Robust and Fast Vehicle Turn-counts at Intersections via an Integrated Solution from Detection, Tracking and Trajectory Modeling

Zhihui Wang[1], Bing Bai[1], Yujun Xie[1], Tengfei Xing[1],
Bineng Zhong[2]*, Qinqin Zhou[2], Yiping Meng[1], Bin Xu[1], Zhichao Song[1],
Pengfei Xu[1], Runbo Hu[1], Hua Chai[1]
1.Didi Chuxing
2.Huaqiao University
jillianwangzhihui, xupengfeipf@didiglobal.com, bnzhong@hqu.edu.cn

Abstract

In this paper, we address the problem of vehicle turn-counts by class at multiple intersections, which is greatly challenged by inaccurate detection and tracking results caused by heavy weather, occlusion, illumination variations, background clutter, etc. Therefore, the complexity of the problem calls for an integrated solution that robustly extracts as much visual information as possible and efficiently combines it through sequential feedback cycles. We propose such an algorithm, which effectively combines detection, background modeling, tracking, trajectory modeling and matching in a sequential manner. Firstly, to improve detection performances, we design a GMM like background modeling method to detect moving objects. Then, the proposed GMM like background modeling method is combined with an effective yet efficient deep learning based detector to achieve high-quality vehicle detection. Based on the detection results, a simple yet effective multi-object tracking method is proposed to generate each vehicle's movement trajectory. Conditioned on each vehicle's trajectory, we then propose a trajectory modeling and matching schema which leverages the direction and speed of a local vehicle's trajectory to improve the robustness and accuracy of vehicle turn-counts. Our method is validated on the AICity Track1 dataset A, and has achieved 91.40% *in effectiveness,* 95.4% *in efficiency, and* 92.60% *S1-score, respectively. The experimental results show that our method is not only effective and efficient, but also can achieve robust counting performance in real-world scenes.*

*Corresponding Author.

1. Introduction

In recent years, there has been an interest in detailed monitoring of road traffic, particularly in intersections, to obtain a statistical model of the flow of vehicles through them. These traffic monitoring systems regularly use computer vision since the videos are high in information content which enable smarter than conventional spot sensor. For example, with vision techniques it is possible to provide flow, speed, vehicle classification, and detection of abnormalities at the same time.

To the best of our knowledge, even with increased processing power and improved vision techniques, there are very few works that explicitly address turn-counts at intersections. Turn-counts plays an important role in intersection analyses, including traffic operations analyses, intersection design, and transportation planning applications. Besides, turn-counts are needed for developing optimized traffic signal timings leading to various benefits such as fuel consumption reduction, air pollution reduction, travel time improvement and anticipated vehicle crash reduction.

In this paper, we focus on the challenging task of vehicle turn-counts by class at multiple intersections using a single floating camera. As shown in Figure**??**, the scenario is greatly challenging due to various factors: frequent occlusions between vehicles, heavy weather, background clutter, illumination changes, and varying and large numbers of moving objects. To address the issues, we integrate detection, background modeling, multi-object tracking, trajectory modeling and matching in a sequential manner. Our integrated method designs careful interplay between several different vision components to effectively combine them. Motion-based tracking is paired with an early prediction of vehicle trajectories methods for accurate turn-counts of an intersection using a single floating camera. This setup is interesting due to the possibility of tracking vehicles and thus knowing how many vehicles go from a certain en-

trance to a certain exit. However, the camera perspective produces the problems of occlusions, which makes the vehicle detection and tracking process potentially inefficient. To solve this problem, the turn-count accuracy is improved by specifically handling broken trajectories through the cooperation of trajectory modeling module and trajectory matching module. When tracking fails due to noise or occlusion, incomplete trajectories also can be match to the most likely turn movements with the help of the trajectory matching module. The main contributions of this work are threefold.

- To the best our knowledge, this work is among the pioneering work of vehicle turn-counts by class at multiple intersections.
- We design an integrated solution from detection, background modeling, multi-object tracking, trajectory modeling and matching in a sequential manner to effectively and efficiently the vehicle turn-counts task.
- Our method not only achieves robust counting results, but also runs at 13 FPS on the AICity Track1 dataset A.

The paper is organized as follows: Section 2 introduces several related works about object detection, background modeling, tracking and trajectory matching, Section 3 describes our whole pipeline, Section 4 shows our experimental results and we conclude our paper in Section 5.

(a) (b) (c)

Figure 1. Examples of vehicle counting scene. (a) Rainy (b) Complex traffic scene (c) Objects occlusion.

2. Related Work

2.1. Object Detection

In recent years, object detection has made tremendous progress and are widely used in various fields, such as security, virtual reality, smart transportation, autonomous driving, etc. Object detection has two evaluation standards, accuracy and efficiency. Two-stage object detection methods represented by Faster R-CNN[17] have excellent accuracy performance. FPN[9] exploits inherent multi-scale, pyramidal hierarchy of deep convolution networks to construct feature pyramids, provides a rich feature representation. Mask RCNN[7] adds an additional mask prediction branch to improve detection accuracy through multi-task learning. But these methods have a region proposal network that needs tremendous computation cost and is less efficient. In order to reduce the computational overhead, YOLO[16] equates object detection as a regression problem, which directly predicts the bounding boxes and associated class probabilities without proposal generation. SSD[12] uses feature maps of different layers to predict bounding boxes of various sizes; the above one-stage networks greatly improved the detection network efficiency but sacrificed accuracy.

In recent years, a lot of works have been devoted to finding a better trade-off between effectiveness and efficiency among detection methods. RetinaNet[10] uses FPN and Focal Loss to alleviate the problem of unbalanced positive and negative samples in the one-stage detection algorithm. At the same time, with the rise of Neural Architecture Search(NAS), network automatic search gradually replaces the original handcraft-designed network structure. NAS-FPN[6] uses RetinaNet as the baseline and adopts Neural Architecture Search to discover a new feature pyramid architecture. EfficientDet[20] uses the powerful EfficientNet[19] as the backbone network, and uses efficient BiFPN to improve the accuracy and speed of the network.

2.2. Object Tracking

With the development of object detection methods, tracking-by-detection is becoming the most popular strategy for multi-object tracking. Base on detections on each frame, the trajectories of targets can be obtained by data association between adjacent frames, which is a crucial part of multi-object tracking methods[3, 4, 5, 15]. Specifically, the deterministic algorithm is often used to solve the data association problem in online multi-object tracking. In [1], Wojke et al. adopt a deterministic Hungarian algorithm [8] in the proposed tracker with an association metric that measures bounding box overlap, which achieves overall good performance in terms of tracking precision and accuracy. [21] replaces the association metric with a more robust metric that combines motion and appearance information to alleviate identity switches in [1]. Base on the tracking framework of [21], we can get more accurate tracking results and reduce the identity switches by refining and customizing the parameters of Kalman filter under the traffic scene.

2.3. Background Modeling

Background modeling is a commonly used method of moving target detection. Combined with the multi-frame image information, the static background is extracted. Moving targets of the current frame are extracted through background difference. Commonly used background modeling methods include average method, maximum value statistical, single Gaussian modeling, weighted average, mixture of Gaussia, etc. Zhong et al. [23] propose the standard variance of a set of pixels' feature values, which captures mainly co-occurrence statistics of neighboring pixels

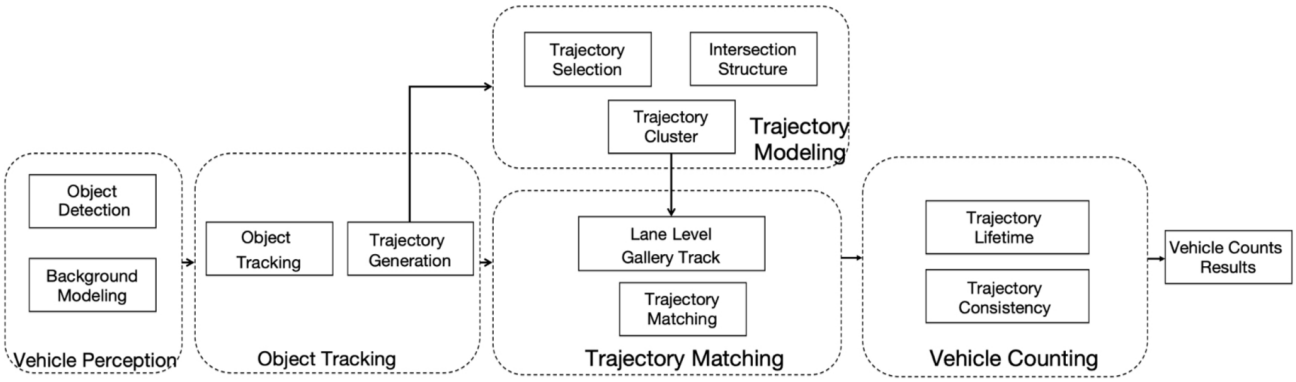


Figure 2. Framework of our system.

in an image patch to model the background and use mixture of Gaussians models to model background. Zhong et al. [22] propose a multi-resolution framework to model background. The Gaussian mixture model is implemented in each level of the pyramid. Feature maps from different levels are combined via AND operator to finally get a more robust and accurate background subtraction map.

2.4. Trajectory-Matching

Trajectory-Matching, also known as Map-Matching, is the process of associating a series of ordered user or vehicle positions to the road network of the map. Its main purpose is to track vehicles, analyze traffic flow and find the driving direction of each vehicle, which is a basic work in the field of maps. The more famous method is based on the Hidden Markov Model[14], and the accuracy rate is as high as 90% under certain conditions; the nearest neighbor algorithm directly calculates the projection distance between the corresponding GPS point and the candidate road, and matches the GPS point to the nearest candidate road, because of a high density of urban roads and drifts of GPS points, this method is not effective usually;

2.5. Automatic Vehicle counting

Automatic vehicle counting is a fundamental technique for intelligent transportation. But there are few works that explicitly deal with vision-based automatic vehicle counting [13]. Most algorithms combine background subtraction and simple feature-based tracking to generate object position during video [18]. Then entrance and exist zones based methods are used for vehicle counting. [18] adopts a typical path to match non-regular sequence, but it is helpless in dealing with short trajectories in complex traffic scenes.

3. Our Method

To achieve robust and fast vehicle turn-counts, we design an integrated solution that contains five modules in a sequential manner. An overview of our proposed framework is illustrated in Figure 2. In the vehicle perception module, object detection and background modeling are combined to detect and classify vehicles. Then the detection-based object tracking model generates trajectories of different objects through the whole video. After matching these trajectories with modeling lane level gallery tracks, all eligible trajectories are counted into corresponding movements with the consideration of its' lifetime and the spatial-temporal consistency. In this section, we will introduce every module in detail.

3.1. Object Detection

We evaluate four efficient detection algorithms, SSD[12], YOLOv3[16], EfficientDet[20] and NAS-FPN[6], from two aspects, effectiveness and efficiency. When comparing the efficiency of four algorithms, meanwhile, we put post processing time into account. The final comparison is published as Tabe 1. Considering both effectiveness and efficiency, we use NAS-FPN as the deep detection model for vehicle detection, which is based on RetinaNet[10]. RetinaNet is composed of backbone and FPN modules. FPN uses a top-down cross-layer connection to merge high-level semantic features and low-level detail features, improving the detection effect of small-scale targets. NAS-FPN has been further optimized on the basis of FPN, drawing on the classification network architecture search method NAS-Net. NAS-FPN uses as inputs features in 5 scales $C3, C4, C5, C6, C7$ with corresponding feature stride of $8, 16, 32, 64, 128$ pixels. It also proposes merging cells to merge any two input feature maps from different layers into an output feature map with a desired scale. And similar to NAS-Net, an RNN controller decides to use which two candidate feature maps and a binary operation to generate a new feature map. More details are available in the official paper. In the framework, we choose NAS-FPN based on mmdetection[2], which uses Res-Net50 as the

Table 1. Comparison of effectiveness and efficiency on different detection algorithms on COCO2017 test set[11]. Inference time means the total process of forward propagation and post-processing.

Algorithm	mAP	Inference
SSD-300[12]	29.3	0.08s
YOLOv3-960[16]	33.0	0.40s
EfficientD0[20]	32.4	0.20s
NAS-FPN-640[6]	37.0	0.09s

backbone model and 640 as the input resolution.

3.2. GMM like Background Modeling

A deep learning-based detection model can perceive most vehicle objects in normal scenes. But it leads to high miss recall in extreme scenarios, such as rainy or illumination changes, in which the appearance information of objects are limited. Therefore, we introduce a background modeling algorithm based on Hybrid Gaussian to extract moving vehicle targets. In order to further reduce the influence of dynamic backgrounds, such as raindrops and lightness, we learn from [23], which introduces particle background modeling. Assuming that the size of input image is $M \times N$. An average pooling operation with kernel size k=10 is performed and generates a small pooled image. Then mixed Gaussian modeling is performed to model the background of multi-frames (nearly 5 seconds images) to generated a robust background model. As for input image of frame t, $t^{'}$ is the pooled images after average pooling. We can get feature map $t_b^{'}$ after background difference. Then, $t_b^{'}$ will be scale to the original image size. And the final feature map is used to perform contour detection for moving object detection after a serial of morphological operations, such as erosion and dilation. An example of background difference is shown in Figure 3.

(a) (b)

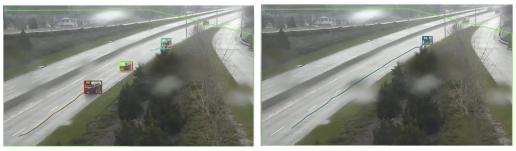
Figure 3. Examples of background subtraction. (a) is related to $t_b^{'}$, (b) is the final moving object detection results based on background modeling.

3.3. Object Tracking

Following the tracking-by-detection paradigm, we use DeepSort [21] algorithm to perform data association on targets. The algorithm is mainly composed of three parts: motion prediction, data association and trajectory management. Furthermore, post-processing is adopted to improve the quality of trajectories.

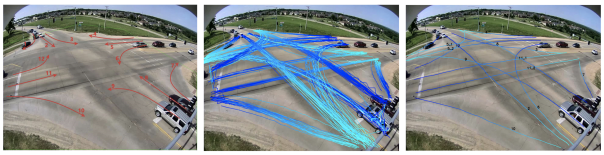
Motion Prediction The Kalman filter is used for motion prediction and state updates. When initializing a new target, it uses the unmatched detections to initialize the target state. And the matched detections are applied to update the target state.

Data association For the same targets between adjacent frames, the distances are calculated to measure the similarity. So we employ a greedy match algorithm to associate predicted targets and detections on current frame based on motion and position information. First, we use the Mahalanobis distance to calculate the motion similarity between detections and predicted positions of Kalman filter. Then, the intersection-over-union(IoU) distance is used to assign the rest of detections to unmatched targets, which can reduce the identity switches of targets that have similar motions in the first step. In addition, we follow the idea of cascading matching in DeepSort to assign matching priority to more frequently occurring targets. Finally, the Hungarian algorithm is used to solve the optimal solution of the cost matrix.



(a) (b)

Figure 4. Comparison of tracking results before and after tracking post-processing. (a) is the results before post-processing. (b) is the integral trajectory after post-processing.



(a) (b) (c)

Figure 5. Schematic diagram of trajectory modeling. (a) is the movement supplied by organizer. (b) is the satisfying trajectory after filtering. The driving directions are represented by the color of trajectories, where the deep blue represents the start of the trajectories. (c) is the modeling trajectory, which is aggregated from selected trajectories in (b).

Track management When a moving target enters or leaves the RoI area in the video sequence, the target needs to be initialized and terminated accordingly. The detection is

Algorithm 1 Trajectory Optimization and Trajectory Association

Input: Tracks of each target T_{target} and Tracks of targets in each frame T_{frame}

Output: Trajectories after post-processing T_{new}

for each $track \in T_{target}$ **do**
  if $track\_length < 2$ or $track$ is static **then**
    Delete
  end if
  Smooth $track$;
end for

for each $new_{track} \in T_{frame}$ **do**
  for each $old_{track} \in T_{prev_{frame}}$ **do**
    Skip $old_{track} \in T_{next_{frame}}$;
    Constraint motion angle between old_{track} and new_{track}
    Constraint time between old_{track} and new_{track}
    Compute distance between old_{track} and new_{track}
    Choose old_{track} of minimum distance to association
    Add new_{track} to T_{new}
  end for
end for

initialized as a new target if its IoU with all existing targets on the current frame is less than IoU_{min}. In order to avoid false targets caused by false-positive detections, the new target is only regarded as initialized successfully after accumulating matched for n_{init} frames. If the target is not matched with detections for accumulated max_{age} frames, the trajectory of target is terminated to prevent the prediction error after long-term tracking and the growth in the number of disappeared targets.

Post-processing Due to the false detections and the instability of the tracker, the trajectory of target can be discrete, and the identity switches problem between targets can be more serious. Thus, we performed post-processing on the tracking trajectories to optimize the trajectory information and obtain the clean and consistent target trajectories for better counting results. Post-processing of the trajectory mainly includes two parts: trajectory optimization and trajectory association. The process is defined as Algorithm 1 and the comparison of the results is shown in Figure 4.

3.4. Trajectory Modeling

After target detection and tracking, the driving trajectory of each target can be generated, and the intersection connections in all directions would be combined. With a large number of driving trajectories for aggregation, we can model the movement of vehicles at lane-level. We develop a trajectory matching algorithm. By calculating the similarity between the query and modeled trajectories in dimensions of position and direction, the driving direction of each trajectory can be verified precisely, and this will provide stable characteristics for vehicle counting.

Figure 5 (a) illustrates the trajectories of cam_6, the red line indicates the driving directions of the intersection, and green polygon is the RoI area. A large number of vehicle trajectories are aggregated at each intersection to model the vehicle trajectories at lane-level. Trajectory modeling can be divided into three steps: trajectory selection, aggregation and template fitting.

3.4.1 Trajectory Selection

Affected by illumination changes or occlusions, identity switches and trajectory breaks may occur during the process, leading to some low-confidence or short tracking trajectories. To generate a high-quality trajectory model, we select the trajectories by considering integrity, continuity, and confidence.

Integrity Integrity is defined based on the entrance and exit area of a specific driving movement. If the starting point and endpoint falls in corresponding areas, and the entire trajectory runs through the RoI area, then it's an integral trajectory. Integrity judgment can effectively filter out distracted trajectories.

Continuity If a trajectory of each frame has a corresponding detection result, we defined the trajectory as a continuous one. Normally, occluded targets with no detection results have a high risk of identity switches. By checking continuity, unreliable trajectories can be effectively removed.

Confidence In target tracking, every detection in the current frame will match the corresponding tracking trajectory. By defining the threshold of matching score, mismatched trajectories will be eliminated, which ensures the reliability of the trajectories.

A large number of distracted or low-quality trajectories can be filtered out through the three dimensions mentioned above.

Eventually, to make sure there are enough and balanced trajectories for modeling, the number of trajectories connecting two intersections in each scene is guaranteed to be $[n, m]$. The selection results is shown in Figure 5 (b).

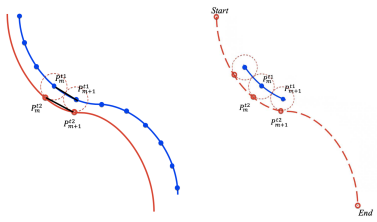
3.4.2 Trajectory aggregation

After trajectory selection, we get a sufficient amount of high-quality trajectories. Using aggregation algorithm, tra-

jectories in the same driving direction can be clustered together. When there are multiple lanes in one driving direction, we can still obtain lane-level trajectory information. Let the complete trajectory set $Traj_M = [Traj_{m_1}, Traj_{m_2}, \dots]$, and $m_i = [p_m^{t_1}, p_m^{t_2}, \dots, p_m^{t_n}]$, where $p_m^{t_i} = (x_m^{t_i}, y_m^{t_i})$. That is, m_i enters the RoI at the t_1 frame, leaves the RoI at the t_n frame, and the target position at time t_i is $(x_m^{t_i}, y_m^{t_i})$. By calculating the Euclidean distance between any two tracks, the similarity between the tracks can be obtained. This paper uses K-means to cluster the trajectories, K is the number of lane-level movements. The aggregated trajectory is shown in Figure 5 (c).

3.4.3 Trajectory template fitting

After trajectory aggregation, the lane-level trajectory clustering results can be obtained. Based on the evaluation of the three dimensions mentioned above, top N trajectories can be selected from each lane-level driving trajectory to perform template fitting. The final model of the lane-level trajectory can be expressed as a discrete sequence extracted from the curve equation by trajectory fitting. Figure 5 is the visualization result of trajectory modeling.



(a) (b)

Figure 6. Trajectory segmentation and matching. The blue curve is the query trajectory, and the red one is the modeling trajectory. (a) is trajectory segmentation, in which the modeling trajectory divided adaptively according to the query. (b) is trajectory matching scheme. The lifetime of query trajectory can be predicted according to it's matching position with modeling trajectory.

Inspired by map matching, which associates the orderly GPS position to electronic maps by converting the GPS to road network coordinate, we use the center point coordinates of the target vehicle in the image coordinate system as the GPS position of the current vehicle, and associate it with the image coordinate system. Therefore, for each trajectory in the tracking result, the nearest neighbor matching algorithm can find the best movement of the current target vehicle and remove the influence of the non-target trajectory no matter whether the trajectory is complete.

3.4.4 Trajectory segmentation

Since the distance between the starting point and the endpoint of the target is far, only using the full amount of trajectory information for matching will result in significant errors. We divide the trajectory into several segments and calculate the similarity in segments-wise. Finally, the entire trajectory similarity is the sum of the segment-wise measure, normalized by segment number.

For each trajectory, our experiment two different segmentation methods: time and space. In time dimension, we segment the trajectory according to the number of frames in which the target appears. And in space, the trajectory is divided according to the distance of target's position in image coordinate system.

Since the influence of traffic lights and other reasons caused the vehicle to stop for a long time, the use of time-division will introduce more useless trajectory segments, which will have a negative impact on the final result. However, it can perform a certain smoothing process on the trajectory position to eliminate the negative effects caused by long-term parking and trajectory jitters. Therefore, in the end, we chose to use spatial segmentation. The example of trajectory segmentation is shown as Figure 6 (a).

About the definition of trajectory in this article, the set of gallery trajectories for a certain scene is $Traj_G = [Traj_{g_1}, Traj_{g_2}, \dots]$, where $Traj_{g_i}$ represents the i_{th} modeling trajectory in the scene, expressed as a set of discrete points $Traj_{g_i} = [p_g^1, p_g^2, \dots]$, where $p_g^i = (x_g^i, y_g^i)$. For the query trajectory $Traj_{q_i}$, is divided into k small segments, that is, $Traj_{g_i} = [p_q^1, p_q^2, \dots]$. In the subsequent trajectory matching, the weighted average is calculated in units of segments to calculate the overall matching degree.

3.4.5 Trajectory similarity measurement

In order to determine the exact driving direction of each query trajectory, it is necessary to match the query trajectory with lane-level gallery trajectories. For each endpoint P_q^i of the trajectory segment in query trajectory, we can find the best-matched point P_g^i in the gallery trajectory base on the minimum Euclidean distance metric. The gallery trajectory then is divided into k segments adaptively base on these matched points, as shown in Figure 6 (a). The trajectory similarity of each segment pair will be computed further. Trajectory similarity includes two parts: distance similarity and angle similarity. The distance similarity of one segment pair is the sum of the distance between the corresponding endpoints and then normalized by the length of the query trajectory segment. The angle similarity of one segment pair is the intersection angle of two vectors defined by the segment endpoints. The final distance similarity is the average distance score of all trajectory segments between two trajectories. So does angle similarity. The detailed equation is listed as follows.

Figure 7. Case analysis of statistical learning. The blue one is the bad case with average distance and angle. And the red one is the results after parameter optimizer.

$$D_{segmentation} = \frac{D(P_g^i, P_q^i) + D(P_g^{i+1}, P_q^{i+1})}{D(P_q^i, P_q^{i+1})} \quad (1)$$

$$D_{trajectory} = \frac{\sum_{n=1}^{N} D_{segmentation}}{n} \quad (2)$$

3.4.6 Statistical learning

After obtaining the similarity measure between the query trajectory $Traj_{q_i}$ and all the gallery trajectories in $Traj_G$, we decide on the matched trajectory by setting the distance and angle threshold. In this paper, we learn to set parameters through statistical learning. We count the distance and angle distribution between trajectories of the same movement and take the average of trajectory distance and angle as the threshold.

However, after analysis, there are some trajectory outlines in the scene, such as some vehicles turning left in the straight lane, resulting in the average distance and angle can not adapt to the trajectory, like the blue curve in Figure 7. Therefore, taking the analytical results as the initial value, we further optimize the parameters under different distance measures and search for suitable distance and angle thresholds.

3.5. Count numbers

Base on the tracking and trajectory matching algorithm, each trajectory has been matched to an exact gallery trajectory. Due to imperfect detection results or tracking results, a whole trajectory may be divided into several short tracks. In order to get an accurate counting result of all vehicles, we design different counting methods for complete trajectories and incomplete ones. A complete trajectory is defined by the integrity criterion, as mentioned before. Base on the entrance and exit zone of each road intersection and trajectory matching results, we can easily confirm which galleries these trajectories belong to.

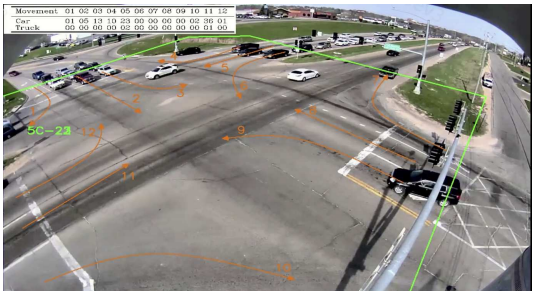


Figure 8. Vehicle counting examples. Orange curves represent for specific movements. Green polygon is an interesting region to do vehicle counting. The table on the top left corner is the counting results of two classes of vehicles counting results in different movements.

For incomplete trajectories, we have to know which unique vehicles they belong to, which is important for accurate vehicle counting. As shown in Figure 6 (b), for each short trajectory, we can confirm how long the journey remains left or already past base on it's matched gallery trajectory, we also can calculate the approximate speed base on the short trajectory length and its time interval. Then the approximate appear timestamp and disappear timestamp of the vehicle are confirmed. We name the appear and disappear timestamp of one vehicle as its survival time. If two short trajectories have the same survival time and have no conflict in location and driving velocity, they will be treated as two trajectory fragments of one vehicle.

The above operation will be repeated until all short trajectories matched to the corresponding vehicle.

4. Experiment

Our experiments are based on the public dataset of Track1 in AI City 2020, which has 20 different scenes and 31 videos. The detailed information about the dataset and evaluation is mentioned on the official website: https://www.aicitychallenge.org/.

All our experiments are tested on Tesla P40 with 12G and two-thread Xeon CPU with 2.20GHz. As mentioned in the official effectiveness evaluation, our base factor is 0.39. Meanwhile, in our framework, only the detection module relies on GPU, which is based on mmdetection [2]. The inference time of the whole procedure, which includes video parsing, detection, tracking and vehicle counting, on more than 180,000 images is 12361.744 seconds. And we get 95.40% points on final efficiency evaluation.

To be clear, we cite the example of vehicle counting published on the official website as Figure 8.

Modules	V1	V2	V3	V4	V5
Traj-PostProc		✓	✓	✓	✓
I-Det			✓	✓	✓
Traj-M			✓	✓	✓
Param-Opt				✓	✓
Bg-Modeling					✓
Effectiveness	82.03	88.87	90.39	91.30	91.40
Efficiency	80.32	80.32	94.00	94.62	95.40
Score	81.52	86.30	91.48	92.30	92.60

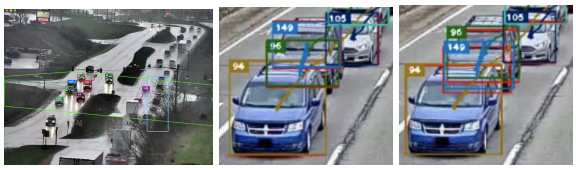
Table 2. Ablation experiments. Traj-PostProc is trajectory post process. I-Det is interval frame detection. Traj-M is trajectory modeling. Param-Opt is parameter optimization. Bg-Modeling is background modeling.

4.1. Ablation experiments

Based on the NAS-FPN and DeepSort trackers, we obtained the V1 version of the baseline model and carried out four improvement experiments to steadily improve the results. In the baseline model, the tracking trajectory breaking caused a large number of counting errors. In V2, we propose trajectory post-processing to associate short-term broken trajectory, which improves the quality of trajectory a lot and increase the effectiveness score by 6.87%; In V3, we add trajectory modeling to increase the trajectory matching efficiency by 1.52%. At the same time, detection is performed every two frames to reduce time-consuming of detection and efficiency score increased by 13.68%; In order to further improve the effectiveness, we optimize the parameters of trajectory modeling and get 0.91% increasing; In V5, we introduce background modeling, and relief the framework from low recall rate caused by bad scenes, such as rainy days (such as cam_2_rain). At this point, our algorithm's total performance in accuracy and efficiency has increased from 81.52% to 92.60%, an increase of 10.08%. Details are shown in Table 2.

4.2. Results Analysis

After several model refinements, our algorithm has achieved significant improvements in effectiveness and efficiency, but there are still many shortcomings and deficiencies. First, due to the limitation of the competition rules, only public detection models can be used, so the detection performance cannot be refined on our specific scene, for example, Figure 9(a). Second, background occlusion or inter-object occlusion leads to obviously id fractures and identity switches in multi-object tracking, like Figure 9(b) and (c). Third, the algorithm doesn't take advantage of apparent features of trajectory for short-track matching, which causes part of the loss ineffectiveness.



(a) (b) (c)

Figure 9. Bad case analysis. (a) is an example in cam_7_rain, which has many miss recall cased by occlusion and illumination. When vehicles park for traffic lights, id-switch is common in (b) and (c).

5. Conclusion

This paper has proposed an integrated system for vehicle turn-counts by class at multiple intersections from a single floating camera. The different components (i.e., NAS-FPN based vehicle detection, GMM like background modeling, Deep-sort multi-target tracking framework, and trajectory modeling and matching schema) were integrated in a sequential yet complementary manner. Vehicle detection recall was greatly improved by effectively combing the NAS-FPN based object detection and GMM like background modeling. Deep-sort multi-target tracking framework was used to obtain target movement trajectories. Moreover, we used the vehicles' direction and speed information for trajectory modeling, trajectory matching, etc. Furthermore, interval detection and multi-threading was applied to improve overall efficiency. Our algorithm has realized robust and fast vehicle turn-counts by class at multiple intersections.

In future work, we will try to improve each component further, both with respect to speed and robustness. For example, we will address typical detection failures, e.g., false positives due to dynamic backgrounds or reflections and missing detections at too large or small vehicles. On the other hand, instead of using the sequential combination manner, how to further fuse each component with other components in an effective feedback cyclic is an interesting direction.

6. Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 61972167 and No. 61802135), and the Open Project Program of the National Laboratory of Pattern Recognition (NLPR) (No. 202000012).

References

[1] 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. IEEE, 2016.

[2] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[3] 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.

[4] 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.

[5] 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.

[6] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7036–7045, 2019.

[7] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[8] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[9] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[10] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

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

[12] 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.

[13] Stefano Messelodi, Carla Maria Modena, and Michele Zanin. A computer vision system for the detection and classification of vehicles at urban road intersections. *Pattern analysis and applications*, 8(1-2):17–31, 2005.

[14] Paul Newson and John Krumm. Hidden markov map matching through noise and sparseness. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 336–343, 2009.

[15] Aljoša 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.

[16] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[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] 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.

[19] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[20] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. *arXiv preprint arXiv:1911.09070*, 2019.

[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] Bineng Zhong, Shaohui Liu, Hongxun Yao, and Baochang Zhang. Multl-resolution background subtraction for dynamic scenes. In *2009 16th IEEE International Conference on Image Processing (ICIP)*, pages 3193–3196. IEEE, 2009.

[23] Bineng Zhong, Hongxun Yao, and Shaohui Liu. Robust background modeling via standard variance feature. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1182–1185. IEEE, 2010.