Towards Real-time Traffic Movement Count and Trajectory Reconstruction Using Virtual Traffic Lanes

Awad Abdelhalim, Montasir Abbas
Department of Civil and Environmental Engineering
Virginia Polytechnic Institute and State University
Blacksburg, Virginia, USA
atarig@vt.edu, abbas@vt.edu

Abstract

In this paper, we discuss our framework and observations for AI City Challenge Track 1: Vehicle Counts by Class at Multiple Intersections. The framework we propose utilizes creating virtual traffic lanes for the movements of interest. Using a Python Graphical User Interface (GUI), the entry polygons for the movements of interest are identified. This leads to labeling the trajectories for the vehicles that have been first detected entering the region of interest via those entry polygons. Those vehicles, forming what we refer to as "virtual traffic lanes" inside the region of interest, are then used as identifiers for other vehicles detected further downstream using a nearest neighbors search. The framework we propose can run as an additional layer to any multi-object tracker with minimal additional computation. Our results and evaluation for the challenge track indicate the high potential of our proposed framework, and showcase the momentous value of incorporating domain knowledge in computer-vision applications.

1. Introduction

Car crashes are one of the leading causes of death around the world and in the United States, where they cost around 40,000 lives every year. The field of Intelligent Transportation Systems (ITS) aims to improve traffic safety through transforming our transportation systems as we know them from being passive and unaware to smart, dynamic, proactive systems.

One of the major fields of ITS is the recognition and trajectory tracking of vehicle movements in roadways and freeways. Obtaining reliable vehicle trajectories plays a crucial role in various aspects of ITS; including and not limited to dynamic urban traffic signal timing, behavioral modeling, and active traffic management. With the ever-growing use of traffic cameras for traffic monitoring and dynamic signal timing applications, tremendous amounts of traffic video data became available. This data raised the need for developing reliable computer-vision applications to extract information from this video data. This is a very difficult and challenging task due to the image quality and area of view provided by on-site cameras, occlusion in high traffic, and privacy concerns regarding the vehicles involved.

In a typical controlled traffic intersection in the United States, the traffic movement assignment for the signal controller follows the National Electrical Manufacturers Association (NEMA) phases shown in Fig. 1 [1]. In a recent study [2], we proposed incorporating this prior knowledge of traffic movement patterns in vehicle trajectory tracking framework, and the results of our exploratory analysis on a small dataset we collected and annotated indicated the effectiveness of our framework in obtaining reliable turn counts and addressing the vehicle re-identification problem due to occlusion in traffic. In this paper, we discuss our framework and the results we obtained applying it to this year's AI City Challenge Track 1: Vehicle Counts by Class at Multiple Intersections.

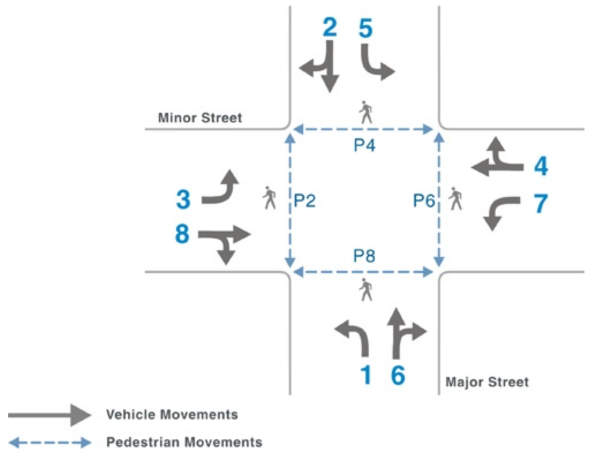


Figure 1. Typical 8-phase traffic controller operation [1].

1.1. Object Recognition and Tracking in Transportation and Traffic Safety Applications

Object detection and recognition problems have been classically tackled by a myriad of computer-vision techniques and manipulations, alongside the use knowledge about the scenery, geometry, and the emergence of deformable part models [3], [4], [5]. Momentous recent advancements have been accomplished with the evolution of Convolutional Neural Networks (CNNs), now forming the basis of majority of the state-of-the-art object detection and tracking algorithms. The introduction of Fast Region-based Convolutional Neural Network (Fast R-CNN) [6], followed by Faster R-CNN in 2015 [7] moved the fields closer towards real-time application of object detection and tracking, which quickly became a reality with the introduction of You Only Look Once (YOLO) and Single-Shot Detector (SSD) models in 2016 [8], [9]. Exponential improvements in those models have followed in recent years [10], [11], [12], [13].

In the fields of transportation and traffic engineering, vehicle detection and tracking is a research interest that has been around for a while. Studies in the early 1990s have been conducted to assess various techniques to identify, track and count automobiles [14], [15], [16]. A vast majority of those earlier techniques failed in congested traffic conditions due to partial and full occlusion, until models that track sub-features rather than full vehicle features were introduced [17] , [18], [19]. This set up the foundation for various future studies in the field [20], [21], [22].

The aforementioned recent advancements in deep learning and the rise of GPU computing led to real-time performance [23], [24], [25]. Albeit these improvements in terms of accuracy and speed, vehicle tracking in heavy traffic conditions still remains a very challenging task. Where legal around the world, multiple studies have utilized license-plate recognition as a better and more efficient way of tracking vehicles instead of the overall vehicle features [26], [27]. While others made use of video data taken from Unmanned Aerial Vehicles (UAVs) to completely overpass the occlusion problem [28], [29]. To help in developing more reliable algorithms, the field has pushed in recent years to make multiple benchmark datasets publicly available [30], [31].

2. Methodology

2.1. Base Object Tracker

For our previous study, we implemented a combination of YOLO v3 [13, 32] and DeepSORT [33] algorithms. We used the same combination for our base tracker for this challenge, both pre-trained on the Microsoft Common Objects in Context (COCO) dataset [34], running at $\sim$13 fps on a machine equipped with NVIDIA GTX 1080 GPU and a 2.4 GHz 16-core Intel Xeon e5 2630 v3 processor.

2.2. Predefined Movement-Based Virtual Traffic Lanes

In our previous study, we created a Python Graphical User Interface (GUI) that allows the user to define polygons representing entry planes to each NEMA movement for vehicles entering an urban intersection. The same GUI was used for this challenge, where the number of movements and entry polygons is adjustable for each camera location. We utilized homography transformation to obtain bird-eye view of region of interest (ROI), which can also be defined using the GUI but specific ROIs were provided for this challenge and hence were used. The definition of entry polygons can be carried out in the normal camera space or perspective transformed space. Vehicles for which the first detection occurred inside the movement polygon receive a label of that movement, as multiple vehicles carrying those labels move across the ROI they create what we refer to as virtual traffic lanes within the ROI, and those vehicles are hence referred to as *Virtual Lane Vehicles (VLV)*. Late detections past entry polygon can occur simply due to the late identification of a vehicle, but can also happen as a result of identity switches to occlusion in heavy traffic, lights and shadows, or the infrastructure. As we are using a pre-trained model on regular images, all the detection tasks were carried out in the regular camera space while the post-processing of vehicle trajectories is carried out in the transformed space. The perspective transformation and calculation of homography transformation matrix (M) were carried out using OpenCV library [35]. The x and y coordinates in destination space $destination(x, y)$ for all points on the regular image space are transformed using the following equation:

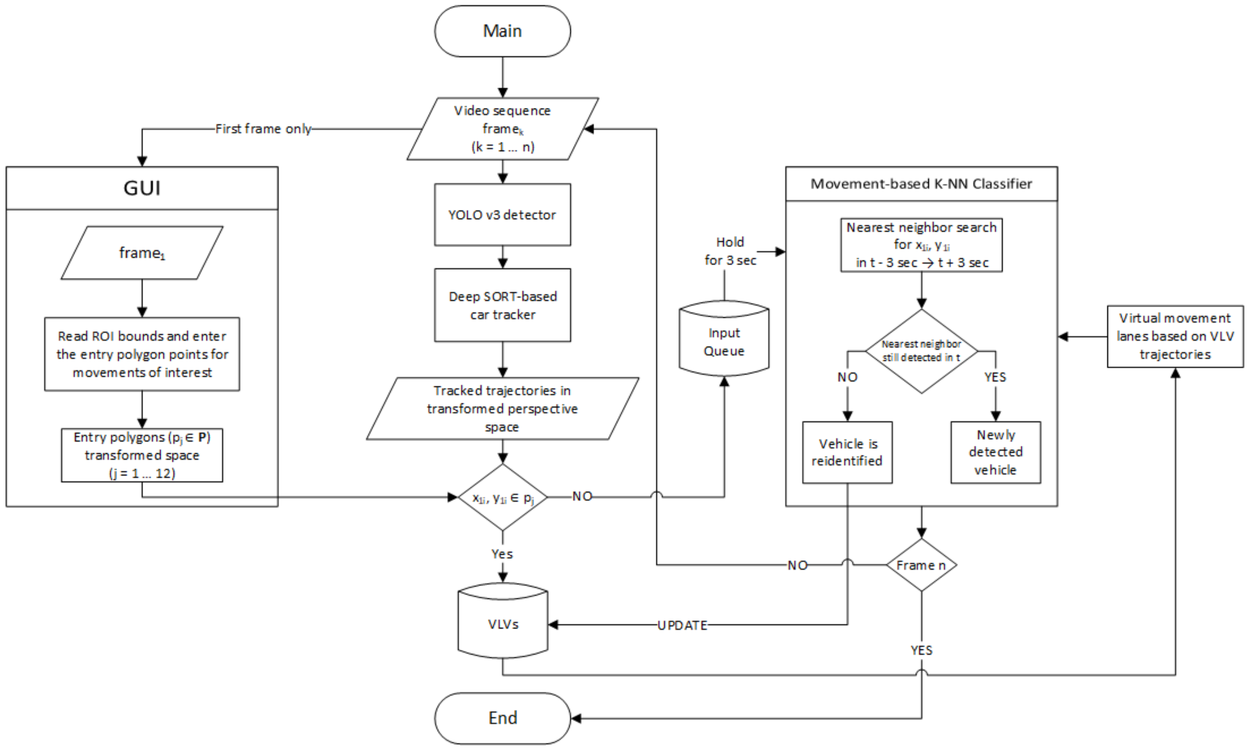
$$source\left(\frac{M_{11}x + M_{12}y + M_{13}}{M_{31}x + M_{32}y + M_{33}}, \frac{M_{21}x + M_{22}y + M_{23}}{M_{31}x + M_{32}y + M_{33}}\right) \tag{1}$$

Where:
x & y = are the coordinates for each point in the source space from the camera video frame.
M = 3×3 homography matrix.

$$Movement_i = \begin{cases} label\ j, & \text{if } x_{1i}, y_{1i} \in p_j \quad \forall_{i,j} \\ unknown, & \text{otherwise} \end{cases} \tag{2}$$

Where:
$Movement_i$ = movement label for vehicle i.
x_{1i} & y_{1i} = the coordinates for the first point in the trajectory of each detected vehicle i.
$label\ j$ = predefined movement of interest for each camera location $j \in \mathbb{Z} : 1 \leq j \leq 12$.
p_j= entry polygon for movement of interest j.

Fig. 2 provides an overview of the data flow in our framework. It is worth mentioning that while the use of en-



Main
Video sequence frame$_k$ (k = 1 … n)
First frame only
GUI
frame$_1$
Read ROI bounds and enter the entry polygon points for movements of interest
Entry polygons (p$_j$ ∈ P) transformed space (j = 1 … 12)
YOLO v3 detector
Deep SORT-based car tracker
Tracked trajectories in transformed perspective space
x$_{1i}$, y$_{1i}$ ∈ p$_j$
NO
Yes
VLVs
UPDATE
Input Queue
Hold for 3 sec
Movement-based K-NN Classifier
Nearest neighbor search for x$_{1i}$, y$_{1i}$ in t - 3 sec → t + 3 sec
Nearest neighbor still detected in t
NO
YES
Vehicle is reidentified
Newly detected vehicle
Virtual movement lanes based on VLV trajectories
Frame n
NO
YES
End

Figure 2. Flow chart of the framework's data flow.

try polygons to identify the movements of interest provides an efficient way of labeling the vehicle trajectory and movement patterns, it is insensitive of vehicles conducting illegal movements within the area of interest (e.g. in the case of an intersection, a vehicle that turns left albeit entering from a thru-only lane).

Algorithm 1: Movement-Based K-Nearest Neighbor Classifier

```
Result: Identifying the movement label for vehicles
        where the first detection did not take place
        within one of the entry polygons.
initialization;
for $frame_k$ do
    for $x_{1i}, y_{1i} \notin P$ do
        $neighbors = VLVs \in \{frame_{k-30} \cdots frame_{k+30}\}$;
        $NEMA_i = majority\ vote$;
        if $nearest \in frame_{k-10}$ & $\notin$
          $\{frame_k \ldots frame_n\}$ then
            $ID_i =$
              $ID_{nearest} \iff NEMA_i = NEMA_{nearest}$
        else
            $ID_i = ID_{detector}$
        end
    end
end
```

Algorithm 1 describes the K-NN search for the vehicles with unidentified movements. The number of nearest neighbors is the minimum of vehicles in the search space and 3. In our previous study (which was only limited to an urban intersection environment), we allowed a 3-second delay before classifying newly identified vehicles to account for unidentified vehicles that enter the ROI at the beginning of video sequence or a signal green (in case of an intersection), where no other vehicles are in the ROI. A search back time $t-1$ second allows to identify re-identifications due to occlusion. The same parameters were used in for this challenge. Vehicles' movement label is predicted by a majority vote of VLVs within the given temporal range. Weighted euclidean distance is the metric in use. An identity switch is recognized when the nearest neighbor of vehicle$_i$ is present before but no longer detected on or after $frame_k$. Note that the given algorithm description is for video sequences with frame rate of 10 fps. Look-back frames are adjusted accordingly for videos with higher and lower frame rates.

2.3. AI City Challenge Dataset

The dataset provided for this challenge contained 31 video sequences totaling over 9 hours (4 of which were reserved for testing) from 20 different camera locations. Specific movements and ROIs were predefined for each camera location. Fig. 3 shows one of the camera locations for which a subset of the ground truth labels were provided.

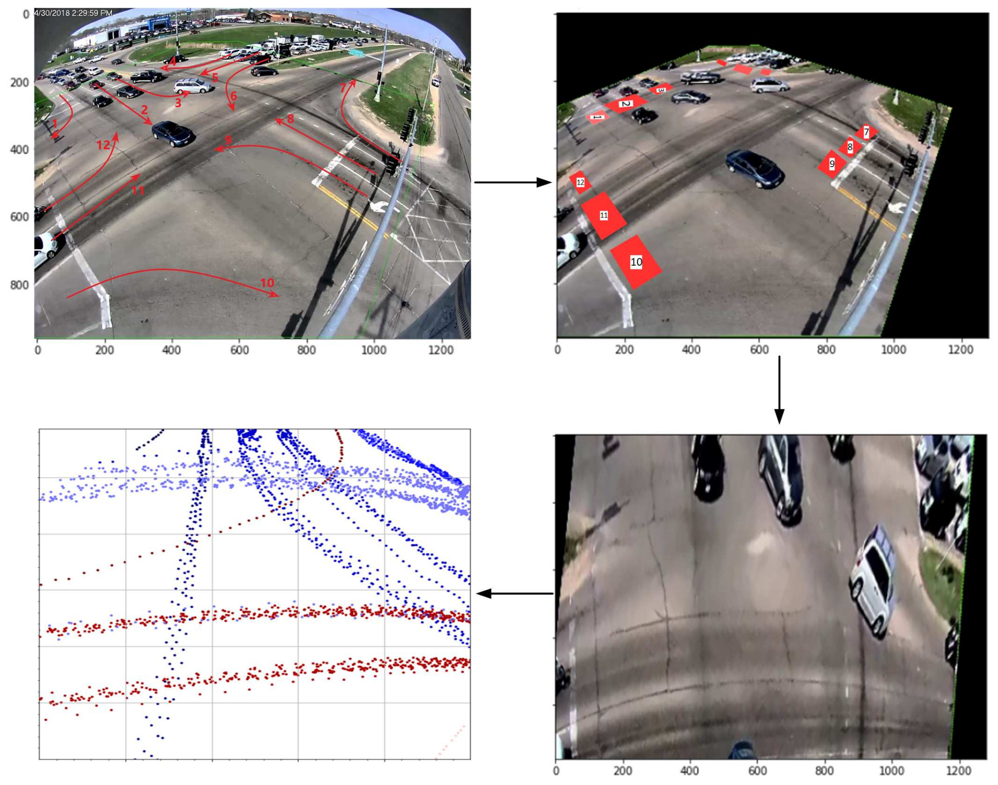


Figure 3. Example of video data processing pipeline in our framework.

The figure shows the video data processing pipeline in our framework, where the given ROIs are used to define the detection and tracking area, then the user manually defines polygons corresponding to each movement of interest, after which virtual movement lanes obtained via perspective transformation. In the post-processing of detected trajectories for intersection scenes, we only include the trajectories within the intersection (past the stop lines) to avoid storing data for stationary vehicles awaiting green light, hence all trajectory data starts past the stop lines where the entry polygons are defined. We initially evaluated our the accuracy of our framework on this given subset by evaluating the count accuracy for each of the 12 movement labels in that given intersection.

The conglomerated results from all 31 video sequences were then uploaded to the challenge's evaluation server, which evaluates the overall results on effectiveness and the computational efficiency of the algorithm. The final $S1$ score is a combination of both.

$$S1 = \alpha S1_{efficiency} + \beta S1_{effectiveness} \tag{3}$$

Where: α = 0.3 β = 0.7

$$S1_{efficiency} = \max\left(0, 1 - \frac{time \times base\ factor}{5 \times video\ total\ time}\right) \tag{4}$$

Where:
$time$ and $video\ total\ time$ = respectively the execution time and total video time for the given dataset in sec.
$base\ factor$ = score obtained through a Python script provided with the dataset.

$$wRMSE = \sqrt{\sum_{i=1}^{k} w_i(\hat{x_i} - x_i)^2} \tag{5}$$

Where:

$$w_i = \frac{i}{\sum_{j=1}^{k}} = \frac{2i}{k(k+1)} \tag{6}$$

The $S1_{effectiveness}$ score is calculated based on the weighted average of normalized weighted root mean square error across all video sequences, movements, and vehicle classes.

3. Results and Evaluation

Table 1 shows the movement counts given for the provided sample video sequence and the total and TP counts obtained for each movement through our framework. Fig 4 shows the trajectories and their respective predicted labels.

Table 2 shows the results from the evaluation system. We selected two runs where we used slightly different thresholds for re-identification of vehicles within a virtual lane.

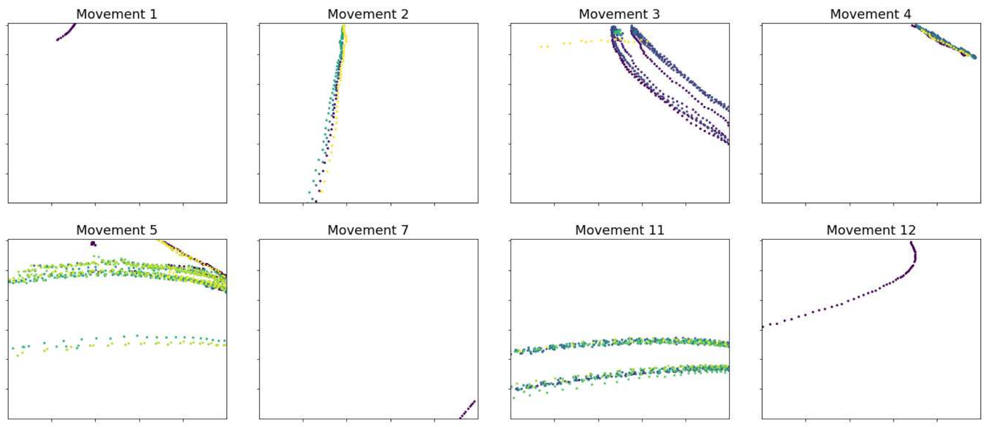


Figure 4. Detected vehicle trajectories in the sample video sequence and their predicted movement labels.

Run 1 assumes that an identity switch occurs *only* when the nearest neighbor of vehicle i is no longer detected in the following frames, where as in run 2 it could be anyone of the k-nearest neighbors.

Although our code currently distinguishes between car and truck classes, the version of the code that was used before the closure of the evaluation server did not. Hence, these evaluations were affected by the misclassification of all trucks in the dataset.

Table 1. The Ground Truth, Obtained Total and True Positive Counts for All Movements in the Provided Sample Video Sequence

Movement	Ground Truth	Total Count	TP
1	1	1	1
2	5	4	4
3	13	14	12
4	10	8	8
5	25	32	25
6	0	0	0
7	1	1	1
8	0	0	0
9	0	0	0
10	2	0	0
11	38	36	36
12	1	1	1

Table 2. Select Results From the Evaluation Server

Run	wRMSE	Effectiveness	Efficiency	S1
1	12.2	0.767	0.906	0.809
2	11.87	0.774	0.906	0.814

Fig. 5 shows an example of a resolved identity switch using our framework. Vehicle 108 was re-identified as 124 as it moves in traffic due to partial occlusion. The output trajectory of vehicle 108 via our framework includes that of 124, and vehicle vehicles 124 is no longer reported as a separate trajectory. A total of 8 out of 12 similar cases were resolved in the sample video sequence. Considering that this was only a 1-minute sequence, this gives an indication of the value our framework can add when obtaining trajectories from large video datasets.

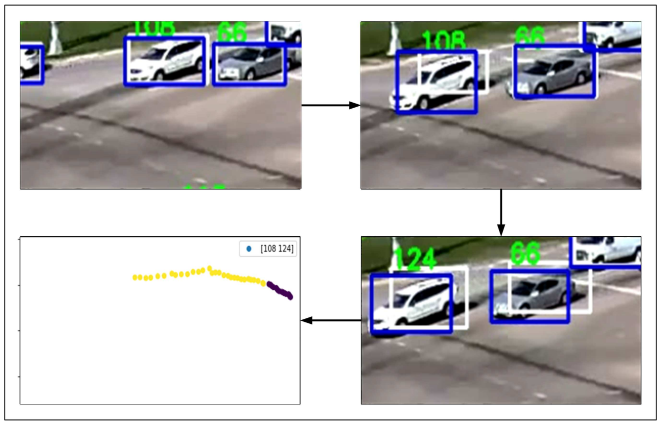


Figure 5. Example of resolved ID switch.

4. Conclusions and Discussion

We attempted the AI City Challenge Track 1: Vehicle Counts by Class at Multiple Intersections by implementing a simple framework utilizes our prior knowledge of traffic movement patterns to identify vehicle movements, obtain real-time counts and resolve identity switches that occur due to occlusion and detection errors. While the movements of interest are predefined for camera locations in this challenge, in general the traffic patterns in a certain road segment or through an intersection do follow similar predefined patterns, and providing practitioners with a simple GUI to identify those movements of interest to obtain reliable traffic counts, vehicle trajectories and speed profiles can significantly aid in developing ITS application of origin-destination estimation, conflict monitoring, incident detection, and dynamic signal timing. Our GUI can also be used to further isolate lanes for multi-lane movements, allowing to obtain traffic density and lane occupancy.

The K-NN search within virtual lanes is very efficient, running up to 100 HZ in low traffic sequences and between 40-60 HZ in high traffic on our machine, which is a minimal increase when compared to the running time of a multi-object tracking network. Our effectiveness can be improved after resolving a code error that went unnoticed which led to the misclassification of trucks as cars in our output files. Since our pipeline mainly utilizes homography transformation to isolate vehicle movement lanes and obtain those virtual lanes, the lack of knowledge of camera and lens specifications and angle prevented us from accounting for distortion, which is even more impactful in the case of fish-eye lens sequences.

Moving forward, we will be training our baseline tracker (which as only pre-trained on COCO) on datasets that are tailored for traffic detection or using other models that have proven to be more efficient for traffic detection. Having access to the full ground truths of the dataset following the competition will also allow us to identify and improve the shortcomings of our framework, and would provide the opportunity to optimize the different parameters including the extent of the ROI for perspective transformation, the look back and look ahead time frames within virtual lanes, and the number of neighbors to considered in the K-NN search.

References

[1] U. FHWA, “Traffic signal timing manual,” 2008.

[2] A. Abdelhalim and M. Abbas, “Vt-lane: An exploratory study of an ad-hoc framework for real-time intersection turn count and trajectory reconstruction using nema phases-based virtual traffic lanes,” *Under Review. Submitted to The 23rd IEEE International Conference on Intelligent Transportation Systems*.

[3] P. Felzenszwalb, D. McAllester, and D. Ramanan, “A discriminatively trained, multiscale, deformable part model,” in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, IEEE, 2008.

[4] M. C. Burl, M. Weber, and P. Perona, “A probabilistic approach to object recognition using local photometry and global geometry,” in *European conference on computer vision*, pp. 628–641, Springer, 1998.

[5] D. Crandall, P. Felzenszwalb, and D. Huttenlocher, “Spatial priors for part-based recognition using statistical models,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 1, pp. 10–17, IEEE, 2005.

[6] R. Girshick, “Fast r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[7] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Advances in neural information processing systems*, pp. 91–99, 2015.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[10] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[11] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, “Dssd: Deconvolutional single shot detector,” *arXiv preprint arXiv:1701.06659*, 2017.

[12] J. Redmon and A. Farhadi, “Yolo9000: better, faster, stronger,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263–7271, 2017.

[13] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.

[14] K. Karmann and A. Brandt, “Moving object recognition using an adaptive background memory. cappellini v, ed. time-varying image processing and moving object recognition,” 1990.

[15] T. Huang, D. Koller, J. Malik, G. Ogasawara, B. S. Rao, S. J. Russell, and J. Weber, “Automatic symbolic traffic scene analysis using belief networks,” in *AAAI*, vol. 94, pp. 966–972, 1994.

[16] D. Koller, K. Daniilidis, and H.-H. Nagel, "Model-based object tracking in monocular image sequences of road traffic scenes," *International Journal of Computer 11263on*, vol. 10, no. 3, pp. 257–281, 1993.

[17] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik, "A real-time computer vision system for measuring traffic parameters," in *Proceedings of IEEE computer society conference on computer vision and pattern recognition*, pp. 495–501, IEEE, 1997.

[18] D. Beymer and J. Malik, "Tracking vehicles in congested traffic," in *Proceedings of Conference on Intelligent Vehicles*, pp. 130–135, IEEE, 1996.

[19] S. Kamijo, Y. Matsushita, K. Ikeuchi, and M. Sakauchi, "Traffic monitoring and accident detection at intersections," *IEEE transactions on Intelligent transportation systems*, vol. 1, no. 2, pp. 108–118, 2000.

[20] S. Messelodi, C. M. Modena, and M. Zanin, "A computer vision system for the detection and classification of vehicles at urban road intersections," *Pattern analysis and applications*, vol. 8, no. 1-2, pp. 17–31, 2005.

[21] N. Saunier and T. Sayed, "A feature-based tracking algorithm for vehicles in intersections," in *The 3rd Canadian Conference on Computer and Robot Vision (CRV'06)*, pp. 59–59, IEEE, 2006.

[22] A. Sanchez, P. D. Suarez, A. Conci, and E. Nunes, "Video-based distance traffic analysis: Application to vehicle tracking and counting," *Computing in Science & Engineering*, vol. 13, no. 3, pp. 38–45, 2010.

[23] M. S. Shirazi and B. T. Morris, "Trajectory prediction of vehicles turning at intersections using deep neural networks," *Machine Vision and Applications*, vol. 30, no. 6, pp. 1097–1109, 2019.

[24] C.-E. Wu, W.-Y. Yang, H.-C. Ting, and J.-S. Wang, "Traffic pattern modeling, trajectory classification and vehicle tracking within urban intersections," in *2017 International Smart Cities Conference (ISC2)*, pp. 1–6, IEEE, 2017.

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

[26] L. Zhu, S. Wang, C. Li, and Z. Yang, "License plate recognition in urban road based on vehicle tracking and result integration," *Journal of Intelligent Systems*, vol. 29, no. 1, pp. 1587–1597, 2019.

[27] C. N. E. Anagnostopoulos, I. E. Anagnostopoulos, V. Loumos, and E. Kayafas, "A license plate-recognition algorithm for intelligent transportation system applications," *IEEE Transactions on Intelligent transportation systems*, vol. 7, no. 3, pp. 377–392, 2006.

[28] M. A. Khan, W. Ectors, T. Bellemans, Y. Ruichek, D. Janssens, G. Wets, *et al.*, "Unmanned aerial vehicle-based traffic analysis: A case study to analyze traffic streams at urban roundabouts," *Procedia computer science*, vol. 130, pp. 636–643, 2018.

[29] H. Zhang, M. Liptrott, N. Bessis, and J. Cheng, "Real-time traffic analysis using deep learning techniques and uav based video," in *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–5, IEEE, 2019.

[30] Z. Tang, M. Naphade, M.-Y. Liu, X. Yang, S. Birchfield, S. Wang, R. Kumar, D. Anastasiu, and J.-N. Hwang, "Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8797–8806, 2019.

[31] L. Wen, D. Du, Z. Cai, Z. Lei, M.-C. Chang, H. Qi, J. Lim, M.-H. Yang, and S. Lyu, "Ua-detrac: A new benchmark and protocol for multi-object detection and tracking," *Computer Vision and Image Understanding*, p. 102907, 2020.

[32] J. Redmon, "Darknet: Open source neural networks in c." `http://pjreddie.com/darknet/`, 2013–2016.

[33] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE international conference on image processing (ICIP)*, pp. 3645–3649, IEEE, 2017.

[34] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014.

[35] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.