

Multi-Class, Multi-Movement Vehicle Counting on Traffic Camera Data

Vineet Shenoy*, and Rama Chellappa
Johns Hopkins University

{vshenoy4, rchella}@jhu.edu,

Abstract

Multi-Class, multi-movement vehicle counting is essential for city-scale intelligent transportation. Understanding traffic patterns is critical to proper timing of signals for congestion mitigation and usage along certain corridors. This paper presents an end-to-end pipeline for real-time multi-class, multi-movement vehicle counting. We match the tracking results from off-the-shelf trackers to movements represented as parametric curves, outputting results in an online fashion. Our novel contribution is the definition of vehicle actions and the projection of vehicle trajectories in space onto potential actions. We submit our proposed results to the AICITY 2021 evaluation server, where we place in the top 15. The code for the results can be found at <https://github.com/vineetrshenoy/VehicleCounting>

1. Introduction

In recent years, the proliferation of traffic cameras has produced a constant stream of vehicle-related data that can be used for intelligent transportation systems. This data can be used for tasks such as speed estimation, dynamic traffic routing, multi-camera tracking, traffic anomaly detection, and more. In addition, with the recent advances of Deep Convolutional Neural Networks (DCNNs) [5], significant improvements in classification, recognition, and detection have led to corresponding improvements in traffic analytics. It is expected that traffic systems will continue to develop given current algorithms and new vehicle-specific datasets.

The problem of multi-class, multi-movement vehicle counting seeks to detect vehicles, classify them into object categories, and assign them to pre-defined movements as they exit a scene. Classified actions can include passing through an intersection, making right or left turns, and even U-turns. This involves building robust algorithms that classify vehicle movements as vehicles stop at signals, occlude and be occluded by other vehicles, and more. It also includes maintaining performance under adverse lighting and

weather conditions such as rain and snow. The AI City Challenge dataset provides this variety through video feeds captured at 20 different intersection and highway scenes.

We propose an algorithm that efficiently classifies movements and vehicles on this dataset. Our contributions include:

- Creating an end-to-end pipeline that ingests raw video and classifies various movements; pipeline components include a *detection* module, *tracking* module, and *matching* module.
- A novel definition on a vehicle movement (i.e. “left-turn”, “right-turn”) using parametric curves in image space.
- A simple and fast matching algorithm that associates the trajectory of a vehicle with a certain movement.

We build this pipeline in three stages – a detection stage, a tracking stage, and a counting stage.

The rest of the paper is organized as follows: Section 2 discusses related works for vehicle counting. Section 3 discusses our approach and the mathematics behind our approach. Section 4 discusses the parameters used to implement the algorithm. Finally, we conclude in Section 5 and discuss some future work.

2. Related Works

Successfully matching vehicles to movements requires correct implementation of detection, tracking, and matching modules. Previous work in this area is described below.

Detection: Detection is a critical first step for most algorithms. The AI City evaluation metric, described in section 4, measures run-time performance of algorithms, so previous competitors [1], [3] chose the YOLOv3 detector [10]. Though the YOLOv3 detector does not achieve the state-of-the-art mAP scores on datasets like MSCOCO, it compromises accuracy for speed as it is a one-stage detector. Others, such as [7] [9] used two-stage region proposal networks like Faster R-CNN [12] with Feature Pyramid Networks to

capture multi-scale information. Accounting for the variety of ID switches when occluded, [7] improve their detections with a re-match and a single object tracking strategy.

Tracking: Another fundamental task is tracking; this is critical for correct and robust matching. Many competitors such as [1], [3] used DeepSORT [15], which integrates both appearance and motion information, as opposed to SORT [2], which only uses motion information. Both algorithms use bounding box information from previous video frames to estimate the motion of objects in the scene; DeepSORT also considers feature descriptors of objects across frames to associate global IDs to unique vehicles, and uses the cosine distance to match the feature descriptors for each vehicle. While a step like this may improve the accuracy of the tracker, it does pay a penalty in terms of speed.

Both trackers described above are considered to be “two-stage” trackers i.e. detections from a previous stage are associated to track objects globally. Other work such as [13] create one-stage trackers which perform detection and tracking simultaneously. It is noted that this tracker only performs single-class tracking.

Matching: Previous work such as [1] [4] used the concept of “entry polygons” to classify movements. After detection and tracking occurred, a vehicle entering the movement’s “entry polygon” was classified as that movement. If a vehicle was tracked but for some reason did not enter the movement polygon, then a movement-based K-NN classifier would assign a movement to that vehicle. Similarly, [3] used the notion of “distinguished regions” pairs, two linked regions for each movement, to reduce identity switches and occlusions that frequently result in erroneous movements. As opposed to using polygons, [9] used “line crossings”, where a movement is identified when crossing starting and ending lines. Using a vector from the center of the first and last tracked boxes, [9] obtain intersection and orientation information that contributes to their matching algorithm.

A variety of previous work has used trajectory-based methods. The team in [7] used computed and manually drawn trajectories as ground-truth movements, and then matched the vehicle tracklets to trajectories. As a major addition, they modified the SORT matching distance, the Mahanoblis distance, to account for non-linear motion by adding an identity matrix to the covariance term. During the counting stage they considered the vehicle track as points in space and use the Hausdorff distance to match the points to a trajectory. These modifications helped this team earn the top spot on the leaderboard.

3. Approach

In this section we present our proposed approach, **Parametric Curve Tracking (PACT)**, for Multi-Class, Multi-Movement vehicle matching. The pipeline consists of three modules: a detection module, tracking module, and

matching module.

3.1. Detection

A requirement of the 2021 NVIDIA AICITY challenge was to use no external datasets to improved detectors. Therefore, we used off-the-shelf detectors trained on the MS COCO [6] dataset, specifically Faster-RCNN [11]. This detector is a two-stage detector, which generates regions of interest using a Region Proposal Network, and then sends these regions downstream for classification and bounding box regression. This detector was used by the top-two performing teams at the challenge. We obtain detections per-frame, and retain only those that belong the classes of “car”, “bus”, and “truck”.

3.2. Tracking

All detections for a single video stream are inputted to the tracking algorithm; we use Simple Online and Real-time Tracking (SORT) [14]. Assuming a constant velocity model, a Kalman Filter handles the motion prediction module, while the Hungarian Algorithm handles the data association problem. To improve tracking results, DeepSort [15] was proposed, which integrates a deep visual association metric into the tracking algorithm. Successful tracking outputs a series of bounding boxes for each frame of the video along with a vehicle identifier consistent for all frames in which the vehicle appears. While the tracker in [13] is a single-class tracking module, we use the association metric and integrate visual features into the tracking results. The results from both trackers are presented in table 1.

3.3. Counting

We propose a novel counting method using parametric curves that can be used for any arbitrary vehicle trajectories. As compared to methods such as [1] [3] [4] which only account for a vehicle’s starting and ending location, we can estimate a vehicle’s *entire* trajectory.

3.3.1 Movement Definition

The first step is to mathematically define a certain movement, such as a “left turn” or a “right turn”. To do this, we use Bézier curves, first used by Pierre Bézier when designing bodywork of new vehicles. Give points $\mathbf{P}_1, \mathbf{P}_2, \dots, \mathbf{P}_n$ a Bézier curve is defined by the polynomial

$$\mathbf{B}(t) = \sum_{i=1}^n \binom{n}{i} (1-t)^{n-i} t^i \mathbf{P}_i, \quad 0 \leq t \leq 1 \quad (1)$$

The curve begins and ends at point \mathbf{P}_1 and \mathbf{P}_n , respectively, while points $\mathbf{P}_2 \dots \mathbf{P}_{n-1}$ serve as “control” points, determining any “change of direction” of the curve. A first-order

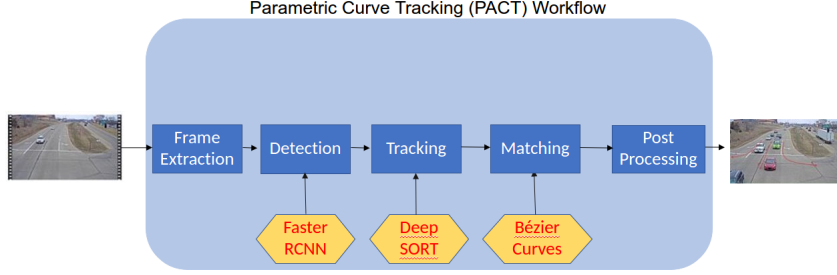


Figure 1. The pipeline for multi-class, multi-movement vehicle counting

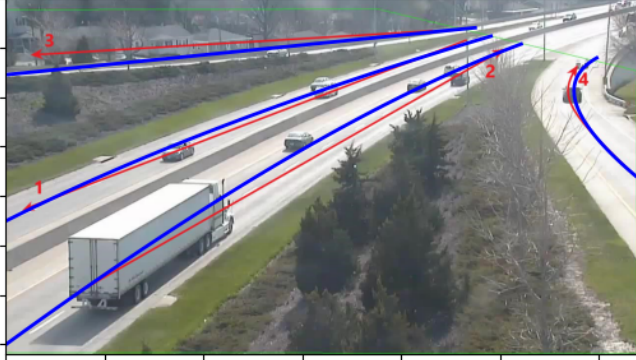


Figure 2. Parametric movement curves (blue) overlaid on camera scene and pre-defined movement curves and number (red). The green polygon is the region of interests and signals when a vehicle should be counted (upon exit). Movement 3 uses a first-order curve while movement 1,2, and 4 use second-order curves

Bézier curve (a line) has no change in direction; a second-order curve has one change of direction, a third-order curve has two changes of direction and so on. For the matching module, only a first order curve

$$C(t) = (1 - t) \cdot P_1 + t \cdot P_2 \quad (2)$$

and a second order curve

$$C(t) = (1 - t)^2 \cdot P_1 + 2 \cdot (1 - t) \cdot t \cdot P_2 + t^2 \cdot P_3 \quad (3)$$

for $0 \leq t \leq 1$, are used in the module. First-order curves are used for any “straight through” movement, while second-order curves are used for turns. After choosing point P_i appropriately in image coordinates, we can closely match the trajectory of various movements as shown in figure 2.

3.3.2 Movement Matching

The goal of matching is to find the parametric curve that most similarly resembles the trajectory of a vehicle. For each bounding box center $\langle x_i, y_i \rangle$ from the tracker, the closest point $t_{i,j}$ to Bézier curve $C_j(t)$ $\forall j$ can be found, and a distance $d_{i,j}$ from the tracker to a point on the curve is

obtained; summing $d_{i,j}$ over i for each curve results in a cumulative distance from the trajectory to the curve; the curve which has the smallest cumulative distance to the tracker points can be assigned. The full algorithm is defined in algorithm 1. We describe below how to find the closest points from the tracked vehicles to the curve for first and second order curves.

Both the tracker output, a set of bounding box centers $\langle x_i, y_i \rangle$ for $i = 1 \dots n$ per unique vehicle, and the curve itself can be viewed as a series of vectors; the vector on curve $C_j(t_r) = \langle x_r, y_r \rangle$ is the tangential direction of the curve at point t_r . Using an orthogonality argument, the point $t_{i,j}$ closest to the curve $C_j(t)$ from the tracked vehicle vector $\langle x_i, y_i \rangle$ is that point for which the distance vector is orthogonal to the direction of the curve. Re-writing equation 2 in vector notation as $C_j(t) = \langle x_1 + t \cdot (x_2 - x_1), y_1 + t \cdot (y_2 - y_1) \rangle$

$$\begin{aligned} &\langle x_1 + t_{i,j} \cdot (x_2 - x_1), y_1 + t_{i,j} \cdot (y_2 - y_1) \rangle \cdot \\ &\langle (x_1 - x_i) + t_{i,j} \cdot (x_2 - x_1), (y_1 - y_i) + t_{i,j} \cdot (y_2 - y_1) \rangle = 0 \end{aligned} \quad (4)$$

The first term in the dot product is the tangent vector of the curve. The second term is the distance vector from the point to the curve. See figure 3 for a visual understanding of the algorithm. This can easily be solved for $t_{i,j}$ in the first-order case. The second-order case involves solving a third-order polynomial and ignoring the roots that are out of the range $[0, 1]$ or selecting the root which results in the shortest distance vector to $\langle x_i, y_i \rangle$.

The above procedure is performed for all curves $C_j(t)$, and for each curve, a vector \mathbf{t}_j and \mathbf{d}_j corresponding to the points on the curve and shortest distance to each tracker point is obtained. The distance vector with the smallest cumulative sum corresponds to the movement. See algorithm 1 for a pseudo-code implementation.

3.4. Post-Processing

The majority of movements can be classified based solely by choosing the movement with the smallest cumulative distance. However, the associated vector \mathbf{t} provides

Algorithm 1: Vehicle Movement Matching

Result: Movement ID**Input:**

- N Tracker Locations, labeled $(x_j, y_j), j = 1 \dots N$ for vehicle k .
- Control Points for Curves C_i

foreach Curve C_i **do** $\mathbf{t}_i \leftarrow$ Closest point t_j on C_i to (x_j, y_j) ,
aggregated; $\mathbf{d}_i \leftarrow$ Distance d_j from C_i to (x_j, y_j) ,
aggregated;**if** t_i decreasing **then**
| $d_i \leftarrow \text{MAXINT}$;**end****end**mvt-id = $\arg \min d_i$

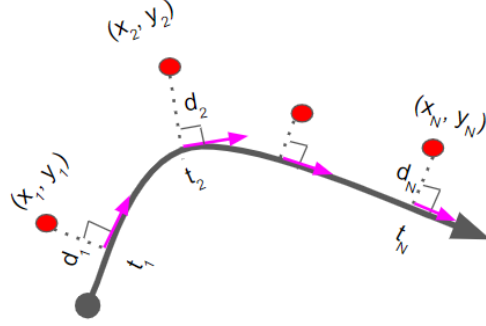


Figure 3. Visualization of the matching algorithm. The gray curve is the Bézier curve for a certain movement. For the for each point from the tracker (red), the closest point on the curve is found, as well as the corresponding distance from the point to the curve. The purple arrows are vectors show the direction of the curve at each point.

very valuable information that can be used for matching as well. Firstly, all values of t_i in \mathbf{t} must be between zero and one; a large proportion out of this range signifies that an incorrect match to that movement. Secondly, elements of \mathbf{t} must be increasing, usually monotonically. Checking this condition can be especially important for parallel paths in which a vehicle’s trajectory may be closer to an incorrect movement (due to jittering of the tracker, imprecise curve definition, etc). Consider movements 1 and 2 in figure 2, and a vehicle’s tracked trajectory that corresponds to movement 2 (ground truth). Due to noise in the tracker, the distance vector may be closer to movement 1 on the aggregate; however, the \mathbf{t} will clearly be decreasing from one to zero for movement 1 and increasing from zero to one for movement 2, unambiguously classifying this movement correctly. Clearly, \mathbf{t} encodes a notion of *direction*. Other

strange behavior corresponding to incorrect movements was observed; for example, elements of the \mathbf{t} vector increased to a certain point, then only decreased. This helps eliminate unlikely paths.

Analysis of the \mathbf{t} vector can provide insights beyond the scope of the challenge; for example, an approximate speed can be determined by differentiating the vector. Displacement can also be measured. Stationary or stopped vehicles can also be identified by viewing the highly oscillatory behavior of the \mathbf{t} vector, which has applications in anomaly detection. When designing curves on roads, civil engineers can use this information to understand how closely vehicles “hug” the intended trajectories. Entry/exit region-based methods provide no such information. Applications to all these areas can be done in a computationally efficient way.

4. Experiments and Results

4.1. Dataset

One dataset for this problem has been developed as part of the AICITY 2021 workshop [8]. The vehicle counting dataset consists of twenty different camera views from traffic cameras in Iowa, with varying lighting conditions (i.e. dawn, mid-day, etc), as well as different weather conditions (i.e. rain and snow). The majority of scenes focus on intersections, with a limited number of highway scenes for capturing on-ramp and off-ramp movements. The number of movements to classify per-scene varied from as little as two different movements to as many as twelve different movements for a busy intersection. The movement themselves included “right turn”, “left turn”, “no turn”, “highway on-ramp”, and “highway off-ramp”. Vehicle classification was split into two classes – “car” which was assigned to many smaller vehicles such as cars, minivans, pick-up trucks and mail trucks, and “trucks”, which encompassed 18-wheel vehicles, garbage trucks, and many other large vehicles. A region-of-interest (ROI) was also defined to indicate when teams should start and stop tracking of vehicles. A total of nine hours of video was captured at a resolution of 960 pixels or better and the majority of videos were captured at 10 frames per-second. As a part of the challenge, algorithms were evaluated for both “effectiveness” and efficiency; seventy-percent of the score was attributed to “effectiveness” while thirty percent of the total score was obtained from the speed of the algorithm.

4.2. Evaluation Criteria

The algorithms were evaluated for both performance/accuracy of the algorithm as well as the run-time. A combined $S1$ score was obtained a the sum of the performance score and efficiency score:

$$S1 = \alpha S1_{efficiency} + \beta S1_{effectiveness} \quad (5)$$

Tracker Type	$S1$	$S1_{effectiveness}$	$S1_{efficiency}$
SORT [14]	0.5350	0.7553	0.0209
JDE Tracker [13]	0.4155	0.5683	0.0591

Table 1. The results on the AI City Challenge

where $\alpha = 0.3$ and $\beta = 0.7$. The $S1_{efficiency}$ score was governed by the the execution time (in seconds) and the efficiency base factor, a number encapsulating the performance characteristics of the system:

$$S1_{efficiency} = \max(0, 1 - \frac{time \times basefactor}{1.1 \times video_{total}time}) \quad (6)$$

The effectiveness score was computed as the weighted average of a normalized weighted root mean square error (nWRMSE):

$$wRMSE = \sqrt{\sum_{i=1}^k w_i (\hat{x}_i - x_i)^2} \quad (7)$$

where $w_i = \frac{2i}{k(k+1)}$. Teams were scored based on the $S1$ score only. We implemented the base detector from the Detectron2 library [16].

4.3. Result

We report our scores in table 1. We used the Faster-RCNN detector as for all experiments but varied the tracker. The first tracker was SORT [14] and the second tracker was the JDE Tracker (appearance matching portion only) [13].

We found that the appearance descriptor used in [13], while providing track IDs through the frames, suffered greatly from ID switching. In the counting algorithms, this usually resulted in double counting vehicles or not counting vehicles at all (due to many broken trajectories). The SORT algorithm was more consistent in this scenario, which resulted in better performance. Using techniques from [7] for more robust trajectory matching is an important future step to further improve performance.

5. Conclusion

In this paper, we summarize our contributions to the problem of Multi-Class, Multi-Vehicle counting and show our results on the 2021 NVIDIA AI City dataset. We have proposed an end-to-end pipeline that ingests raw video, detects vehicles of different classes, tracks each unique video through the video frames, and assigns a movement to the tracked vehicle. Our novel contribution involves defining vehicle movements using Bézier and matching a vehicle’s trajectory to that of the curve. This work also sets the foundation for work in related tasks, such as vehicle speed estimation and anomaly detection. Future work would include

improving the tracking results by limiting ID switching and stitching broken trajectories.

6. Acknowledgement

This research is supported in part by Anticipatory Ground-Level Image Analytics (AGLIA) grant from the National Geospatial-Intelligence Agency (NGA). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied of the NGA or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

References

- [1] Awad Abdelhalim and Montasir Abbas. Towards Real-time Traffic Movement Count and Trajectory Reconstruction Using Virtual Traffic Lanes. *CVPR Workshop*, pages 2527–2533, 2020.
- [2] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468, 2016.
- [3] Khac-hoai Nam Bui, Hongsuk Yi, and Jiho Cho. A Vehicle Counts by Class Framework using Distinguished Regions Tracking at Multiple Intersections. pages 2466–2474, 2020.
- [4] J. Folenta, J. Špaňhel, V. Bartl, and A. Herout. Determining vehicle turn counts at multiple intersections by separated vehicle classes using cnns. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2544–2549, 2020.
- [5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F Pereira, C J C Burges, L Bottou, and K Q Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.
- [6] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, volume 8693 of *Lecture Notes in Computer Science*, pages 740–755. Springer, 2014.
- [7] Zhongji Liu. Robust Movement-Specific Vehicle Counting at Crowded Intersections.
- [8] Milind Naphade, Shuo Wang, David Anastasiu, Zheng Tang, Ming-Ching Chang, Xiaodong Yang, Liang Zheng, Anuj Sharma, Rama Chellappa, and Pranamesh Chakraborty. The 4th AI City Challenge. 2020.
- [9] Andres Ospina, Rue Josephine, Baker Stains, and Felipe Torres. Countor : count without bells and whistles.
- [10] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. 2018.

- [11] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems* 28, pages 91–99. Curran Associates, Inc., 2015.
- [12] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.
- [13] Zhongdao Wang, Liang Zheng, Yixuan Liu, Yali Li, and Shengjin Wang. Towards real-time multi-object tracking, 2020.
- [14] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric, 2017.
- [15] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. *Proceedings - International Conference on Image Processing, ICIP*, 2017-Sept:3645–3649, 2018.
- [16] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. <https://github.com/facebookresearch/detectron2>, 2019.