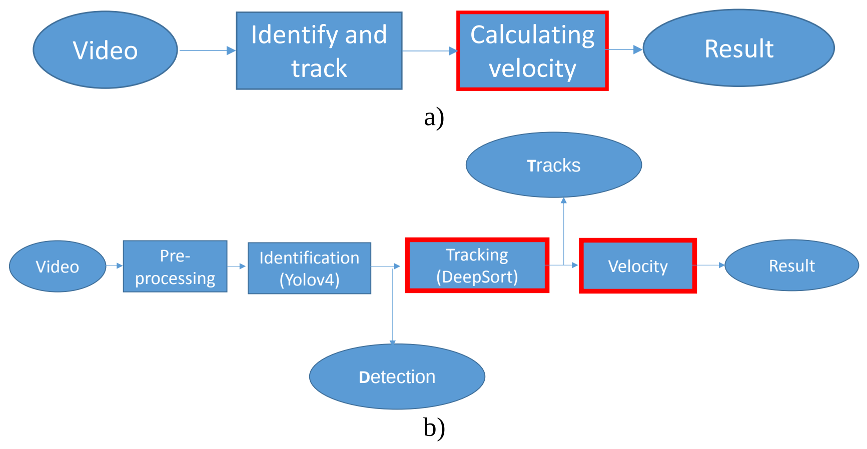


Figure 1. The proposal model (a) overview and (b) detailed implementation.

3.1.1 Identification block

In context of highway where the vehicles move very fast, it is necessary to use a recognition algorithm with fast processing speed and suitable accuracy. Therefore, we use YOLOv4 for identification module.

YOLOv4 is a recognition algorithm suitable for real-time recognition problems [21-24]. It strikes a balance between processing speed and accuracy. Besides, we can use YOLOv4-tiny, a stripped-down version of YOLOv4, with almost 10 times fewer parameters than original version. Table 1 is a chart comparing processing speed and accuracy of YOLO-v4 and YOLOv4-tiny with other recognition algorithms [25].

Table 1. Comparison of AP among YOLOv4 with other algorithms using MS COCO object detection.

FPS	ASFF*	EfficientDet	YOLOV3	YOLOV4
10	45	49	36	45.5
30	44	45	35	45
50	41	39	34	44
70	34	31	33	43.5
90	10	10	32.5	43
110	5	5	31.5	40
120	5	5	30	38

YOLO architecture includes the following blocks. Base network is convolutional and fully connected layers to extract featured. YOLOv4 uses backbone as cspdarknet35. In YOLOv4, neck block has function of getting rich informing for typical maps. The next part is extra layers applying to detect objects on characteristic map of base network. YOLO architecture is also quite diverse and able to customize for different input shapes.

The YOLO architecture includes base networks that are convolution networks to perform feature extraction. The following are extra layers applying to detect objects on feature map of base network. Base network uses mainly convolutional and fully connected layers. YOLO architectures are also quite diverse and can be customized into versions for different input shapes.

Darknet architechture component (base network) is used to extract features. Output of base network is a 7 × 7 × 1024 feature map that will be used as input for extra layers that predict label object and bounding box coordinates.

In YOLO version 3, the authors apply network feature extractor Darknet-53. The network consists of 53 consecutively connected convolutional layers and each of them is followed by batch normalization and leaky Relu activation. To reduce output size after each convolution layer, the authors downsample with filters of size 2. This help to reduce number of parameters for model.

Output of YOLO model is a vector that will include following components as

$$y=\left[p_o,\left(t_x,t_y,t_w,t_h\right),\left(p_1,p_2,\ldots,p_c\right)\right], \tag{1}$$

where p_0 is predicting probability that object will appear in bounding box. (t_x, t_y, t_w, t_h) help to define the bounding box where t_x, t_y are coordinates of center and t_w, t_h are width and length dimensions of bounding box. (p_1, p_2,... p_c) is predictive probability distribution vector of classes.

To evaluate accuracy of Yolo, loss function is used. It is divided into two parts: L_{loc} (localization loss) is used to measure error of bounding box and L_{cls} (confidence loss) is used to measure error of probability distribution of classes as

$$\mathcal{L}_{loc} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + \left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2, \tag{2}$$

$$\mathcal{L}_{cls} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} (1_{ij}^{obj} + \lambda_{noobj} \left(1 - 1_{ij}^{obj}\right)) \left(C_{ij} - \hat{C}_{ij}\right)^2 + \sum_{i=0}^{S^2} \sum_{c \in C} 1_i^{obj} \left(p_i(c) - \hat{p}_i(c)\right)^2, \tag{3}$$

$$\mathcal{L} = \mathcal{L}_{loc} + \mathcal{L}_{cls}. \tag{4}$$

More detail of architecture is shown in Fig. 2.

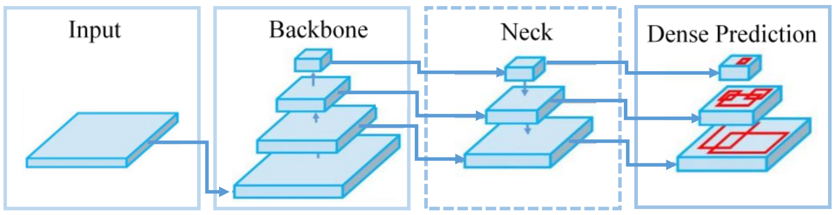


Figure 2. Yolov4 network structure [25, 26].

3.1.2 Tracking block

After identifying object with Yolo algorithm, we use DeepSORT algorithm [17] to perform monitoring and assigning IDs for vehicles. The method of monitoring object using DeepSORT includes following steps:

(1) Step 1: Using Yolo to detect objects of current frame.

(2) Step 2: Using Kalman filter to predict new tracks based on past ones. These statuses at the beginning will be assigned a tentative value. If the value is still guaranteed to be maintained in next three frames, the status will move from probe to confirmation status. It will try to maintain following 30 frames.

Otherwise, if losing signs when there is not enough three frames, the state will be deleted from tracking process.

(3) Step 3: Using confirming tracks, we take into matching cascade to link detection objects based on distance and characteristics.

(4) Step 4: Tracks and detections that have not been linked will be taken to a subsequent filter layer. We use Hungarian algorithm [27] to solve assignment problem with IOU cost matrix to second link.

(5) Step 5: Handling, classifying the detection and tracks

(6) Step 6: Using Kalman filter to correct value of track from detections that are linked to it and initialize new ones.

3.1.3 Velocity block

In the block, we rely on travel time of vehicle for fixing distance to determine its speed. More details is shown in Fig. 3.

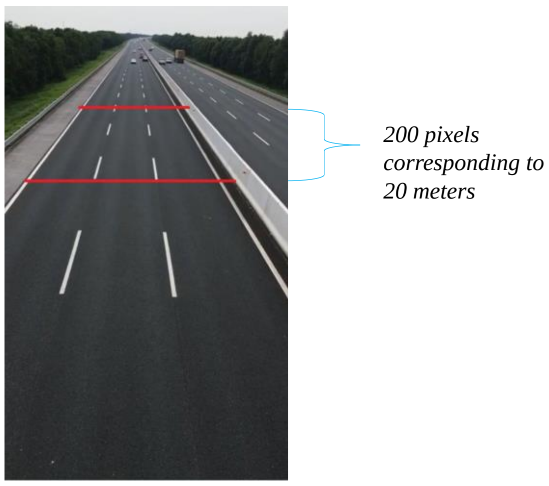


Figure 3. An example of determination of velocity based on fixing distance.

The distance *S* is determined in fact using main measurement method. It is distance between l_1 and l_2 lines of frame. The movement time between l_1 and l_2 is calculated by the formula:

$$t = \frac{N}{FPS}, \tag{5}$$

where *t* is time to move between l_1 and l_2 and *N* is number of frames that move from l_1 and l_2, and FPS (frame per second) is the number of frames within one second.
Since we calculates vehicle speed v of formula:

$$v = \frac{s}{t}. \tag{6}$$

The most difficult of block is that we must rely on actual measurement and video to determine how many pixels on the image corresponding to actual one meter.

3.2. Recycle track

Deleting track after 30 frames in section 3.1 will result in vehicles of same shape appearing in consecutive frames that causes ID vehicle to be mistaken. Therefore, we use a solution calling recycle track to discard tracks after being identified and display velocity to minimize errors and free up memory for them.

In [27], life-cycle management of track will include 3 states, namely probing, confirming, and deleting steps, as shown in Fig. 5. The process is as follows:

(1) These states are initially assigned an exploratory value (tentative).

(2) If this value is still guaranteed to be maintained for next three frames, state will change from probe to confirmed.

(3) Tracks with confirmed status will try to stay tracked. Deep SORT will still maintain tracking for next 30 frames even if they disappear.

(4) Otherwise, status will be deleted from tracker if track is lost when less than three frames.

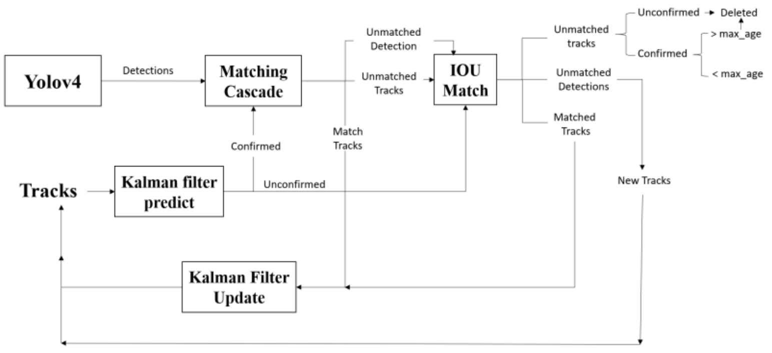


Figure 4. Process flow diagram of DeepSORT [14].

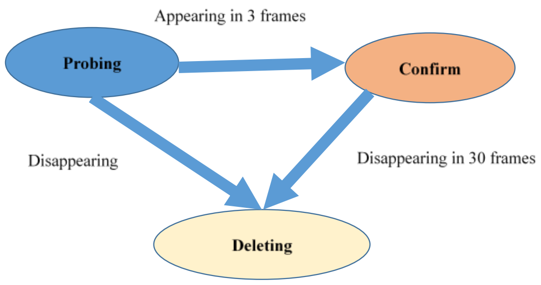


Figure 5. Managing track life-cycle in DeepSORT.

The processing flow of DeepSORT is performed sequentially as shown in Fig. 4 through the following steps:

(1) Step 1: Using faster region CNN (with backbone VGG16) to detect objects of current frame,

(2) Step 2: DeepSORT uses Kalman filter to predict new track states based on past ones. These states are initially assigned a tentative value. If this value is still guaranteed to be maintained for next three frames, it will change from probe to confirmed and will try to stay tracked for next 30 frames. Otherwise, it will be removed from tracker if the track is lost when less than three frames are reached,

(3) Step 3: Using verified tracks and performing matching cascade to associate detections based on their distance and feature metrics,

(4) Step 4: Unlinking tracks and detections will be passed to next filter layer. Using Hungarian algorithm to solve assignment problem with IOU cost matrix to link,

(5) Step 5: Processing and classifying detections and tracks,

(6) Step 6: Using Kalman filter to recalibrate value from detections associating with track and initialize new ones.

DeepSORT has improved problem of SORT with association strategy as well as using of appropriate metrics. The number of switches ID is reduced from 1423 to 781 that is a 45% reduction and also reduces errors relating to objects being obscured or disappearing. Although processing speed is slightly reduced, DeepSORT still ensures approximately real-time speed with GPU.

Details of recycle track process are shown in Fig. 6.

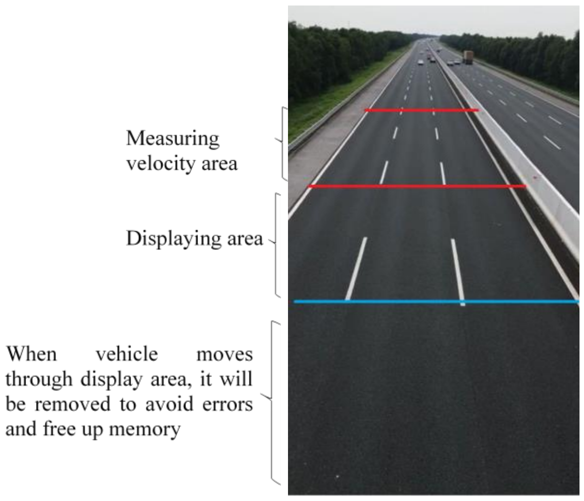

Measuring velocity area
Displaying area
When vehicle moves through display area, it will be removed to avoid errors and free up memory

Figure 6. Details of steps of recycle track process.

3.3. Prediction model

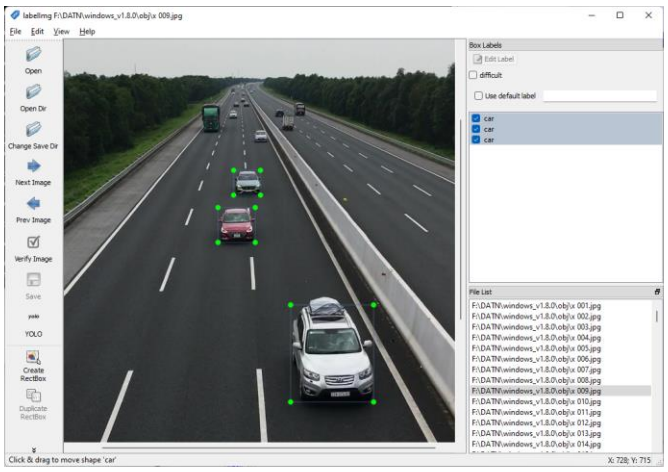
3.3.1. Preparing data

Data using in the paper consists of 3000 images cut from videos taken on highway as shown in Fig. 7. This is the data that we built ourselves.

Figure 7. Images of vehicles on highway are taken by ourselves.

3.3.2. Data labeling

In the step, we perform ROI block determination based on manual labeling. In the paper, we use an available tool calling labeling. The process is basically drawing boxes around object of image. Figure 8 shows an example using LabelImg tool that automatically creates a ".txt" file describing location of object in image.


labelImg F:\DATN\windows_v1.8.0\obj\x 009.jpg
File Edit View Help
Open
Open Dir
Change Save Dir
Next Image
Prev Image
Verify Image
Save
YOLO
Create RectBox
Duplicate RectBox
Box Labels
Edit Label
difficult
Use default label
car
car
car
File List
F:\DATN\windows_v1.8.0\obj\x 001.jpg
F:\DATN\windows_v1.8.0\obj\x 002.jpg
F:\DATN\windows_v1.8.0\obj\x 003.jpg
F:\DATN\windows_v1.8.0\obj\x 004.jpg
F:\DATN\windows_v1.8.0\obj\x 005.jpg
F:\DATN\windows_v1.8.0\obj\x 006.jpg
F:\DATN\windows_v1.8.0\obj\x 007.jpg
F:\DATN\windows_v1.8.0\obj\x 008.jpg
F:\DATN\windows_v1.8.0\obj\x 009.jpg
F:\DATN\windows_v1.8.0\obj\x 010.jpg
F:\DATN\windows_v1.8.0\obj\x 011.jpg
F:\DATN\windows_v1.8.0\obj\x 012.jpg
F:\DATN\windows_v1.8.0\obj\x 013.jpg
F:\DATN\windows_v1.8.0\obj\x 014.jpg
Click & drag to move shape 'car'
X: 728; Y: 715
Figure 8. An example of data labeling.

The operations performing as shown in Fig. 9 are based on [28]. After assigning data labels, we divide them into train/test files and create configuration file of parameters and upload it to drive.Model is next trained by Googlecolab. Finally, the values are put into model for evaluation.

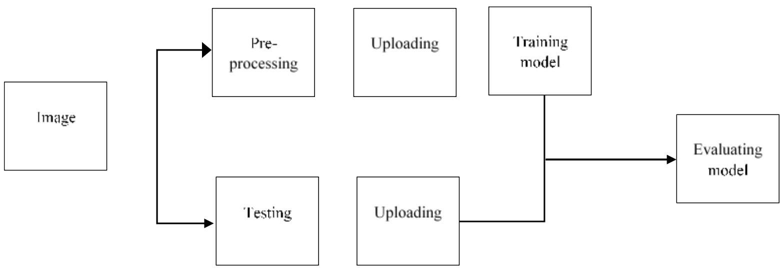


Figure 9. Detailed model of labeling implementation.

After training model, we proceed to identify and track vehicle. The output of identification process includes Id, bbox, and frame. They are input data to velocity module. The results of training process are shown in Tab. 2.

Table 2. Result of training model.

Type	Filters	Size	Input	Output
0 conv	16	3 × 3/1	416 × 416 ×3	416 × 416 × 16 0.150 BF
1 max		2 × 2/2	416 × 416 × 16	208 × 208 × 16 0.003 BF
2 conv	32	3 × 3/1	208 × 208 × 16	208 × 208 × 32 0.399 BF
3 max		2 × 2/2	208 × 208 × 32	104 × 104 × 32 0.001 BF
4 conv	64	3 × 3/1	104 × 104 × 32	104 × 104 × 64 0.399 BF
5 max		2 × 2/2	104 × 104 × 64	52 × 52 × 64 0.001 BF
6 conv	128	3 × 3/1	52 × 52 × 64	52 × 52 × 128 0.399 BF
7 max		2 × 2/2	52 × 52 × 128	26 × 26 × 128 0.000 BF
8 conv	256	3 × 3/1	26 × 26 × 128	26 × 26 × 256 0.399 BF
9 max		2 × 2/2	26 × 26 × 256	13 × 13 × 256 0.000 BF

10 conv	512	3 × 3/1	13 × 13 × 256	13 × 13 × 512 0.399 BF
11 max		2 × 2/1	13 × 13 × 512	13 × 13 × 512 0.000 BF
12 conv	1024	3 × 3/1	13 × 13 × 512	13 × 13 × 1024 1.595 BF
13 conv	256	1 × 1/1	13 × 13 × 1024	13 × 13 × 256 0.089 BF
14 conv	512	3 × 3/1	13 × 13 × 256	13 × 13 × 512 0.399 BF
15 conv	18	1 × 1/1	13 × 13 × 512	13 × 13 × 18 0.003 BF
16 yolo				
17 route	13			
18 conv	128	1 × 1/1	13 × 13 × 256	13 × 13 × 128 0.011 BF
19 upsample		2 ×	13 × 13 × 128	26 × 26 × 128
20 route	19			
21 conv	256	3 × 3/1	26 × 26 × 384	26 × 26 × 256 1.196 BF
22 conv	18	1 × 1/1	26 × 26 × 256	26 × 26 × 18 0.006 BF
23 yolo				
Total BFLOPS = 5.448				
Learning rate =0.001				
Momentum = 0.9				
Decay = 0.0005				

4. SIMULATION AND RESULT

4.1. Setup

We use videos taken on any highway to evaluate accuracy of tracking model and calculate vehicle speed in the paper. For tracking identification module, we perform as following

(1) Using two modules yolov4 + DeepSORT, yolov4-tiny + DeepSORT,

(2) Evaluating of accuracy and performance of model.

For velocity calculation module, we evaluate the results through two methods as follows:

(1) Determine velocity for every tracking vehicle of frame [29, 30],

(2) Determine vehicle speed to be tracked in specified location in proposal method.

4.2. Result

4.2.1. Tracking module

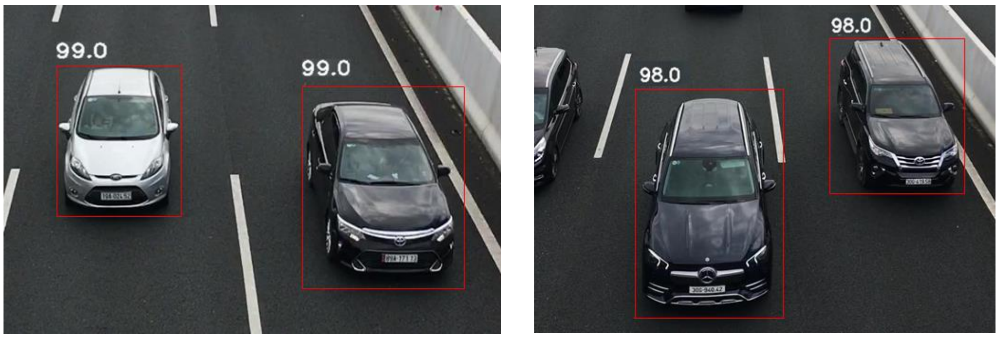
Firstly, we perform for identification and tracking module on highway. The results are shown in Figs. 10 and Tabs. 3 and 4.

Table 3. Result of frame rate of yolov4-tiny+DeepSORT.

Track ID	Class	BBox Coord (xmin, ymin, xmax, ymax)
10	car	(1123, 445, 1402, 660)
27	car	(603, 252, 762, 362)
29	car	(350, 232, 521, 364)
FPS =41.98		
Frame number = 262		
10	car	(1133, 452, 1404, 663)
27	car	(608, 252, 766, 361)
29	car	(358, 229, 534, 365)
FPS =40.94		
Frame number = 263		
10	car	(1140, 460, 1405, 665)
27	car	(611, 252, 768, 362)
29	car	(361, 227, 541, 367)
FPS =41.48		
Frame number = 264		

Table 4. Comparing accuracy of Yolov4 and Yolov4-tiny.

Critical	Yolov4 + DeepSORT	Yolov4-tiny + DeepSORT
Scene	Good weather, bright enough	Good weather, bright enough
Training time	30 hours	3 ÷ 4 hours
Video length	5 minutes	5 minutes
Accuracy	93 ÷ 99%	90 ÷ 98%
Processing speed	7 ÷ 10 FPS	29 ÷ 70 FPS



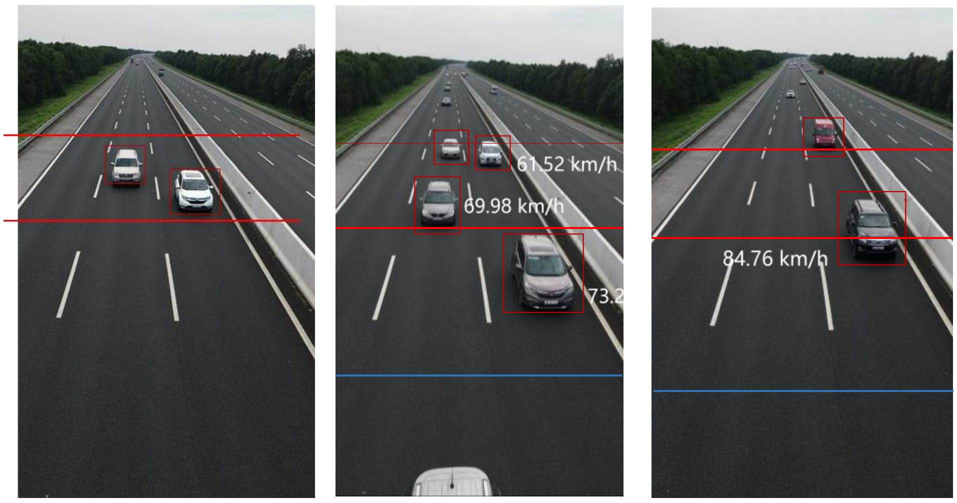
a) b)

Figure 10. Tracking results based on bounding box with (a) Yolov4 and (b) Yolov4-tiny.

From the results of Tab. 4, we see that Yolov4 + DeepSORT method has higher accuracy under same conditions. However, computation time is large when the Yolov4 tiny + DeepSORT method has fast computation time that is suitable for real-time applications. Besides, its accuracy is not significantly lower.

4.2.2. Velocity module

After using DeepSORT to track vehicles to calculate their speed, we get the result as shown in Fig. 11. In Fig. 11 (a), we show that the results do not match reality. In several cases, vehicle ID is mistaken leading to wrong speed because of not recycle track. In Fig. 11 (b), the results of model yolov4 give high accuracy with actual processing speed from 7 to 12 FPS.



a) b) c)

Figure 11. The results of calculating the speed achieved by model in three cases (a) without recycle track [29, 30], (b) Yolov4, and (c) Yolov4-tiny.

In Fig. 11, although the vehicles [29, 30] are updated with speed continuously for each frame, processing time is greatly increased due to have to save position of each vehicle of previous images. The second disadvantage is that distance of pixels is highly dependent on aspect ratio that leads to lower accuracy than selecting and measuring points. Besides, determining the time based on FPS for each frame leads to great dependence on processing speed. As a result, same video is performed twice times and produces two different results. In our proposal, processing time increases slightly and accuracy is higher due to pre-measured distance. The only disadvantage is limit on number of velocity updating in frame.

Table 5. Comparing accuracy with [29, 30].

Critical	Proposal	[29, 30]
Scene	Good weather, bright enough	Good weather, bright enough
Video length	5 minutes	5 minutes
Accuracy	93 ÷ 99%	<50%
Processing speed	30 ÷ 70 FPS	<7 FPS

Besides, we also evaluate the performance and processing time of proposal system with [29, 30] on computers with core I5 configuration and 16 GB RAM. The results are shown in Tab. 5. We see that the system achieves an accuracy of over 90% with execution time up to 70 FPS that is suitable for speed monitoring applications.

5. CONCLUSION

The paper focuses on using of neural networks for recognizing and tracking speed of vehicles moving on highways. In the paper, we have identified vehicles on highway with an accuracy of over 90%. The determining speed is consistent with reality and processing time of model is low (20 Fps for computers without VGA and up to 70 Fps for I5 configuration 9400 ram 16Gb using VGA 1050Ti). However, speed tracking and monitoring only achieves high results in bright enough daytime weather conditions.

Therefore, we will improve the model in bad weather conditions at night to increase accuracy of model in all cases the next direction. In addition, the future will aim to build a model capable of calculating vehicle speed at every position of frame.

ACKNOWLEDGMENTS

This research is carried out in the framework of the project funded by the Ministry of Education and Training (MOET), Vietnam under the grant B2020-BKA-06. The authors would like to thank the MOET for their financial support.

REFERENCES

[1] S. Hua, M. Kapoor, and D. C. Anastasiu, "Vehicle tracking and speed estimation from traffic videos," in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2018, pp. 153–1537.

[2] B. Alefs and D. Schreiber, "Accurate speed measurement from vehicle trajectories using adaboost detection and robust template tracking," in 2007 IEEE Intelligent Transportation Systems Conference, 2007, pp. 405–412.

[3] L. Lou, Q. Zhang, C. Liu, M. Sheng, Y. Zheng, and X. Liu, “Vehicles detection of traffic flow video using deep learning,” in 2019 IEEE 8th Data Driven Control and Learning Systems Conference (DDCLS), 2019, pp. 1012–1017.

[4] E. Bochinski, V. Eiselein, and T. Sikora, “High-speed tracking-by-detection without using image information,” in 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 2017, pp. 1–6.

[5] Z. Soleimanitaleb, M. A. Keyvanrad, and A. Jafari, “Object tracking methods:a review,” in 2019 9th International Conference on Computer and Knowledge Engineering (ICCKE), 2019, pp. 282–288.

[6] L. Fan, Z. Wang, B. Cail, C. Tao, Z. Zhang, Y. Wang, S. Li, F. Huang, S. Fu, and F. Zhang, “A survey on multiple object tracking algorithm,” in 2016 IEEE International Conference on Information and Automation (ICIA), 2016, pp. 1855–1862.

[7] A. Ellouze, M. Ksantini, F. Delmotte, and M. Karray, “Single object tracking applied to an aircraft,” in 2018 15th International Multi-Conference on Systems, Signals Devices (SSD), 2018, pp. 1441–1446.

[8] R. Seth, S. Kumar Swain, and S. Kumar Mishra, “Single object tracking using estimation algorithms,” in 2018 2nd International Conference on Power, Energy and Environment: Towards Smart Technology (ICEPE), 2018, pp. 1–6.

[9] B. Mocanu, R. Tapu, and T. Zaharia, “Single object tracking using offline trained deep regression networks,” in 2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA), 2017, pp. 1–6.

[10] M. Han, A. Sethi, W. Hua, and Y. Gong, “A detection-based multiple object tracking method,” in 2004 International Conference on Image Processing, 2004. ICIP ’04. vol. 5, 2004, pp. 3065–3068.

[11] S. Sun, N. Akhtar, H. Song, A. Mian, and M. Shah, “Deep affinity network for multiple object tracking,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 43, no. 1, pp. 104–119, 2021.

[12] W. Li, J. Mu, and G. Liu, “Multiple object tracking with motion and appearance cues,” in 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), 2019, pp. 161–169.

[13] F. Du, B. Xu, J. Tang, Y. Zhang, F. Wang, and H. Li, “1st place solution to eccvtao-2020: Detect and represent any object for tracking,” 2021.

[14] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, “Simple online and realtime tracking,” in 2016 IEEE International Conference on Image Processing (ICIP), 2016, pp. 3464–3468.

[15] K. R.E., “A new approach to linear filtering and prediction problems,” Journal of basic Engineering, vol. 82, no. 1, pp. 35–45, 1960.

[16] J. Tang, X. Xiong, C. Xie, Y. Zhang, P. Wang, F. Wang, F. Du, L. Han, Y. Zheng, P. Pan, and H. Li, “Min-cost network flow and trajectory fix for multiple objects tracking,” in In Conference on Computer Vision and Pattern Recognition Workshop, 2020, pp. 1–4.

[17] 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), 2017, pp. 3645–3649.

[18] N. Ferrier, S. Rowe, and A. Blake, “Real-time traffic monitoring,” in Proceedings of 1994 IEEE Workshop on Applications of Computer Vision, 1994, pp. 81–88.

[19] Z. Tang, G. Wang, T. Liu, Y.-G. Lee, A. Jahn, X. Liu, X. He, and J.-N. Hwang, “Multiple-kernel based vehicle tracking using 3d deformable model and camera self-calibration,” 2017.

[20] A. Gholami, A. Dehghani, and M. Karim, “Vehicle speed detection in video image sequences using cvs method,” International Journal of Physical Sciences, vol. 5, pp. 2555–2563, 12 2010.

[21] M. Maity, S. Banerjee, and S. Sinha Chaudhuri, “Faster r-cnn and yolo based vehicle detection: A survey,” in 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), 2021, pp. 1442–1447.

[22] C. Gao, Q. Cai, and S. Ming, “Yolov4 object detection algorithm with efficient channel attention mechanism,” in 2020 5th International Conference on Mechanical, Control and Computer Engineering (ICMCCE), 2020, pp. 1764–1770.

[23] A. K. Shetty, I. Saha, R. M. Sanghvi, S. A. Save, and Y. J. Patel, “A review: Object detection models,” in 2021 6th International Conference for Convergence in Technology (I2CT), 2021, pp. 1–8.

[24] K. Li and L. Cao, “A review of object detection techniques,” in 2020 5th International Conference on Electromechanical Control Technology and Transportation (ICECTT), 2020, pp. 385–390.

[25] A. Bochkovskiy, C.-Y. Wang, and H.-y. Liao, “Yolov4: Optimal speed and accuracy of object detection,” pp. 1–17, 04 2020.

[26] P. Adarsh, P. Rathi, and M. Kumar, “Yolo v3-tiny: Object detection and recognition using one stage improved model,” in 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), 2020, pp. 687–694.

[27] R. Sanchez-Matilla, F. Poiesi, and A. Cavallaro, “Online multi-target tracking with strong and weak detections,” vol. 9914, 10 2016, pp. 84–99.

[28] R. Phadnis, J. Mishra, and S. Bendale, “Objects talk - object detection and pattern tracking using tensorflow,” in 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), 2018, pp. 1216–1219.

[29] J. Lmer, D. Cymbalak, and F. Jakab, “Computer vision based object recognition principles in education,” in 2013 IEEE 11th International Conference on Emerging eLearning Technologies and Applications (ICETA), 2013, pp. 253–257.

[30] A. Rosebrock, “Simple object tracking with opencv,” July 23, 2018, retrieved from official website: https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/.

BIOGRAPHY OF AUTHORS

	Phat Nguyen Huu received his B.E. (2003), M.S. (2005) degrees in Electronics and Telecommunications at Hanoi University of Sceience and Technology (HUST), Vietnam, and Ph.D. degree (2012) in Computer Science at Shibaura Institute of Technology, Japan. Currently, he lecturer at School of Electronics and Telecommunications, HUST Vietnam. His research interests include digital image and video processing, wireless networks, ad hoc and sensor network, and intelligent traffic system (ITS) and internet of things (IoT). He received the best conference paper award in SoftCOM (2011), best student grant award in APNOMS (2011), hisayoshi yanai honorary award by Shibaura Institute of Technology, Japan in 2012.
	Manh Bui Duy is student of Electronic and Telecommunications at Hanoi University of Science and Technology (HUST), Vietnam. Currently, he is working in Future Network lab. at HUST. His main duty is developing smart products which relates to digital image, video processing, machine learning and internet of things (IoT).