Real-time and Robust System for Counting Movement-Specific Vehicle at Crowded Intersections

Vu-Hoang Tran, Le-Hoai-Hieu Dang, Chinh-Nghiep Nguyen, Ngoc-Hoang-Lam Le,
Khanh-Phong Bui, Lam-Truong Dam, Quang-Thang Le, Dinh-Hiep Huynh
Ho Chi Minh City University of Technology and Education, Vietnam
hoangtv@hcmute.edu.vn

Abstract

In order to reduce traffic congestion and improve the efficiency of traffic light signals, intelligent traffic systems are being developed by researchers, and vehicle counting is one of the key techniques in the system. The traditional methods mostly focus on increasing the vehicle counting effectiveness without regard to the program execution efficiency. The practical value of these systems will be reduced if they cannot be operated in real-time on compact IoT device. Therefore, in this paper, we mainly focus on designing a real-time and robust system for the problem of counting specific-movement vehicles. The system is able to detect and track objects in the area of interest, then count those tracked trajectories using the movements. To improve performance of tracking multiple objects, a high recall detection method and an efficient feature matching strategy were proposed. Moreover, to minimize the wrong direction of movement prediction and improve the results of vehicle counting, a cosine similarity-based vehicle counting scheme is applied. Experiments are conducted on AI City 2021 Track-1 dataset. Our method is evaluated on both sides of efficiency and effectiveness.

1. Introduction

To reduce traffic congestion, control the flow of traffic and improve the efficiency of traffic lights, designing multiple movement vehicle counting systems are attracting the attention of researchers. The goal of these systems is to find out the number of vehicles that follow pre-defined movements in a period of time, and the time stamp when these vehicles move out of the region of interest (ROI). These information allow the system to predict how many vehicles will arrive at the next intersections over a period of time, the traffic demand and freight ratio on individual corridors. So that the system can design appropriate intersection signal timing plans and traffic congestion mitigation strategies. However, accurate vehicle counting is still challenging at crowded intersections, due to the difficulties such as the occlusions between different vehicles, various lighting and weather conditions (including dawn, rain, and snow …), not to mention environmental

Figure 1. Some images of roadside tree and weather conditions

conditions that affect the visibility of the camera as shown in Fig. 1. Traditional vehicle counting problems are solves by using some method like frame-wise vehicle counting [6] [1] [19][22], for the purpose of counting vehicles in one frame. Others use the density detection strategy [3][2] for regressing the number of vehicles. Or many other studies turn to discovery-based strategies [5][6][1][18][19], that is, detecting the vehicle then counting. However, the aforementioned methods cannot handle the mutual occlusions between vehicles and the occlusions caused by roadside trees shown in Figure 1, detection and density-based approaches often miss congested vehicles. Moreover, the computational efficiency is also an issue that needs to be addressed.

Therefore, in this paper, we propose to use ScaledYolov4 [23] and Deep SORT [21] as the baseline methods for vehicle detection and multiple object tracking. In detector, ScaledYolov4 is carefully fine-tuned to not only minimize miss detection but also improve the computational efficiency. Then, to handle the problem of tracking objects that are missing in some frames due to occlusions, Deep SORT with the designed efficient features is used. Furthermore, to deal with vehicle counting by movement of interest (MOT) problem, we firstly propose to use cosine similarity to quickly eliminate the directions that are not interested, then, use the orbit-based nearest neighbor analysis to determine the correct MOT.

The main contributions of our paper are summarized as follows: (1) we re-designed a detection-tracking-counting (DTC) for movement-specific vehicle counting problem regard to both effectiveness and efficiency. (2) We modified Deep SORT with the efficient features to improve

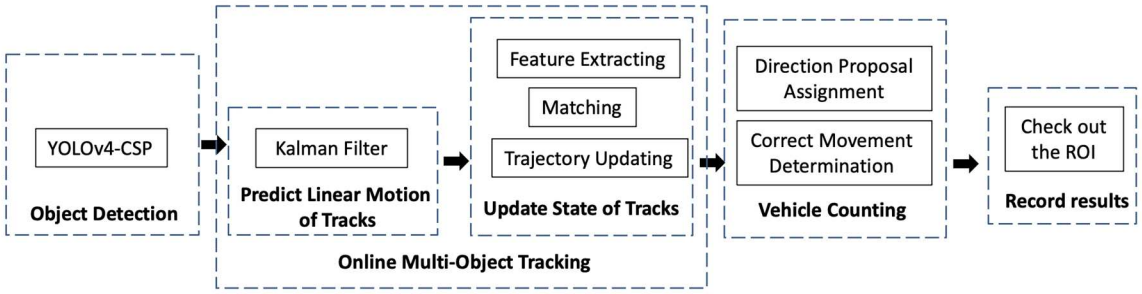

Figure 2. Our framework.

the multiple objects tracking performance. (3) We proposed the cosine similarity-based and orbit-based nearest neighbor analysis to improve the vehicle counting performance.

2. Related Works

Object Detection: Early object recognition implementation involved the use of classical algorithms, like those supported in Open CV. Feature extraction is one of the primary tasks in object detection and most object detection algorithms [10] [20] are designed based on manual features (HOG [4], SIFT [13]), then the traditional classifiers such as Naive Bayes, SVM are used to obtain the results of the detection. However, these methods cannot achieve sufficient performance when working under different conditions. To solve these problems, the state-of-the-art methods from two perspectives, effectiveness and efficiency, such as YOLOv4-CSP [23] EfficientDet-D1 [24] and RetinaNet [25] were proposed. The comparison about the effectiveness and efficiency using AP [27] and FPS of the three algorithms is given in Table 1. Based on the Table, Yolov4-CSP with input resolution of 512 is both effective and efficient. Therefore, our detector will be designed based on Yolov4-CSP structure.

Algorithms	AP	FPS	Image Size
EfficientDet-D1	40.5	74	640
YOLOv4-CSP	47.5	73/70	640
RetinaNet	41.5	53	640
YOLOv4-CSP	46.2	97/93	512

Table 1. Comparison of effectiveness and efficiency on different detection algorithms on COCO2017 test set [14]

Multi-Object Tracking: Multi-Object tracking (MOT) is another very important task in computer vision. In recent years, with the improvement of object detection, many existing MOT studies adopts the tracking-by-detection strategy [7] [32] [33] [35], which performs object detection first and then associate the detections afterwards. In [21], to determine the bounding box's orbital motion then predict the next bounding box, the deterministic Hungarian algorithm [34] was used. Besides, to increase tracking accuracy, some movement and visual information were used to replace the traditional association metric. This framework got the good performance in terms of tracking precision and accuracy. However, because of using the deep features, which take a lot of time to extract, the processing speed of the above method is not very impressive. Therefore, to improve the execution speed, we will slightly modify it using the more efficient features.

Vehicle Counting: As a fundamental technique for intelligent transportation, vehicle counting is also under investment in recent research. The existing methods of vehicle counting can be mainly divided into two categories: density-aware approach [6] and detection-aware approach [38] [39] [40]. In the detection-aware approach, the object can be firstly detected by some deep learning detectors or background subtraction models then the simple feature-based tracking is used to generate object position during video [36]. By setting the entrance and exist the vehicle counting can be then performed. However, these methods may not yield high results for counting movement-specific vehicle, where vehicle counting should be performed separately for different pre-defined movements such as left-turning, right-turning or through traffic at a given intersection. Therefore, to improve the performance for this particular task, we first find and disassemble the typical trajectories for each movement using statistical method. Then, we assign one movement for each vehicle by measuring the distances between the discrete vehicle's trajectory and the discrete typical trajectories. Finally, vehicle counting is performed by determining when the certain vehicle is fully exiting the ROI.

3. Methodology

The overview of our framework is given in Fig. 2 including three main parts: object detection, online multi-object tracking, and vehicle counting, which will be specified below.

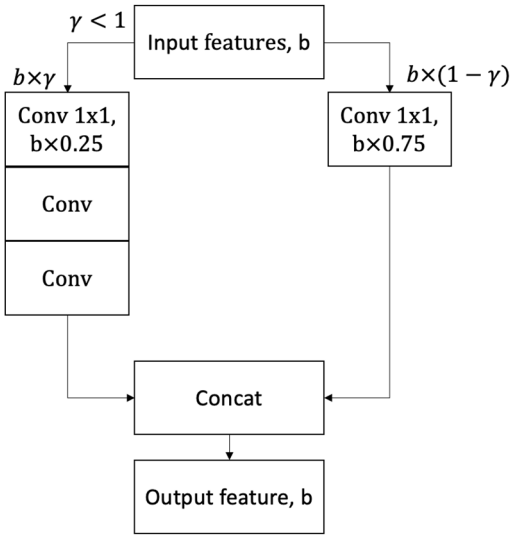

Figure 3. CSP Bottle Neck structure.

3.1 Object Detection

Considering both effectiveness and efficiency, we choose Yolov4-CSP as the base-line model for vehicle detection with the input resolution of 512x512. According to the structure of the CSP [29] the features are divided into two branches as shown in Fig. 3, one will pass through some more layers to be further enhanced and another will skip these layers. Normally, input features will be split in half (with $\gamma = 0.5$). But in this paper, to speed up the model, we reduce γ to 0.25, this will reduce the number of parameters and thereby increase FPS, but as a result it will decrease mAP. To compensate for the reduction in terms of mAP, synchronized batch normalization [27][28] is used to pre-train our model.

3.2 Online Multi-Object Tracking

After detecting objects, Deep SORT [21] is used as the baseline method for online MOT. We slightly modified Deep SORT algorithm to improve the execution speed.

3.2.1 Preliminary

Similar to [21], we also adopts a single hypothesis tracking methodology with recursive Kalman filtering [31] and frame-by-frame data association. State space in Kalman filtering is defined in the eights dimensional state spaces, including the bounding box center position (u, v), aspect ratio γ, height h, and their respective velocities in image coordinates. Object motion follows a standard Kalman filter with constant velocity model. However, different from [21], we combine motion feature, time feature, movement feature and shape feature to obtain similarity matrix between detection and predicted Kalman states of tracks. For each track k we count the number of frames since the last successful measurement association. Tracks

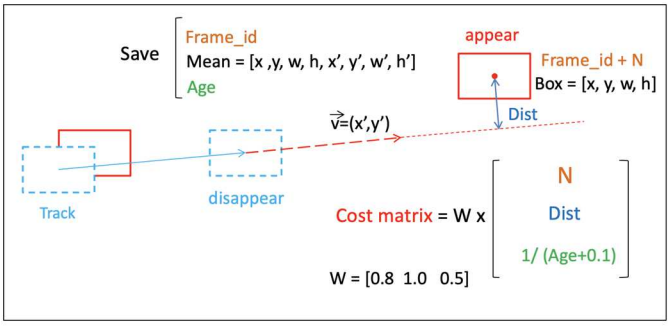

Figure 4. Calculate the cost matrix from efficient features

that exceed a predefined maximum age A_{max} are considered to have left the scene and are deleted from the track set. New track is initiated each detection that cannot be associated to the existing tracks. In order to be lighter to implement in real-time, we propose some methods to improve speed without reduce performance.

3.2.2 Efficient features matching

One of typical challenges in AIC is the limited camera vision, which is usually caused by common obstacles (bushes, road signs, utility poles, bad weathers…). This challenge makes object detection model become less reliable, leading to the frequent occurrence of inconsistent bounding boxes. Therefore, using the similarities in object's features is much more effective than using IOU for track-matching. However, deep feature extractor requires a lot of computing power, so a combined feature is suggested, this combined feature contain some simple features, which are information of shape, direction and age of track.

Direction feature: Track's direction is predicted based on its velocity vector using the Kalman filter. Whenever a bounding box appear in detection task, its direction will be defined by how close its current location to different initial trajectories.

Appearance time feature: The frame when an object suddenly disappears within ROI will be saved as shown in Fig. 4. The bounding boxes that appeared suddenly after this frame will be considered more by using the cost matrix function described in Figure 4. In this cost matrix, some new parameters are used including: **N**, **Dist** and **Age. N** means appearance time feature. The smaller **N** means the object appeared closer to the disappearance time. **Dist** is the distance from the object to our prediction vector, the closer it is, the higher the possibility of merging. **Age** is the age of the object being tracked; we prioritize match for those with high value of **Age**.

3.3 Vehicle Counting

In this section, we predict the trajectory of each track and determine frame id on which the track disappeared from the ROI. Firstly, we find direction proposals of each track by cosine similarity. Next, we predict the correct movement by

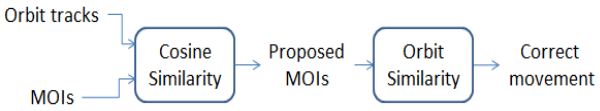

Figure 5. The process of determining the direction of movement

measuring the similarity between trajectory of track and the pre-defined trajectories of the movements. Finally, we record the result when any track exiting the ROI. Figure 5 describes the process of determining the direction of movement of a vehicle.

3.3.1 Pre-defined Trajectories

For each camera, we first visualize all centers of the labeled bounding boxes in training data. Then, we manually draw a set of points as the pre-defined trajectory based on that as shown in Figure 6.

3.3.2 Direction Proposal Assignment

Trajectory of Object (Ob) and pre-defined trajectories of movement of interesting (MOI) are both divided into sevaral segments as shown in Figure 7. MOI proposal M_i of k^{th} object Ob_k is defined as Eq. 1. Usually, when determining the direction of a track, we are only interested in the starting and ending points in the track's trajectory. This will not be correct when applied to curved orbits. We proposed dividing the orbits into several segments, then calculating the cosine similarities between each pair segments and then adding them up as shown in Eq. 2. The more segments, the more accurate.

$$MOI\_proposal\_Ob_k = [i: \cos(Ob_k, M_i) \geq \alpha], \quad (1)$$

where, $\cos(Ob_k, M_i)$ is defined as in Eq. 2, α is a pre-defined threshold. In this paper, we set α equal to 2.2.

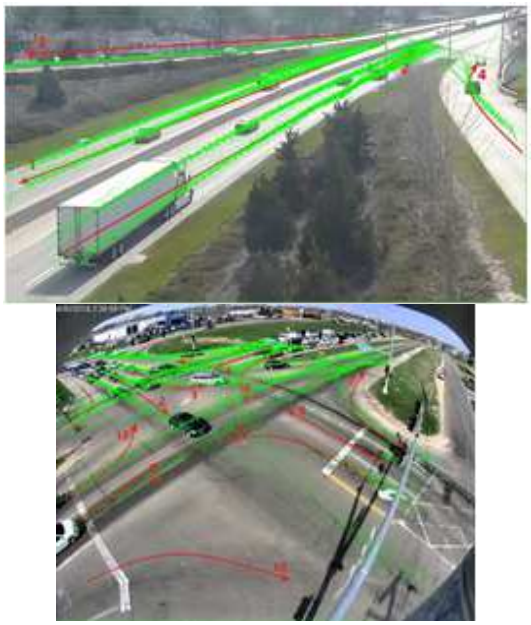

Figure 6. The pre-defined trajectories

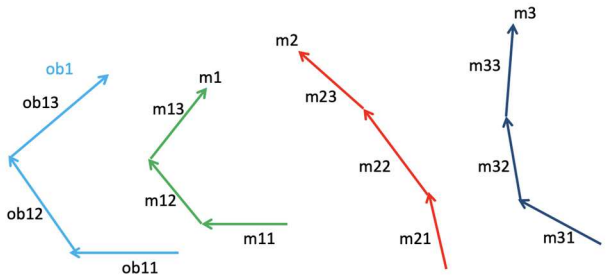

Figure 7. Trajectories is segmented

$$\cos(Ob_k, M_i) = \sum_{j=1}^{s} \cos(Ob_{kj}, M_{ij}). \quad (2)$$

In equation 2, $\cos(Ob_{kj}, M_{ij})$ is cosine of the angle between Ob_{kj} vector and M_{ij} vector, the illustration of these vectors is shown in Fig. 7. Ob_{kj} is vector j-th vector in the trajectory segmentation of Ob_k. M_{ij} is vector j-th vector in the trajectory segmentation of M_k. s is the number of segments, in our case, s is set to 3.

3.3.3 Determine Correct Movement of Interesting

Sometimes we have cases where the two pre-defined directions (M1, M2) and the actual directions are very close to each other, causing confusion as shown in Fig. 8. So, we deal with it by using Algorithm 1, named as Orbit-based nearest neighbor analysis, to calculate the distance and determine the "number of nearest points" for each pre-defined direction. Based on this, we are able to accurately determine the direction of movement limiting the aforementioned confusions. Algorithm 1 shows our proposed Nearest Neighbor Analysis based on movement prediction method. In Algorithm 1, all trajectories are fragmented into point sets. For each point in the trajectory of the track, we find the closest point that is in the pre-defined trajectories. Variable counter_error is used to count the number of points track's trajectory that are too far from the pre-defined MOI. If counter_error is too large, this track will be removed. The exact movement is the movement whose index appears most frequently in array C.

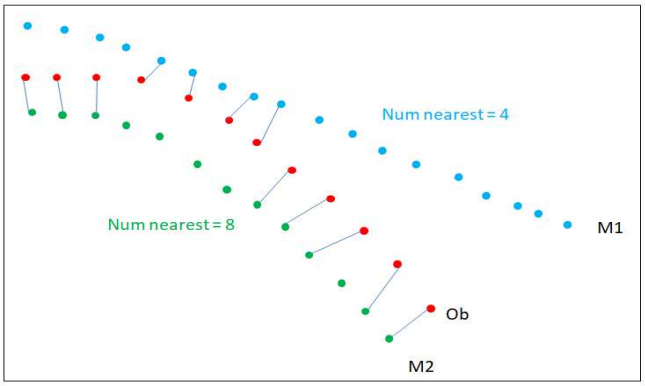

Figure 8. Orbit-based nearest neighbor analysis

Algorithm 1: Orbit-based nearest neighbor analysis

Input: Given one track:

Trajectory of track with n point: $Orbit = \{p_0, p_1, ... p_n\}$;

Pre-defined Trajectories of each MOI: $MOIs = \{M_1, M_2, ... M_i\}$;

Proposed MOIs indices: D;

Distance threshold: H;

Output: Correct movement: res;

$MultiPoint = [M_i : i \in D]$;

$counter\_error = 0$;

Initialize list $C = []$;

for p in $Orbit$ **do**

  find point in $MultiPoint$ nearest to p:

  Ne = nearest_points(p, $MultiPoint$);

  if $dis_{p,Ne} \geq H$

  then

    $counter\_error++$;

    continue;

  end if

  for d in D **do**

    if $Ne \in M_d$

    then

      Append d to list C: $d >> C$;

      break;

    end if

  end for

end for

if $counter\_error > n/2$

then

  return 0;

end if

return $res = \max(C, key = C.count)$;

4. Experiments

4.1. Implementation Environment.

Hard ware: All of our experiments are tested on Tesla T4 with 16G and Intel(R) Xeon(R) Silver 4216 CPU @ 2.1GHz.

Software: Python 3.7, Torch 1.7, CUDA 10.2.

4.2. Datasets

AI City 2021 Track-1 Dataset [30]: The data for this challenge comes from multiple traffic cameras from a city in the United States as well as from state highways in Lowa which contains 31 video clips (about 9 hours in total) captured from 20 unique camera views (some cameras provide multiple video clips to cover different lighting and weather conditions.). Each camera view comes with a detailed instruction document describing the region of interest (ROI) and movements of interest (MOI). The 9 hours of video in track 1 are split into two datasets A and B. Dataset A (about 5 hours in total) are made available to participating teams, dataset B (about 4 hours in total) will be used for later testing and not available to participates. So all our test results are done on dataset A. Detection model is fine-tuned on the AI City 2021 Track-1 dataset A with pre-trained model on COCO. Totally 2860 frames from the video in AICity2021 dataset A were annotated, where only cars and trucks are considered.

4.3. Evaluation Metrics

We adopt the official evaluation metrics in AI City 2021 Challenge [30]. The final score S1 is a weighted combination between efficiency score $S_{1efficiency}$ and effectiveness score $S_{1effectiveness}$. It is defined as Eq. 3.

$$S_1 = 0.3 \times S_{1efficiency} + 0.7 \times S_{1effectiveness}. \quad (3)$$

In (3), the $S_{1efficiency}$ score is based on the total execution time provided by the contestant, adjusted by the efficiency base factor, and normalized within the range [0, 1.1x video play-back time]. $S_{1effectiveness}$ is computed as a weighted average of normalized weighted root mean square error scores nwRMSE across all videos, movements, and vehicle classes in the test set, with proportional weights based on the number of vehicles of the given class in the movement. To reduce jitters due to labeling discrepancies, each video is split into segments and we consider the cumulative vehicle counts from the start of the video to the end of each segment.

4.4. Object detection result

On this experiment of object detection, we will test models in the test dataset which is 10% of AI City 2021 Track-1 dataset. Evaluation results are based on mAP scores with priority for recall [26], and FPS as shown in Table 2. During this mission, we expect to identify as many vehicles as possible, so that the result will have lower miss rates, leading to higher effectiveness scores. Also, FPS is directly affected to efficiency. Therefore, the model must care about both mAP and FPS.

We compare Yolov4-CSP, Yolov4-CSP-0.25 and Yolov5x as shown in Table 2. Based on the Table, Yolov4-CSP with gamma=0.25 have higher FPS than Yolov5x and Yolov4-CSP. Its mAP is approximate with Yolov4-CSP, and higher than Yolov5x. Therefore, Yolov4-CSP-0.25 is chosen to ensure both sides: effectiveness and efficiency. After that we pre-trained model Yolov4-CSP-0.25 with synchronized batch normalization to improve mAP with batch-size=8, denoted as Yolov4-CSP-0.25-sync.

Model	mAP_{50}(%)	Precision(%)	Recall(%)	FPS
Yolov4-CSP	87.6	65.5	**93.6**	133
Yolov4-CSP-0.25	86.8	62.1	92.3	164
Yolov5x	88.1	84.0	85.5	59
Yolov4-Csp-0.25-sync	**87.7**	65.7	92.7	**169**

Table 2. The comparison of mAP and FPS with different methods

4.5. Tracking

To improve tracking task, several versions based on Yolov4-CSP are implemented. The results are represented in Table 3. Besides, to show the effectiveness of "efficient features matching", we also show some tracking results comparing between with and without the proposed "efficient features matching" as shown in Fig. 9. With "efficient features matching", we are able to track the objects smoothly without being affected by sudden disappearances due to the effects of weather or visibility.

Model	Effectiveness	Efficiency	Score
Yolov4-CSP 512-s12-fd	**0.9476**	0.8673	**0.9235**
Yolov4-CSP 512-s2	0.93	**0.8822**	0.9156
Yolov4-CSP 512-s1	0.939	0.87	0.9183

Table 3. Different versions of Yolov4-CSP

(a)

(b)

Figure 9. Tracking results (a) after applying efficient feature matching. (b) before applying efficient feature matching.

Yolov4-CSP-512-s1: Using 1 frame-skipping to improve performance of the model, though it may affect the accuracy, the benefit of performance gain is more valuable.

Yolov4-CSP-521-s2: To see whether we can gain more performance without heavily affect model's accuracy, 2 frames skipping is implemented. Although model has become unreliable, gaining more performance. Therefore, 2 frame skipping approach is chosen to be the foundation to develop other improvements.

Yolov4-CSP-512s-s12-fd: In this final version, either 1 or 2 frame-skipping is used depending on video source. By doing this, our model has become more flexible so it can increase both its accuracy and performance at the same time.

4.6. Overall Score on AI City Challenge 2021 Track 1 Dataset

Table 4 show the leader board of AI City Challenge 2021 Track 1. Our proposed vehicle counting method achieved 4th place in the ranking.

Team ID	S1 score
37	0.9467
5	0.9459
8	0.9263
19 (our)	0.9249
118	0.9235

Table 4. Top 5 overall scores of the vehicle counting task in AI City 2021 track 1.

References

[1] A. Abdagic, O. Tanovic, A. Aksamovic, and S. Huseinbegovic. Counting traffic using optical flow algorithm on video footage of a complex crossroad. pages 41–45, 2010.

[2] Shubhra Aich and Ian Stavness. Leaf counting with deep convolutional and deconvolutional networks. pages 2080– 2089, 2017.

[3] J. Barandiaran, B. Murguia, and F. Boto. Real-time people counting using multiple lines. pages 159–162, 2008.

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), volume 1, pages 886–893 vol. 1, 2005.

[5] Luca Del Pizzo, Pasquale Foggia, Antonio Greco, Gennaro Percannella, and Mario Vento. Counting people by rgb or depth overhead cameras. Pattern Recognition Letters, 81:41– 50, 2016.

[6] KV Embleton, CE Gibson, and SI Heaney. Automated counting of phytoplankton by pattern recognition: a comparison with a manual counting method. Journal of Plankton Research, 25(6):669–681, 2003.

[7] Wongun Choi. Near-online multi-target tracking with aggregated local flow descriptor. In Proceedings of the IEEE

international conference on computer vision, pages 3029– 3037, 2015.

[8] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. International journal of computer vision, 111(1):98–136, 2015.

[9] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. International journal of computer vision, 88(2):303–338, 2010.

[10] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In 2008 IEEE Conference on Computer Vision and Pattern Recognition, pages 1–8, 2008.

[11] Ross Girshick. Fast r-cnn. In Proceedings of the IEEE international conference on computer vision, pages 1440–1448, 2015.

[12] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 580–587, 2014.

[13] Luo Juan and Luo Gwon. A comparison of sift, pca-sift and surf. International Journal of Signal Processing, Image Processing and Pattern Recognition, 8(3):169–176, 2007.

[14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollar, and C Lawrence ´ Zitnick. Microsoft coco: Common objects in context. In European conference on computer vision, pages 740–755. Springer, 2014.

[15] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In European conference on computer vision, pages 21–37. Springer, 2016.

[16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 779–788, 2016.

[17] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems, pages 91–99, 2015.

[18] Frank Y Shih and Xin Zhong. Automated counting and tracking of vehicles. International Journal of Pattern Recognition and Artificial Intelligence, 31(12):1750038, 2017.

[19] L. Unzueta, M. Nieto, A. Cortes, J. Barandiaran, O. Otaegui, and P. Sanchez. Adaptive multicue background subtraction for robust vehicle counting and classification. IEEE Transactions on Intelligent Transportation Systems, 13(2):527–540, 2012.

[20] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, volume 1, pages I–I, 2001.

[21] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In 2017 IEEE international conference on image processing (ICIP), pages 3645–3649. IEEE, 2017.

[22] Honghong Yang and Shiru Qu. Real-time vehicle detection and counting in complex traffic scenes using background subtraction model with low-rank decomposition. IET Intelligent Transport Systems, 12(1):75–85, 2017.

[23] Chien-Yao Wang, Alexey Bochkovskiy, Hong-Yuan Mark Liao .

[24] Mingxing Tan, Ruoming Pang, and Quoc V Le. EfficientDet: Scalable and efficient object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[25] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), pages 2980–2988, 2017.

[26] Paul Henderson, Vittorio Ferrari. End-to-end training of object class detectors for mean average precision. Conference: Asian Conference on Computer Vision

[27] Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, Amit Agrawal. Context Encoding for Semantic Segmentation.In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[28] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift, arXiv preprint arXiv:1502.03167.

[29] Chien-Yao Wang, Hong-Yuan Mark Liao, I-Hau Yeh, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh. CSPNet: A New Backbone that can Enhance Learning Capability of CNN. Published in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW).

[30] Milind Naphade and Shuo Wang and David C. Anastasiu and Zheng Tang and Ming-Ching Chang and Xiaodong Yang and Liang Zheng and Anuj Sharma and Rama Chellappa and Pranamesh Chakraborty. The 4th AI City Challenge. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops.

[31] Qiang Li, Ranyang Li, Kaifan Ji, Wei Dai. Kalman Filter and Its Application. Published in: 2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINI).

[32] Caglayan Dicle, Octavia I Camps, and Mario Sznaier. The way they move: Tracking multiple targets with similar appearance. In Proceedings of the IEEE international conference on computer vision, pages 2304–2311, 2013.

[33] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In 2012 IEEE Conference on Computer Vision and Pattern Recognition, pages 3354–3361. IEEE, 2012.

[34] Harold W Kuhn. The Hungarian Method for the Assignment Problem. Naval research logistics quarterly, 2(1-2):83–97, 1955

[35] Aljosa Osep, Wolfgang Mehner, Markus Mathias, and Bastian Leibe. Combined image-and world-space tracking in traffic scenes. In 2017 IEEE International Conference on Robotics and Automation (ICRA), pages 1988–1995. IEEE, 2017

[36] Milind Naphade and Zheng Tang and Ming-Ching Chang and David C. Anastasiu and Anuj Sharma and Rama Chellappa and Shuo Wang and Pranamesh Chakraborty and Tingting Huang and Jenq-Neng Hwang and Siwei Lyu. The 2019 AI City Challenge. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops.

[37] Mohammad Shokrolah Shirazi and Brendan Morris. Vision based turning movement counting at intersections by cooperating zone and trajectory comparison modules. In 17th

International IEEE Conference on Intelligent Transportation Systems (ITSC), pages 3100–3105. IEEE, 2014

[38] Z. Dai, H. Song, X. Wang, Y. Fang, X. Yun, Z. Zhang, and H. Li. Video-based vehicle counting framework. IEEE Access, 7:64460–64470, 2019

[39] Huansheng Song, Xuan Wang, Cui Hua, Weixing Wang, Qi Guan, and Zhaoyang Zhang. Vehicle trajectory clustering based on 3d information via a coarse-to-fine strategy. Soft Computing, 22(5):1433–1444, 2018.

[40] Maojin Sun, Yan Wang, Teng Li, Jing Lv, and Jun Wu. Vehicle counting in crowded scenes with multi-channel and multi-task convolutional neural networks. Journal of Visual Communication and Image Representation, 49:412– 419, 2017.

[41] Milind Naphade and Ming-Ching Chang and Anuj Sharma and David C. Anastasiu and Vamsi Jagarlamudi and Pranamesh Chakraborty and Tingting Huang and Shuo Wang and Ming-Yu Liu and Rama Chellappa and Jenq-Neng Hwang and Siwei LyuMilind Naphade and Ming-Ching Chang and Anuj Sharma and David C. Anastasiu and Vamsi Jagarlamudi and Pranamesh Chakraborty and Tingting Huang and Shuo Wang and Ming-Yu Liu and Rama Chellappa and Jenq-Neng Hwang and Siwei Lyu. The 2018 NVIDIA AI City Challenge. Proc. CVPR Workshops.

[42] Milind Naphade and David C. Anastasiu and Anuj Sharma and Vamsi Jagrlamudi and Hyeran Jeon and Kaikai Liu and Ming-Ching Chang and Siwei Lyu and Zeyu Gao. The NVIDIA AI City Challenge. Prof. SmartWorld.

[43] Zheng Tang and Milind Naphade and Ming-Yu Liu and Xiaodong Yang and Stan Birchfield and Shuo Wang and Ratnesh Kumar and David Anastasiu and Jenq-Neng Hwang. CityFlow: A City-Scale Benchmark for Multi-Target Multi-Camera Vehicle Tracking and Re-Identification. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[44] Yue Yao and Liang Zheng and Xiaodong Yang and Milind Naphade and Tom Gedeon. Simulating Content Consistent Vehicle Datasets with Attribute Descent. The European Conference on Computer Vision (ECCV).