Determining Vehicle Turn Counts at Multiple Intersections by Separated Vehicle Classes Using CNNs

Ján Folenta Jakub Špaňhel Vojtěch Bartl Adam Herout
Graph@FIT, Brno University of Technology, Czech Republic
xfolen00@stud.fit.vutbr.cz, {ispanhel,ibartl,herout}@fit.vutbr.cz
https://medusa.fit.vutbr.cz/traffic/

Abstract

In our submission to the NVIDIA AI City Challenge 2020, we address the problem of counting vehicles by their class at multiple intersections. Our solution is based on counting by tracking principle using convolutional neural networks in detection and tracking steps of the proposed method. We have achieved 6th place on the dataset part A *of Track 1 with score S1 Total* = 0.8829*, (mwRMSE* = 4.3616*, S1 Effectiveness* = 0.9094*, S1 Efficiency* = 0.8212*). The proposed solution was placed at sixth place in the overall ranking on dataset part A.*

Figure 1. The main idea of this task — counting vehicles for every pre-defined movement of interest (travel direction).

1. Introduction

In this paper, we address the task of vehicle counting by their class at multiple intersections of the NVIDIA AI City Challenge 2020 (i.e. *Track1*).

Our solution is based on *counting by tracking*. It benefits from the usage of convolutional neural networks both in the detection and tracking steps. Trajectories of vehicles obtained for CNN feature-based tracker are further processed and analyzed to determine the entry and exit point of each vehicle for correct counting of trajectories.

To put our approach to a broader context, we include a brief overview of state of the art in vehicle counting. After that, we describe the used methods for counting vehicles at multiple intersections in detail.

2. Vehicles Counting in Image and Video

Counting people or objects in image and video has several use cases in various situations such as counting people crowds in public events, city centers or forbidden areas; determining attendance in schools or lectures; urban planning or managing high traffic roads, and more.

Crowd counting attracted wider research community than other object counting topic. In recent years, methods were published for counting people in crowds in different ways. The first group is focused on detection/tracking [16, 17, 18, 3, 2, 9, 23], another group is focused on counting by regression/estimating density maps [6, 7, 22]. Nowadays, convolutional neural networks dominate in a large amount of computer vision tasks and crowd counting is no exception [34, 33, 5, 32, 28, 20].

However, many vehicle counting methods exist based on crowd counting methods. The following sections provide a brief overview of vehicle counting methods and the available datasets.

2.1. Counting Vehicles by Regression/Heat-map Estimation

The characteristic feature of these methods is the use of a heatmap as an expected output of the convolutional neural network. The heatmap is usually created using a normal distribution with pre-defined standard deviation value σ over image annotations. The CNN is then learning the mapping between the input image and the expected output image.

Counting CNN [21] was explicitly designed for the task

2160-7516/20/$31.00 ©2020 IEEE
DOI 10.1109/CVPRW50498.2020.00306

of car counting, and it belongs to this category. Authors are using their CNN architecture, which is rather small and fast to train. They are using patches of 72×72 cropped from an image and heatmap 18×18 is the direct output of the network. *Hydra CNN* [21] contains multiple instances of Counting CNN at different scales and combines their outputs.

It is possible to use even approaches originally designed for crowd counting. A great example is *Context-Aware Crown Counting Network* [20] which adaptively encodes multi-level contextual information into their output features. Rather than running at different scales, the proposed network leverages from spatial pyramid pooling [14]. Even *PDANet* [1] builds upon the idea of CAN. The PDANet uses pyramid feature extraction with spatial and channel attentions are attached to the front-end to produce richer features. The model also distinguishes between images with sparse and dense object instances.

2.2. Vehicle Counting by Tracking

There are several approaches to the task of detecting, tracking, and counting moving vehicles from traffic cameras. Seenouvong et al. [27] proposed a method that uses a background subtraction technique to find foreground objects and vehicle counting is done by using a virtual detection zone. Swamy et al. [29] introduced a technique for detecting and counting the vehicles based on the color space model.

In recent years, object detection made a big progress thanks to convolutional neural network-based detectors. We can divide currently popular detectors into two categories: Detectors that predict bounding boxes for an image in one step such as SSD [19] and YOLO [24] and detectors based on region proposal such as R-FCN [8], R-CNN [12], Fast R-CNN [11], and Faster R-CNN [26]. Thanks to that progress, trackers based on tracking-by-detection paradigm, in which detector firstly detects objects in a frame and secondly, the tracker performs data association is very popular these days [4, 31]. The combination of these detectors and trackers is a suitable solution for counting diverse types of objects in image or video.

2.3. Vehicle Counting Datasets

Although the research area of car counting is not large nowadays, datasets exist dedicated to car counting. The TRANCOS dataset [13] was captured on highways and it contains $1,244$ images with $46,700$ annotated vehicles. Images are split into training, testing, and validation sets. Other datasets are focused more on parking lots than on roads. PUCPR+ dataset [15] is a subset of PKLot dataset [10]. Hsieh et al. [15] added additional annotations of car bounding boxes ($17,000$ cars) on a subset (125 images) of the original PKLot dataset. The PKLot dataset was captured from the 10^{th} floor of a building from a static camera viewing a large parking lot. They also published their drone-based car counting dataset called CarPK [15]; it contains $1,500$ images with approximately $90,000$ cars.

Unfortunately, these datasets are made for single image car counting using regression/heat-map prediction, then continuous video counting, which made them unsuitable for this task.

3. Vehicle Turn Counting Based on Trajectories

Track 1 of the NVIDIA AI City Challenge 2020 is focused on counting four-wheel vehicles and freight trucks that follow pre-defined movements (travel directions) (see Fig. 1) on different camera scenes which should help DOTs' traffic engineers in traffic analysis and planning. In this challenge's track, the emphasis is placed on effectiveness (precision of counting) and along that also on efficiency. The individual processing steps of our solution are described in the following sections.

3.1. Determining Entry and Exit Areas

Assigning a vehicle trajectory to a specific travel direction requires knowledge of vehicle entrance/exit area on the road. This task is crucial to obtain correct counts for every possible travel direction. In our case, the entrance and exit areas were annotated manually using polygonal representation from 3 to 5 points used for each polygon. For each travel direction defined by pair of an entrance/exit area, we also have a user-defined trajectory. Examples of areas defined for each camera together with region-of-interest can be found in Figure 2.

3.2. Vehicle Detection and Tracking

The proposed solution is based on vehicle detection and tracking. For every frame of the input video, we first detect vehicles using a CNN-based detector. In order to achieve real-time detection or even quicker processing, but still have a high detection accuracy, we decided to use the YOLOv3-416 detector [25]. For the tracking task, we decided to use *Simple Online and Real-time Tracking with Deep Association Metric* (DeepSORT) [31].

For each vehicle, we save its trajectory points. The centroid of the detected vehicle's bounding box is used as a point representing the vehicle in the trajectory. When the vehicle is no longer detected, but it still appears in a frame of the video (its trajectory has not left the ROI yet), we predict the next position of the vehicle based on its last trajectory points. We compute the average angle between the last 5 points of vehicle trajectory and the Euclidean distance between the last 2 points. Based on the computed angle and distance, we predict the next trajectory point of the vehicle.

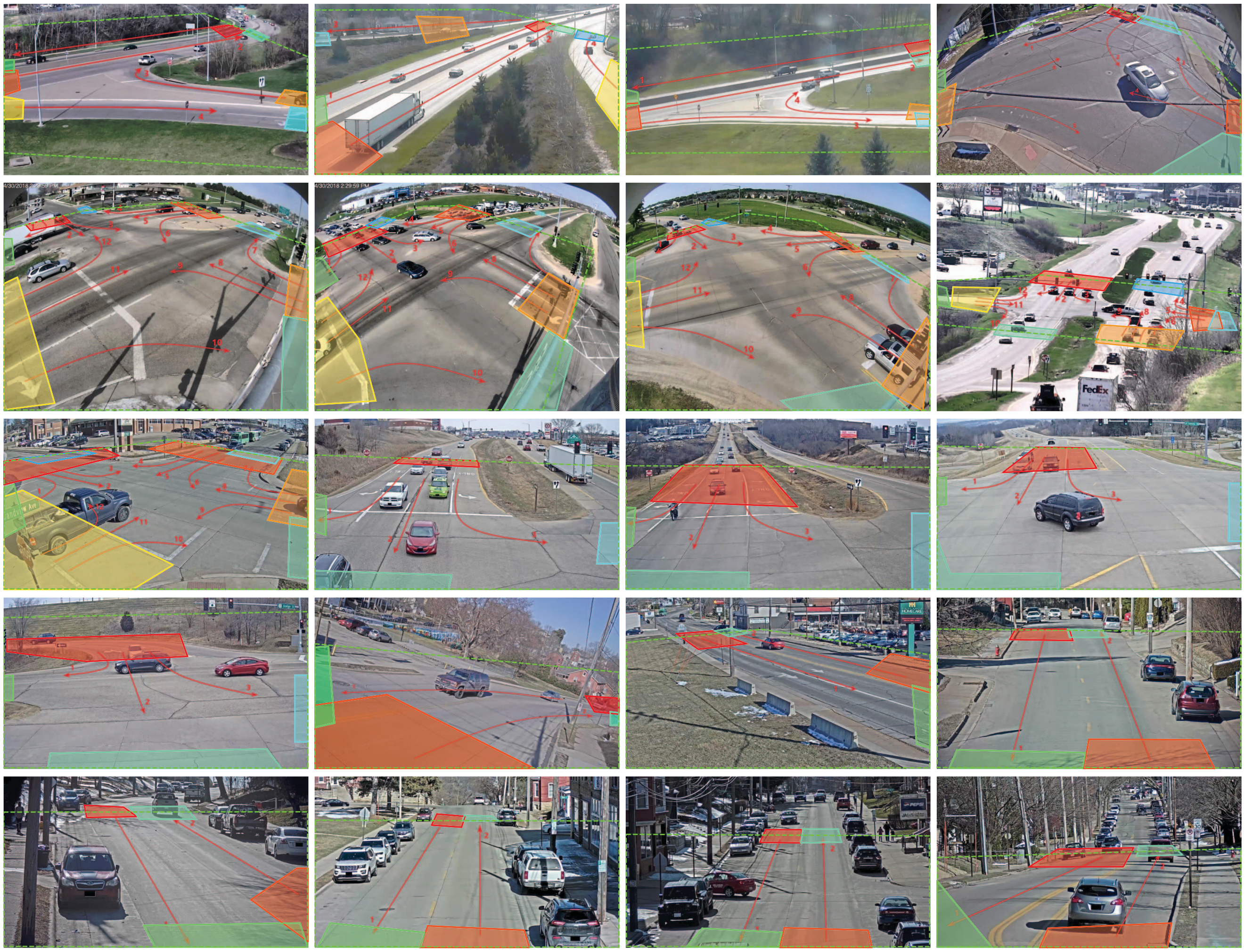

Figure 2. Examples images from 20 cameras in the dataset with annotated entrance/exit areas and pre-defined movements of interests and ROIs. Images were updated to fit the view better. Ordering of images does not fully correspond to the sequence of camera IDs.

3.2.1 Merging Broken Trajectories

Before the tracker prediction step, we try to deal with long-term occlusion and detection inaccuracies by a trajectory merging step. In the case when the vehicle was not visible for a certain period and then reappeared in a subsequent frame, it starts its own new trajectory even though the old trajectory of this vehicle is still predicted.

To deal with this situation, when one vehicle is being tracked twice, the proposed method is trying to merge broken trajectories. For each trajectory predicted by the tracker, we try to find a newly detected vehicle trajectory formed by 2 or 3 points. We compare the latest predicted points of the tracker-predicted trajectory with the location and direction of this new trajectory.

If the trajectories are similar in their location and speed, these two trajectories are merged. This merging prevents counting two vehicles, instead of a single correct one. This approach may lead to identity switching, but this is not a problem if vehicles are travelling in the same direction – which they are, otherwise they would not be merged by the algorithm.

3.3. Travel Direction Assigning

The final step is determining the correct travel direction (movement of interest) for each ended trajectory outside pre-defined ROIs. The travel directions are defined by pairs of the entrance/exit areas.

For each detected trajectory, the entrance area is defined by its intersection with the beginning of this trajectory, or by the distance to the closest one. The same principle is applied even for the exit area of each trajectory. This pair then defines a travel direction.

Unfinished trajectories In some situations, the detector can lose the vehicle in the middle of a pre-defined ROI. This situation may occur, for example, when the vehicle is turning from one road to another. In such case, the prediction-based trajectory of the vehicle will continue in the determined direction (e.g. straight even if the vehicle is turning), while the detection-based trajectory ends **before** it exceeds the border of the ROI. If new detections no longer update the prediction-based trajectory, it may cause that the vehicle will be counted in the wrong direction.

For these cases, our annotations for every camera contain the average trajectory for each travel direction. When every point of the detection-based trajectory can be fit to this average trajectory (point from detection-based trajectory is inside of polygon, which defines average trajectory), the prediction-based trajectory will follow this direction, even if the detection is lost.

4. Experiments

All the experiments were performed on official dataset part *A* provided by organizers of this challenge. It contains 31 video clip captured from 20 unique cameras (see Fig. 2) with about 9 hours of total video length. Provided dataset is a part of *CityFlow* dataset [30].

4.1. Evaluation metrics

The evaluation of this task was done using officially published evaluation metrics.

The Track 1 is ranked based on evaluation score S1 (Eq. 1) is a weighted combination between the Track 1 efficiency score $S1_{efficiency}$ and the Track 1 effectiveness score $S1_{effectiveness}$.

$$S1 = \alpha S1_{efficiency} + \beta S1_{effectiveness}, \tag{1}$$

where $\alpha = 0.3$, $\beta = 0.7$. The $S1_{efficiency}$ (Eq. 2) score is based on the total Execution Time of proposed solution, adjusted by the Efficiency Base factor, and normalized within the range [0, 5x video play-back time].

$$S1_{efficiency} = \max(0, 1 - \frac{time \times base\,factor}{5 \times video\,total\,time}). \tag{2}$$

The $S1_{effectiveness}$ score (Eq. 3) 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. The nwRMSE score is the weighted RMSE (wRMSE) between the predicted and true cumulative vehicle counts, normalized by the true count of vehicles of that type in that movement. If the wRMSE score is greater than the true vehicle count, the nwRMSE score is assigned 0, else it is (1-wRMSE/vehicle count). To further reduce that impact of errors on early segments, the wRMSE score weighs each record incrementally in order to give more weight to recent records.

$$wRMSE = \sqrt{\sum_{i=1}^{k} w_i(\hat{x}_i - x_i)^2}, \tag{3}$$

$$\text{where } w_i = \frac{i}{\sum_{j=1}^{k} j} = \frac{2i}{k(k+1)} \tag{4}$$

4.2. Evaluation of Vehicle Turn Counting

The first evaluation on the evaluation server showed promising results. However, there was still room for improvement, especially in processing speed. We experimented with different YOLO detector thresholds. The baseline experiment (Sub. 001) with detection threshold 0.25 showed effectiveness 0.8993. Efficiency score 0.0000 should not be taken into account as the solution was not running on our testing machine at that time.

After first submission, we fine-tuned position and velocity parameters in the Kalman filter of the Deep SORT tracker. After tuning those parameters and changing detector threshold to 0.20, the effectiveness increased to 0.9094

There were still detection switches between cars and trucks, so we experimented with different detector thresholds for trucks, but the best effectiveness was achieved by using the threshold from the baseline experiment.

Complete list of experiments with detection tresholds or another setting together with an official evaluation can be found in Table 1. Our final ranking is then showed in Table 2.

5. Conclusions

We participated in one task of the NVIDIA AI City Challenge 2020: Vehicle Counts by Class at Multiple Intersections.

Our solution for vehicle counting is based on convolutional neural network detection and CNN feature-based tracking. Created vehicle trajectories are then matched to pre-defined movements of interests (travel directions) using user-defined entrance/exit areas. Proposed solution achieves solid results with score $S1_{total} = 0.8829$, which put us at 6th position in final ranking on dataset part *A*.

Acknowledgements

This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II); project IT4Innovations excellence in science – LQ1602. Also, this work was supported by TACR project "SMARTCarPark", TH03010529.

Submission	Detection thresholds cars / trucks	End area distance tolerance	mwRMSE	S1_Effectiveness	S1_Efficiency	S1_Score
001	0.25 / 0.60	100 px	4.8219	0.8993	0.0000	0.6295
002	0.25 / 0.60	100 px	5.5757	0.8820	0.6725	0.8192
003	0.20 / 0.60	100 px	4.4035	0.9077	0.6776	0.8387
004	0.20 / 0.60	100 px	4.4063	0.9076	0.6666	0.8353
005	0.20 / 0.60	50 px	4.4324	0.9079	0.6669	0.8356
006	0.20 / 0.50	50 px	4.4161	0.9076	0.6660	0.8352
007	0.20 / 0.70	50 px	5.5757	0.8820	0.6671	0.8176
008	0.20 / 0.60	50 px	4.3616	0.9094	0.7868	0.8726
009	**0.20 / 0.60**	**50 px**	**4.3616**	**0.9094**	**0.8212**	**0.8829**

Table 1. Evaluation of challenge Track 1 – Vehicle Counts by Class at Multiple Intersections for individual submitted versions. For Submission 001 the Efficiency Base Factor $= 0.506436$. For the rest of the submission the Efficiency Base Factor $= 1.205131$

Rank	Team ID	Team Name	Score
1	99	Everest	0.9389
2	110	CSAI	0.9346
3	92	INF	0.9292
4	26	Orange-Control	0.8936
5	22	psl2020	0.8852
6	**74**	**GRAPH@FIT BUT**	**0.8829**
7	6	KISTI	0.8540
8	119	PES	0.8254
9	80	HCMUS	0.8064
10	65	BUPT-MCPRL	0.7933
11	40	Insight-DCU	0.7785
12	70	CUIP	0.6922
13	75	Albany_NCCU	0.3116

Table 2. Final team ranking for public part of Track 1 – Vehicle Counts by Class at Multiple Intersections. Our team is highlighted by bold text.

References

[1] Saeed Amirgholipour, Xiangjian He, Wenjing Jia, Dadong Wang, and Lei Liu. Pdanet: Pyramid density-aware attention net for accurate crowd counting. *arXiv preprint arXiv:2001.05643*, 2020.

[2] Margrit Betke, Esin Haritaoglu, and Larry S Davis. Multiple vehicle detection and tracking in hard real-time. In *Proceedings of Conference on Intelligent Vehicles*, pages 351–356. IEEE, 1996.

[3] Margrit Betke, Esin Haritaoglu, and Larry S Davis. Real-time multiple vehicle detection and tracking from a moving vehicle. *Machine vision and applications*, 12(2):69–83, 2000.

[4] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649, 2016.

[5] Lokesh Boominathan, Srinivas SS Kruthiventi, and R Venkatesh Babu. Crowdnet: A deep convolutional network for dense crowd counting. In *Proceedings of the 24th ACM international conference on Multimedia*, pages 640–644, 2016.

[6] Ke Chen, Shaogang Gong, Tao Xiang, and Chen Change Loy. Cumulative attribute space for age and crowd density estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2467–2474, 2013.

[7] Ke Chen, Chen Change Loy, Shaogang Gong, and Tony Xiang. Feature mining for localised crowd counting. In *BMVC*, volume 1, page 3, 2012.

[8] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. *NIPS*, 2016.

[9] MP Daigavane and P Bajaj. Real time vehicle detection and counting method for real time vehicle detection and counting method for unsupervised traffic video on highways unsupervised traffic video on highways. *IJCSNS*, 10(8):112, 2010.

[10] Paulo RL De Almeida, Luiz S Oliveira, Alceu S Britto Jr, Eunelson J Silva Jr, and Alessandro L Koerich. Pklot–a robust dataset for parking lot classification. *Expert Systems with Applications*, 42(11):4937–4949, 2015.

[11] R. B. Girshick. Fast r-cnn. *ICCV*, 2015.

[12] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR*, 2014.

[13] Ricardo Guerrero-Gómez-Olmedo, Beatriz Torre-Jiménez, Roberto López-Sastre, Saturnino Maldonado-Bascón, and Daniel Onoro-Rubio. Extremely overlapping vehicle counting. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 423–431. Springer, 2015.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[15] Meng-Ru Hsieh, Yen-Liang Lin, and Winston H Hsu. Drone-based object counting by spatially regularized regional proposal network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4145–4153, 2017.

[16] Michael Jones and Paul Viola. Fast multi-view face detection. *Mitsubishi Electric Research Lab TR-20003-96*, 3(14):2, 2003.

[17] Roman Juránek, Adam Herout, Markéta Dubská, and Pavel Zemčík. Real-time pose estimation piggybacked on object detection. In *ICCV*, 2015.

[18] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[19] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. *ECCV*, 2016.

[20] Weizhe Liu, Mathieu Salzmann, and Pascal Fua. Context-aware crowd counting. In *Proceedings of the IEEE Con-*

ference on Computer Vision and Pattern Recognition, pages 5099–5108, 2019.

[21] Daniel Onoro-Rubio and Roberto J López-Sastre. Towards perspective-free object counting with deep learning. In *European Conference on Computer Vision*, pages 615–629. Springer, 2016.

[22] Viet-Quoc Pham, Tatsuo Kozakaya, Osamu Yamaguchi, and Ryuzo Okada. Count forest: Co-voting uncertain number of targets using random forest for crowd density estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3253–3261, 2015.

[23] Chomtip Pornpanomchai, Thitinut Liamsanguan, and Vissakorn Vannakosit. Vehicle detection and counting from a video frame. In *2008 International Conference on Wavelet Analysis and Pattern Recognition*, volume 1, pages 356–361. IEEE, 2008.

[24] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CVPR*, 2016.

[25] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.

[26] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE PAMI*, 2016.

[27] N. Seenouvong, U. Watchareeruetai, C. Nuthong, K. Khongsomboon, and N. Ohnishi. A computer vision based vehicle detection and counting system. *2016 8th International Conference on Knowledge and Smart Technology (KST)*, 2016.

[28] Miaojing Shi, Zhaohui Yang, Chao Xu, and Qijun Chen. Revisiting perspective information for efficient crowd counting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7279–7288, 2019.

[29] G. N. Swamy and S Srilekha. Vehicle detection and counting based on color space model. *2015 International Conference on Communications and Signal Processing (ICCSP)*, 2015.

[30] Zheng Tang, Milind Naphade, Ming-Yu Liu, Xiaodong Yang, Stan Birchfield, Shuo Wang, Ratnesh Kumar, David Anastasiu, and Jenq-Neng Hwang. CityFlow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification. In *Proc. CVPR*, pages 8797–8806, 2019.

[31] N Wojke, A Bewley, and D Paulus. Simple online and realtime tracking with a deep association metric. *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468, 2017.

[32] Qi Zhang and Antoni B Chan. Wide-area crowd counting via ground-plane density maps and multi-view fusion cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8297–8306, 2019.

[33] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 589–597, 2016.

[34] Zhikang Zou, Huiliang Shao, Xiaoye Qu, Wei Wei, and Pan Zhou. Enhanced 3d convolutional networks for crowd counting. *arXiv preprint arXiv:1908.04121*, 2019.