Signal, Image and Video Processing
https://doi.org/10.1007/s11760-022-02184-5

ORIGINAL PAPER

Adaptive multi-vehicle motion counting

Xuan-Duong Nguyen[1,2] · Anh-Khoa Nguyen Vu[1,2] · Thanh-Danh Nguyen[1,2] · Nguyen Phan[1,2] · Bao-Duy Duyen Dinh[1,2] · Nhat-Duy Nguyen[1,2] · Tam V. Nguyen[3] · Vinh-Tiep Nguyen[1,2] · Duy-Dinh Le[1,2]

Received: 17 July 2021 / Revised: 15 February 2022 / Accepted: 17 February 2022

Abstract
Counting multi-vehicle motions via traffic cameras in urban areas is crucial for smart cities. Even though several frameworks have been proposed in this task, there is no prior work focusing on the highly common, dense and size-variant vehicles such as motorcycles. In this paper, we propose a novel framework for vehicle motion counting with adaptive label-independent tracking and counting modules that processes 12 frames per second. Our framework adapts hyperparameters for multi-vehicle tracking and properly works in complex traffic conditions, especially invariant to camera perspectives. We achieved the competitive results in terms of root-mean-square error and runtime performance.

Keywords Object detection · Object tracking · Vehicle motion counting

1 Introduction

In worldwide metropolitan areas, traffic surveillance cameras are increasingly used to monitor daily activities of transportation. Exploiting such data allows administrations to find solutions towards mitigating traffic congestion. Among the considerable aspects, vehicle counting along the Motions of Interest (MoI) is a vital problem. Facing the problem, NVIDIA introduced a competition of vehicle counting [1]. The main vehicle types in the dataset are mainly four-wheel cars and freight trucks. To overcome the challenge, many teams proposed their frameworks as [2–4] which can work acceptably in the contest. However, the target objects are usually large vehicles, and the traffic density is relatively low, while Ho Chi Minh City AI Challenge[1] (HCMC AIC) focuses on Vietnamese traffic conditions with four types of vehicles such as motorcycles, cars, buses and trucks. Note that these vehicles move in the same lane and the motorcycles dominate the others in terms of density, population and diversity. That is a popular traffic situation not only in Vietnam but also in other Southeast Asian countries in which motorbikes dominate the streets such as Laos, Thailand, Malaysia, and others.

To address the issues, our approach decomposes the vehicle counting problem into three main stages, namely vehicle detection, tracking and counting. In vehicle detection, we adopt object detector, i.e., YOLOv4 [7] with CSP (Cross Stage Partial Network) [8] to detect the four aforementioned

[1] Original: https://aichallenge.hochiminhcity.gov.vn, for English version: https://shorturl.at/wOPX3.

✉ Tam V. Nguyen
tamnguyen@udayton.edu

Xuan-Duong Nguyen
18520212@gm.uit.edu.vn

Anh-Khoa Nguyen Vu
khoanva@uit.edu.vn

Thanh-Danh Nguyen
danhnt@uit.edu.vn

Nguyen Phan
17520828@gm.uit.edu.vn

Bao-Duy Duyen Dinh
18520658@gm.uit.edu.vn

Nhat-Duy Nguyen
duynn@uit.edu.vn

Vinh-Tiep Nguyen
tiepnv@uit.edu.vn

Duy-Dinh Le
duyld@uit.edu.vn

[1] University of Information Technology, Ho Chi Minh City, Vietnam

[2] Vietnam National University, Ho Chi Minh City, Vietnam

[3] University of Dayton, Dayton, OH, USA

Published online: 26 April 2022

vehicle types. Additional data collection and data augmentation are used with aims at avoiding overfitting with the provided training set and being flexible with various weather conditions. Then, we leverage object tracker, i.e., DeepSORT [9,10], to track the detected objects in order to form the vehicle trajectory. Furthermore, to cope with the problem of vehicle occlusion and high speed velocity of vehicle, we tracked four types of objects separately with tuning hyperparameters of DeepSORT and proposed a trajectory association module as a post-processing step. Finally, we propose various counting algorithms to achieve the goals of vehicle motion counting. The main idea of our methods is assigning trajectories to MoIs if they are similar in direction. Among the three algorithms, we focus on counting vehicle by majority points voting, buffer-based counting and perspective invariant method consecutively. Our system is evaluated on the HCMC AIC testset. Experimental results demonstrate the effectiveness of our proposed work. In this paper, our contributions are fourfold:

- We propose a novel framework for vehicle motion counting system with adaptive label-independent tracking and counting modules.
- We introduce a method that adapts hyperparameters for multi-vehicle tracking.
- We propose two MoI-based vehicle motion counting algorithms that work in complex traffic conditions, especially invariant to camera perspectives.
- Our system achieves the competitive results in both the preliminary round and the final round of HCMC AIC.

2 Related work

Generally, the existing vehicle counting systems can be classified into two categories, namely *vehicle counting* and *vehicle motion counting*. Vehicle counting, a simple problem of vehicle motion counting, aims to count vehicles moving on roads regardless of their moving directions. Specifically, prior works [4,11–13] used density-aware approaches. These works leverage feature extractors such as deep learning or handcrafted features to create feature maps for describing the number of objects in the image. Liang et al. [14] utilized regression analysis for counting and classifying vehicles. The benefit is that the explicit segmentation ground truth and the real trajectory of the vehicle are not required. However, it is unable to detect vehicles in complex conditions like urban traffic or inclement weather conditions. To deal with various conditions, Kamkar [15] proposed a vehicle detection method that selects vehicles using an active basis model and their reflection symmetry. Then, they counted and classified the vehicle by extracting two features: vehicle length regarding time-spatial image and the correlation computed from the grey-level co-occurrence matrix of the bounding boxes of each vehicle. Seenouvong et al. [16] proposed using background subtraction technique to find objects in the foreground in a sequence, yet cannot handle occlusion cases. Furthermore, the size of the region of interest (RoI) should be large enough for counting vehicle insides. Later, Xiang et al. [17] classified moving objects by dividing their status into two situations, i.e., static background and moving background. For each type of background, they designed particular methods to detect objects. Basically, these mentioned works, using the density-aware approaches, only approximately predict the number of objects, not directly count them. Many researchers tend to use detection-aware approaches with higher accuracy when assigning vehicles to a specific movement of interest. Regarding AI City Challenge, we found several works addressing this problem, mostly based on a Detection–Tracking–Counting (DTC) pipeline. Liu et al. [4] proposed a framework for movement-specific vehicle counting using Faster R-CNN [18] as the detector, Deep-SORT [9,10] as the tracker, and an algorithm that requires a typical trajectory of each movement. A so-called system iTASK [5] proposed to track real-time vehicles moving along the desired direction with many features like real-time vehicle flow counting, vehicle re-identification, anomaly detection Bui et al. [6] proposed a distinguished region tracking approach for counting multiple vehicles in complicated motions of interest. While Ospina et al. [3] proposed a system using tracking without bells and whistles [19] and tuning parameters manually. Their system is consequently not able to differentiate vehicle types, they classified them by calculating the area of vehicles. Table 1 shows the comparison between the mentioned systems and ours. Taking the advantages of prior works, we proposed a vehicle motion counting system that combined

Table 1 Comparison between prior works and our proposed method

Method	Zero-VIRUS [2]	Countor [3]	Liu et al. [4]	iTASK [5]	Bui et al. [6]	Ours
Adaptive				✓		✓
End-to-end inference	✓	✓	✓	✓	✓	✓
Real-time performance					✓	✓
Low error rate	✓	✓	✓	✓	✓	✓

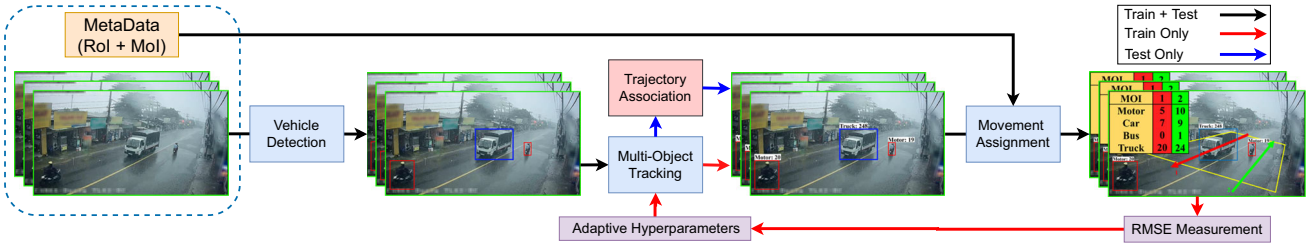


Fig. 1 Overview of our proposed adaptive multi-vehicle motion counting framework

the density-aware and detection-aware approaches and additional techniques to remedy the problems that the earlier works could not well handle.

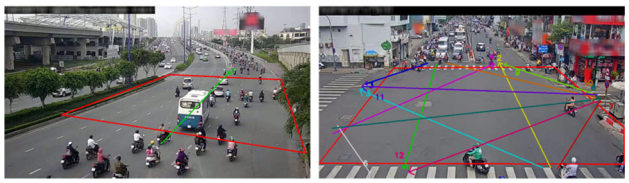
3 Proposed framework

3.1 Adaptive detection–tracking–counting overview

Figure 1 shows the overview of our vehicle motion counting framework. Video frames are first fed into the detection module, i.e., Yolov4 with CSP. Then, we obtain a list containing bounding boxes and vehicle labels belong to these boxes for each frame. DeepSORT works as our main tracking module. We separately track every vehicle type for each individual video with the *adaptive weights of hyperparameters*. We also proposed a trajectory association module to address the ID switches. From the outcomes of tracking, for each bounding box, we retrieve its particular tracking ID. The counting module considers bounding boxes having the same tracking IDs as a trajectory. Finally, depending on video characteristics of the spatially identical geometry, we propose three algorithms to accurately find a MoI for trajectories. Our system adaptively works on many contexts by overcoming detection issue via data augmentation with various conditions. Then the tracking and counting modules perform on any types of vehicles.

3.2 AIC HCMC testsets

There are 2 respective testsets, A and B, for the preliminary round and the final round. The organizers only released testset A (without groundtruth) for the preliminary round. The motion counting groundtruth is not provided for both testsets. Regarding Testset A, each video is about 22 minutes 30 seconds long (10 FPS), totally 13,500 frames per video and 4 types of vehicles such as motorcycles, cars, buses and trucks. Figure 2 shows some exemplary scenes in testset A.



(a) Camera 09 **(b)** Camera 10

Fig. 2 Examples of RoI and MoI on some cameras. The red quadrangles describe for the RoI, the numbered vectors describe for the MoIs

3.3 Detection

We first used a pretrained model on COCO to leverage the presence of the four types of vehicles in HCMC AIC. However, we found that it did not perform well due to the following problems. Firstly, there is a different distribution data of vehicle appearances caused by perspectives between COCO and HCMC AIC dataset. For example, the pretrained model seems to accurately detect the vehicles passing the roads, but miss ones moving toward or far away from camera. Secondly, occlusion is a major factor, which prevents the model from detecting the vehicle especially in narrow roads with a variety of commuters. Multiple scales are another contributing to problems. This makes the detector fail to detect means of transportation in videos. Various scales combined with occluded objects that should be under the consideration as a tough issue for pretrained models. Noted that, Yolov4 with CSP was selected as our detector due to its superiority in comparison with Yolov3 and Faster R-CNN in this challenging scenario. Section 4.3.1 explains it in greater details.

Dataset Collection and Data Augmentation To overcome the mentioned problems in Sect. 3.3, we collected a new dataset to retrain the detector. Our training data was collected from HCMC AIC sample dataset[2] and external sources such as traffic videos from various Vietnam cities including Ho Chi Minh City and Da Nang city. It should be noted that, the total number of cameras in HCMC AIC sample dataset is 25. Moreover, the conditions are categorized into five categories: clear day, clear night, rainy day, rainy night, and black-and-white corresponding to 10, 5, 4, 3, and 3

[2] https://github.com/hcmcaic/ai-challenge-2020.

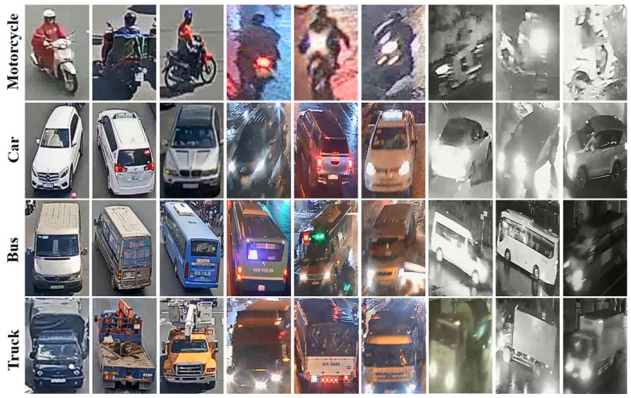


Fig. 3 Exemplary bounding boxes of four types of vehicle: motorcycle, car, bus, and truck

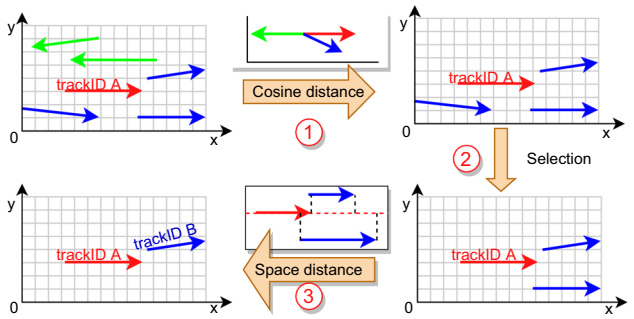


Fig. 4 Overview of our trajectory association module. We have 3 steps to find a new tracking ID B to associate with the mistracking ID A (red vector): (1) calculate the cosine similarity among the considered vectors, (2) select suitable vectors and (3) choose the new tracking ID B by measuring space distance

cameras, respectively. In particular, our annotated detection dataset had 3697 labeled images with 11,530, 5232, 1163 and 2320 bounding boxes for motorcycles, cars, buses and trucks, respectively. Figure 3 shows examples of the four types of vehicle. After several augmentation methods such as horizontal flipping, blurring, etc., our dataset included 27,381 annotated images in total and is divided into three subsets of 19,166, 2739, 5476 images for training, validating and testing, respectively. We refer it as *ADTC27K dataset*

3.4 Tracking

In this tracking component, we used DeepSORT [9,10] as the main tracker and proposed a trajectory association module as a post-processing stage to handle the ID switches problem. This ID switch error is often caused by the overlapping or occlusion among vehicles in high density. For example, in camera 10, when hundreds of motorcycles come across each other in the intersection, the track ID of those motorcycles could be unstable. We observed three problems when applying the pre-trained pedestrian DeepSORT [9,10] on our collected data: (i) Tracker assigns wrong labels of objects when tracking four types of objects together, (ii) Missing tracking ID of objects, especially small objects like motorcycles, (iii) DeepSORT model has many default hyperparameters with not suitable values. To figure out values of hyperparameters which best fit the DeepSORT tracker, we performed Grid Search and then Random Search to choose the appropriate ones.

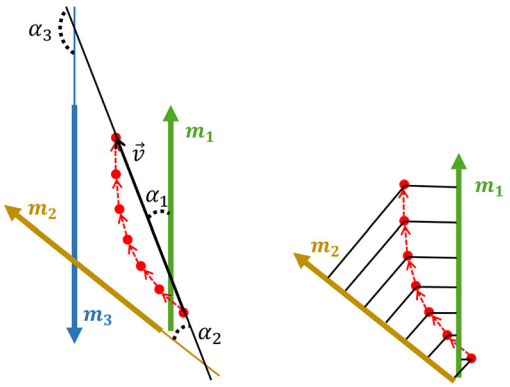
Trajectory association Although the system performance is improved through the previous steps, some cases of ID switch errors are still caused by the data attributes. To tackle the issue, we introduce a post-tracking step named as *Trajectory Association module* (shown in Fig. 4). The goal of the module is to find a new tracking ID for each mistracking ID of the vehicles and unite both of them. To associate the trajectories of a vehicle, we define the vehicle having the mistracking ID as the vehicle whose last bounding box missed in *RoI*. We suppose that the new tracking ID is the nearest mistracking ID in both spatial and temporal dimensions. In other words, a suddenly appeared tracking ID which is considered as a new tracking ID, must be in the same direction and the closest distance to the mistracking ID. Therefore, our proposed module based on that two criteria.

We consider the *mistracking ID A* whose last bounding box missed in *RoI* and the *new tracking ID B* whose first bounding box suddenly appeared in *RoI*. Given $P_A = \{p_{(A,i)}\}_{i=1}^{\Psi}$ and $P_B = \{p_{(B,j)}\}_{j=1}^{\Omega}$ are trajectories of A, B and $\overrightarrow{v_A} = p_{(A,\Psi)} - p_{(A,\Psi-1)}$, $\overrightarrow{v_B} = p_{(B,2)} - p_{(B,1)}$, $\overrightarrow{v_1} = p_{(B,1)} - p_{(A,\Psi)}$, $\overrightarrow{v_2} = p_{(B,2)} - p_{(A,\Psi)}$. B has Ω bounding boxes and $p_{(B,j)} \in \mathbb{R}^2$ is the center point of the jth of ones. First we find a mistracking ID in the trajectory list of the video and vectorize it as vector $\overrightarrow{v_A}$ (red color). For each tracking ID which suddenly appears in RoI in subsequent frames, we vectorize and append it to a list as the potential vector $\overrightarrow{v_B}$. Specifically, we calculate the cosine similarity between $\overrightarrow{v_B}$ and $\overrightarrow{v_A}$ as d_1. Trajectories with d_1 smaller than a threshold θ_1 are eliminated. By this way, we select the tracking IDs which have almost same orientation as $\overrightarrow{v_A}$. Then, we collect missed-tracking IDs appeared after $\overrightarrow{v_A}$ if this condition is satisfied: $\overrightarrow{v_A} \cdot \overrightarrow{v_1} > 0 \vee \overrightarrow{v_A} \cdot \overrightarrow{v_2} > 0$. In the list of remaining candidates, we calculate the projective distance from a point of $\overrightarrow{v_A}$ to the line of $\overrightarrow{v_B}$ as d_2. We choose the track where d_2 is the smallest and is less than a certain threshold θ_2 to combine with $\overrightarrow{v_A}$. In HCMC AIC dataset, we empirically set hyperparameters θ_1, θ_2 to 0.8 and 100, respectively.

3.5 Counting

Bounding boxes and tracking ID of each vehicle from the previous tracker come in the counting module as inputs.

For each camera in the HCMC AIC dataset, we only count vehicles that pass over a region of interest (RoI) in specific motions of interest (MoIs). We denote a sequence of MoIs as



(a) Step 1: Eliminate unexpected MoIs

(b) Step 2: Calculate the distances

Fig. 5 Two core steps of PDP algorithm: measuring the angle between a trajectory and MoIs then eliminating MoIs whose angle greater than 45°; and calculating the distance of each point on the renew trajectory to MoIs. The trajectory is in the red, different MoIs are represented by other colors

$M = \{m_i\}_{i=1...k}$, where m_i is a vector and k is the number of MoIs (shown in Fig. 2). A vehicle is considered inside a RoI if the intersection over union (IOU) between its bounding box and the RoI greater than zero. Let $P = \{p_i\}_{i=1...n}$ is the trajectory of a vehicle, p_i is the center of the bounding box ith ($1 \leq i \leq n$), where n is the number of bounding boxes in RoI. The first point p_1 and the last point p_n are computed as: $\overrightarrow{v} = \overrightarrow{p_1 p_n} = p_n - p_1$, where $\overrightarrow{v}$ is the displacement in a straight line in the direction of the vehicle.

In this paper, we propose improvements to cope with a wide range of topographical features for spatially identical cameras. We first proceed a preprocessing step: unidentified or immobile objects are also removed from this set by remaining trajectories having at least ρ points ($n > \rho$) and eliminating ones consisting of $||\overrightarrow{v}|| < \gamma$ pixels. Based on γ, we determine whether a vehicle is moving or not. In the experiment, we set $\rho = 5$ points and $\gamma = 250$ pixels.

3.5.1 Points2MoI distance polling

Points2MoI distance polling (PDP)-based Counting Algorithm was used to count vehicles for all of videos in the first time. In PDP, we assigned one movement for each vehicle by considering the direction and measuring the distance between its trajectory and MoIs. First, for each trajectory, we only keep MoIs having the direction close to $\overrightarrow{v}$. To do this, we do not consider MoIs having the angle between $\overrightarrow{v}$ and itself greater than ϕ by calculating sequence $D = \{\alpha_1, \alpha_2, \ldots \alpha_k\}$ where α_i is the absolute of the angle between $\overrightarrow{v}$ and $m_i \in M$, respectively. Second, the trajectory of vehicle is equally divided into n' points. From each point, we calculate the Euclidean distance to each MoI, in case the projection of the considering point to a MoI did not belong to a segment of that MoI, the distance is the minimum of distance from the point to start and end point of the MoI. The point belongs to the nearest MoI. The chosen MoI would have the majority of n' points in the trajectory voted for. In our experiment, we set ϕ as 45° and n' as 32 points. Figure 5 shows the overview of these PDP steps.

3.5.2 PDP perspective transformation

Due to the camera perspective, two separate MoIs in reality are close to each other. To deal with this issue, we did a transformation called Perspective Transformation. We changed the perspective of video frame for getting better insights about the required information.

After transformation, we use our PDP algorithm to continue processing.

4 Evaluation

4.1 Evaluation metrics

Preliminary round In the preliminary round, each submission is scored by the effectiveness of counting the number of vehicles in types for each MoI across all videos. The accuracy score is calculated by following these steps: (1) split each video to k segments, each segment is Δ seconds long, (2) let $x[a, b, c, d]$ is number of vehicle, is counted by organizer, in vehicle type a, move on MoI b, and get out of RoI at segment c of video d, $\hat{x}[a, b, c, d]$ is similar to $x[a, b, c, d]$, but it is counted by each team, then calculate the root-mean-square error as $RMSE = \sqrt{mean(x[a, b, c, d] - \hat{x}[a, b, c, d])^2}$ The *RMSE* evaluates the average number of wrong counted vehicles per each type considered by MoI on each segment of testing videos. The smaller *RMSE* means the better solution. *Final round* The score S of each team on final round combines accuracy score S_{acc} and efficiency score S_{eff} as $S = \tau * S_{acc} + \beta * S_{eff}$, where $\tau = 70$, $\beta = 30$ set by the competition organizers. The accuracy score is the normalized value of *RMSE* following these steps: let $RMSE_{max}$ is max of RMSE value of all teams, the Normalized RMSE of each team is calculated by $RMSE_{norm} = RMSE / \min(5, RMSE_{max})$. Then the S_{acc} is normalized of the $RMSE_{norm}$ within the range [0, 1] as $S_{acc} = \max(0, 1 - RMSE_{norm})$. The efficiency score is based on the total Execution Time (*ET*) provided by each team, and normalized within the range [0, 1] as $S_{eff} = \max(0, 1 - ET/(5 * total\_video\_time))$

4.2 AI challenge evaluation

It is worth noting that the organizers disallow self-testing. Thus we only obtained the experiment results from the scoreboard described in Sect. 4.1, also we were not provided any

Table 2 The scoreboards of the preliminary and final round obtained from AIC organizer

Preliminary round			Final round			
#	Team	RMSE	#	Team	RMSE	Score
1	056	2.73	1	056	1.60	74.50
5	**114**	**3.11**	5	015	2.19	66.51
10	131	3.49	**6**	**114**	**2.19**	**64.51**
20	065	4.34	10	112	2.73	58.73
30	100	8.18	15	059	3.76	42.24
40	104	12.32	20	011	14.42	17.62

The bold text is our final results

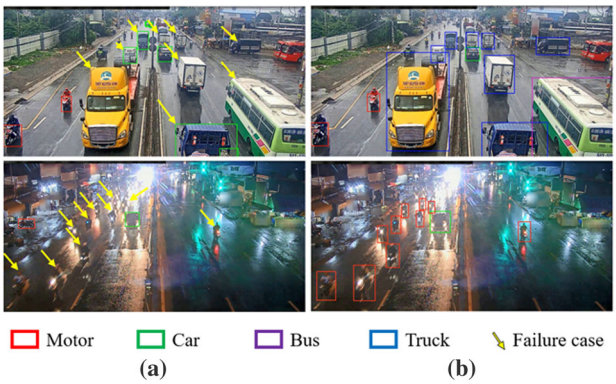


Fig. 6 Comparison performance between pretrained (**a**) and retrained (**b**) models on HCMC AIC. Failure cases between the pre-trained and re-trained detector on our dataset are showed by yellow arrows

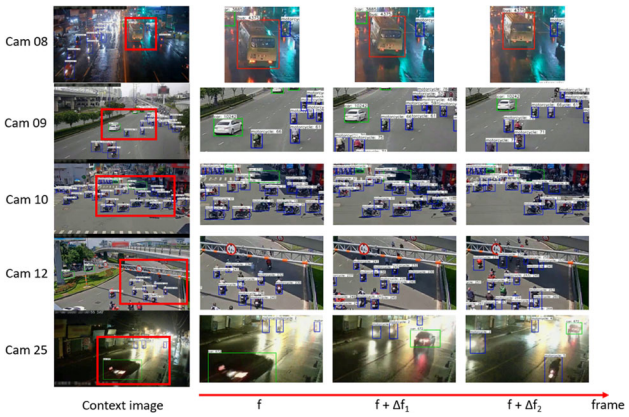


Fig. 7 Exemplary results on tracking multiple objects. Please zoom in 400% for better details

information about the ground-truth. Testset A and B were used to evaluate submission results on the preliminary round and final round, respectively. In the preliminary round, the score of each team is evaluated by the *RMSE* score shown in Sect. 4.1. As shown in Table 2, we achieved the 5th rank out of 217 teams in the preliminary round. In the final round, only 25 top teams submitted their code for the evaluation. The score of each team is computed as introduced in Sect. 4.1. The final score is shown in Table 2. Our team achieved rank 6 with the *RMSE* on par with the 5th team. This shows our competitive performance over other submissions.

4.3 Ablation study

4.3.1 Detection component

In recent years, deep-learning-based detection methods have significant achievements. Specifically, the most popular two-stage object detectors are the R-CNN family involving R-CNN [20], fast R-CNN [21], faster R-CNN [18], R-FCN [22] and Libra R-CNN [23]. As for one-stage object detector, the most representative models are YOLO series [7,24–26], SSD [27], RetinaNet [28]. For the implementation, we consider state-of-the-art object detectors for our evaluation, namely Faster RCNN [18] with a ResNet-101 [29] feature pyramid network (FPN) [30] backbone, YOLOv3-SPP [26] and YOLOv4 [7] with CSP [8]. These approaches have significantly and effectively achieved good performance to tackle tasks of object detection. However, not only accuracy but also efficiency is required in real-time systems. Via our initial experiment, YOLOv4 obtains higher mAP and competitive speed processing on frame rates of 73 FPS as in Table 3. We split our dataset into 3 sets: 19,166 (70%), 2739 (10%) and 5476 (20%) images for train, validation and test, respectively. Table 3 also shows the hyperparameters of these methods for training. Additionally, according to the score measurement given at Sect. 4.1, we opt to YOLOv4 with CSP. Figure 6 shows the improvement of retrained YOLOv4 model over the pretrained one.

Table 3 Hyperparameters and corresponding results on our *ADTC27K* testset of each detection model on GPU Tesla P100

Method	Configuration					Metric					
	LR	S	WD	E	BS	Motor	Car	Bus	Truck	mAP	FPS
YOLOv3-SPP	0.01	(640, 640)	0.0005	15	8	95.4	98.1	98.1	97.0	97.2	80
Faster R-CNN + R101-FPN	0.0025	(600, 1000)	0.0001	15	2	96.6	97.6	97.8	97.5	97.4	7
YOLOv4 + CSP	0.01	(608, 608)	0.0005	15	4	97.2	98.3	97.8	97.9	97.8	73

LR learning rate, *S* image size, *WD* weight decay, *E* epoch, *BS* batch size

Table 4 Benchmark results of our system on different computer hardwares

	CPU	RAM	GPU	Time (s)	FPS
Setting 1 (AIC Organizer)	Unspecified (4-core)	15 GB	Unspecified, but similar to GeForce® GTX 1080 Ti 11 GB	27,405	12.3
Setting 2	Intel® Xeon® Silver 4210 @ 2.20 GHz	252 GB	GeForce® RTX 2080 Ti 11 GB	23,148	14.6
Setting 3	AMD Ryzen™ 9 3900X @ 3.80 GHz	64 GB	GeForce RTX™ 3070 SUPRIM X 8G	18,403	18.3

4.3.2 Tracking component

DeepSORT versus tracktor Among tracking algorithms, DeepSORT [9,10] and Tracktor [19] are prominent candidates for multi-object online tracking. The performance of these two methods were proved on the common challenge dataset of MOT16 [31]. However, in this work, we leverage the HCMC AIC dataset which is specific for vehicle counting only. We *evaluate the performance of our tracker via the total performance of the whole proposed framework* due to the lack of tracking annotation. In order to select a suitable method for our system, we apply the two mentioned methods on a half of testset A. Tracktor is reported to take average 240 min to reach 8.595 RMSE score while DeepSORT takes only average 20 min to achieve 7.800 RMSE score. Notably, the smaller RMSE, the better performance. Our results on multiple vehicles (shown in Fig. 7) demonstrate that DeepSORT is more appropriate to our real-time tracking module as it optimizes the running time and RMSE score of the whole system.

Motorcycle overlap enhancement We also face the trouble of numerous ID switching errors. We notice the problem mostly happened to small objects like motorcycles. Note that the pretrained tracking model was trained to track pedestrian in low speed and stable between frames. Besides, the original training data was only collected on daytime. While in HCMC AIC, we were provided with low-frame-rate videos (only 10 FPS instead of 24 FPS or higher). There exist cases that vehicles suddenly flashed by a far distance. Hence, we tackle this issue by coming up with an idea of upsizing the bounding boxes of motorcycles from the center. Via experiments, the extensions of 20, 50 and 100% of the detected bounding boxes were applied. The value of 100% extended bounding boxes yields the highest RMSE score of 4.19 (while others were around 4.3).

Hyperparameter refinement Generally, one set of hyperparameters does not adapt all the cameras whose properties were different from each other in many aspects such as weather condition, light condition, traffic density and perspective view. Thus, we introduce a searching algorithm for the tuning procedure (hyperparameter optimization). We perform Grid Search *and then* Random Search in the range of values that Grid Search showed major results among the three hyperparameters: max cosine distance (from 0.1 to 1.0), NMS max overlap (from 0.5 to 1.0), max IoU distance (from 0.5 to 1.0) for all vehicles of each camera. Therefore we opt to separately tune hyperparameters for each vehicle type in each camera. However, Random Search values of cars, buses and trucks show minor changes in comparison with Grid Search results. Thus we only apply Random Search to motorcycles.

4.4 Discussion

Overall, our system achieves the high performance; Table 4 illustrates the runtime performance results of our system evaluated on different hardware configurations. Regarding the failure cases, the extremely high traffic density of motorcycles outputs ID switch errors between them caused by the nearly identical appearances or overlapping. In addition, the amount of vehicle types running on the same lane leads to a complete occlusion from big size vehicles on small vehicles or ones moving on the opposite side. Also, the confusing classification among vehicles types such as the bus and the car because of the similar appearances between 16-seat bus and car. Fourth, the small appearances of vehicles, especially motorcycles, result in mis-labeling or mis-detection. Last but not least, the videos are not in the high quality due to the weather conditions such as rainy days, black-white content or low light.

5 Conclusion

In this paper, we introduce adaptive multi-vehicle motion counting with adaptive label-independent tracking and counting modules. The experiments demonstrate that our proposed system successfully detects, tracks and counts four types of vehicles with various camera perspectives. Our method achieved the high ranking in both the preliminary round, and the final round of HCMC AIC. In the future, we aim to reduce ID switch errors by improving trajectory tracking and dealing with small or occluded objects with an end-to-end detector and tracker. In addition, we consider re-identification methods to tackle occurrences of big vehicles over smaller vehicles. The inter-class similarity is also considered to reduce the wrong detection.

Acknowledgements This research is funded by University of Information Technology - Vietnam National University Ho Chi Minh City under grant number D1-2022-06. This project is also supported by National Science Foundation (NSF) under Grant No. 2025234.

References

1. Naphade, M., Wang, S., Anastasiu, D.C., Tang, Z., Chang, M.C., Yang, X., Zheng, L., Sharma, A., Chellappa, R., Chakraborty, P.: The 4th AI city challenge. In: Conference on Computer Vision and Pattern Recognition Workshops, pp. 2665–2674 (2020)
2. Yu, L., Feng, Q., Qian, Y., Liu, W., Hauptmann, A.G.: Zero-virus*: zero-shot vehicle route understanding system for intelligent transportation. In: Conference on Computer Vision and Pattern Recognition Workshops, pp. 2534–2543 (2020)
3. Ospina, A., Torres, F.: Countor: count without bells and whistles. In: Conference on Computer Vision and Pattern Recognition Workshops, pp. 2559–2565 (2020)
4. Liu, Z., Zhang, W., Gao, X., Meng, H., Tan, X., Zhu, X., Xue, Z., Ye, X., Zhang, H., Wen, S., Ding, E.: Robust movement-specific vehicle counting at crowded intersections. In: Conference on Computer Vision and Pattern Recognition Workshops, pp. 2617–2625 (2020)
5. Tran, M.T., Nguyen, T.V., Hoang, T.H., et al.: iTASK—intelligent traffic analysis software kit. In: Conference on Computer Vision and Pattern Recognition Workshops, pp. 2607–2616 (2020)
6. Bui, K.H.N., Yi, H., Cho, J.: A vehicle counts by class framework using distinguished regions tracking at multiple intersections. In: Conference on Computer Vision and Pattern Recognition Workshops, pp. 2466–2474 (2020)
7. Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: YOLOv4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934 (2020)
8. Wang, C.Y., Mark Liao, H.Y., Wu, Y.H., Chen, P.Y., Hsieh, J.W., Yeh, I.H.: Cspnet: A new backbone that can enhance learning capability of CNN. In: Conference on Computer Vision and Pattern Recognition Workshops, pp. 1571–1580 (2020)
9. Wojke, N., Bewley, A., Paulus, D.: Simple online and realtime tracking with a deep association metric. In: International Conference on Image Processing, pp. 3645–3649 (2017)
10. Wojke, N., Bewley, A.: Deep cosine metric learning for person re-identification. In: Winter Conference on Applications of Computer Vision, pp. 748–756 (2018)
11. Huang, S., Li, X., Zhang, Z., Wu, F., Gao, S., Ji, R., Han, J.: Body structure aware deep crowd counting. IEEE Trans. Image Process. **27**(3), 1049–1059 (2018)
12. Barandiaran, J., Murguia, B., Boto, F.: Real-time people counting using multiple lines. In: 2008 Ninth International Workshop on Image Analysis for Multimedia Interactive Services, pp. 159–162 (2008)
13. Wan, J., Chan, A.: Adaptive density map generation for crowd counting. In: International Conference on Computer Vision, pp. 1130–1139 (2019)
14. Liang, M., Huang, X., Chen, C.H., Chen, X., Tokuta, A.: Counting and classification of highway vehicles by regression analysis. IEEE Trans. Intell. Transp. Syst. **16**(5), 2878–2888 (2015)
15. Kamkar, S., Safabakhsh, R.: Vehicle detection, counting and classification in various conditions. IET Intell. Transp. Syst. **10** (2016)
16. Seenouvong, N., Watchareeruetai, U., Nuthong, C., Khongsomboon, K., Ohnishi, N.: A computer vision based vehicle detection and counting system. In: 2016 8th International Conference on Knowledge and Smart Technology, pp. 224–227 (2016)
17. Xuezhi, X., Zhai, M., Ning, L., El Saddik, A.: Vehicle counting based on vehicle detection and tracking from aerial videos. Sensors **18**, 2560 (2018)
18. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans. Pattern Anal. Mach. Intell. **39**(6), 1137–1149 (2017)
19. Bergmann, P., Meinhardt, T., Leal-Taixé, L.: Tracking without bells and whistles. In: International Conference on Computer Vision, pp. 941–951 (2019)
20. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Conference on Computer Vision and Pattern Recognition, pp. 580–587 (2014)
21. Girshick, R.: Fast R-CNN. In: International Conference on Computer Vision, pp. 1440–1448 (2015)
22. Dai, J., Li, Y., He, K., Sun, J.: R-FCN: object detection via region-based fully convolutional networks. In: Proceedings of the 30th International Conference on Neural Information Processing Systems, pp. 379–387 (2016)
23. Pang, J., Chen, K., Shi, J., Feng, H., Ouyang, W., Lin, D.: Libra R-CNN: towards balanced learning for object detection. In: Conference on Computer Vision and Pattern Recognition, pp. 821–830 (2019)

24. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Conference on Computer Vision and Pattern Recognition, pp. 779–788 (2016)
25. Redmon, J., Farhadi, A.: Yolo9000: Better, faster, stronger. In: Conference on Computer Vision and Pattern Recognition, pp. 6517–6525 (2017)
26. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767 (2018)
27. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: SSD: Single shot multibox detector. In: European Conference on Computer Vision, pp. 21–37 (2016)
28. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: International Conference on Computer Vision, pp. 2999–3007 (2017)
29. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
30. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Conference on Computer Vision and Pattern Recognition, pp. 936–944 (2017)
31. Milan, A., Leal-Taixé, L., Reid, I., Roth, S., Schindler, K.: Mot16: A benchmark for multi-object tracking. arXiv preprint arXiv:1603.00831 (2016)