Traffic Flow Analysis with Multiple Adaptive Vehicle Detectors and Velocity Estimation with Landmark-based Scanlines

Minh-Triet Tran *[1], Tung Dinh-Duy[1], Thanh-Dat Truong[1], Vinh Ton-That[1], Thanh-Nhon Do[1], Quoc-An Luong[1], Thanh-An Nguyen[1], Vinh-Tiep Nguyen[2], and Minh N. Do[3]

[1]University of Science, VNU-HCM, Vietnam
[2]University of Information Technology, VNU-HCM, Vietnam
[3]University of Illinois at Urbana-Champaign, U.S.

Abstract

In this paper, we propose our method for vehicle detection with multiple adaptive vehicle detectors and velocity estimation with landmark-based scanlines. Inspired by the idea for tiny object detection, we use Faster R-CNN with Resnet-101 to create different specialized vehicle detectors corresponding to different levels of details and poses. We propose a heuristic to check the fitness of a particular vehicle detector to a specific region in camera's view by the mean velocity direction and the mean object size. By this way, we can determine an adaptive set of appropriate vehicle detectors for each region in camera's view. Thus our system is expected to detect vehicles with high accuracy, both in precision and recall, even with tiny objects.

We exploit the U.S. road rules for the length and distance of broken white lines on roads to propose our method for vehicle's velocity estimation using such landmarks. We determine equally-distributed scanlines, virtual parallel lines that are nearly-perpendicular to the road direction, with reference to the line connecting the corresponding ends of multiple broken white lines. From the timespan for a vehicle to cross two consecutive virtual scanlines, we can calculate the average vehicle's velocity within that road segment. We also refine the speed estimation by detecting when a vehicle stops at a traffic light, and smooth the results with a moving average filter. Experiments on the dataset of Traffic Flow Analysis from NVIDIA AI City Challenge 2018 show that our method achieves the perfect detect rate of 100%*, the average velocity difference of* 6.9762 *mph on freeways, and* 8.9144 *mph on both freeways and urban roads.*

*Corresponding author. Email: tmtriet@fit.hcmus.edu.vn

1. Introduction

More and more surveillance cameras have been deployed in most countries worldwide to capture information, mostly in visual format, of traffic systems. This valuable data source provides potential insights for various practical applications and utilities, such as traffic jam prediction, smart transportation services, urban planning, etc. However, analyzing to understand visual data from surveillance cameras is still a challenging problem because of the huge amount of data, the lack of annotated corpus and efficient models for visual analysis and understanding, etc.

Various tasks on traffic video analysis are becoming popular, such as vehicle detection and localization [23, 11] , car fluent recognition [14], velocity estimation [8, 12], vehicle re-identification [16, 1, 21], vehicle type classification [13, 22]. In this paper, we focus on a fundamental task to detect vehicles and estimate their speeds. For vehicle detection, we propose to use multiple adaptive vehicle detectors to capture and localize vehicles in different poses and at different distances from the camera. Specifically, we use Faster R-CNN with Resnet-101 to train three detectors: the first detector is for the front and back views of a vehicle, the second one is for the side view of a vehicle, and the last one is for tiny instances of vehicles.

For velocity estimation, it is essential to map the distance in an image to the distance in reality. When such mapping is available or can be determined, it is possible to develop different methods for velocity estimation with either traditional techniques or deep networks. For example, using annotated data with relative position and velocity generated by range sensors, Kampelmuehler et.al. propose to use spatiotemporal depth and motion features [12] for velocity estimation.

When there is no direct information about real-world distance or velocity, it is possible to exploit the dimensions of a known-type vehicle to measure the actual distance it runs on the road surface. By this way, the real distance between certain pairs of points in the road surface can be inferred.

In this paper, we aim to propose a method for velocity estimation with limited source of information. Our method depends only on the information of the traffic system itself, without extra information from vehicles or annotated data on vehicle's position or velocity. In our method, we exploit the landmarks, i.e. broken white lines on road surfaces, based on the U.S. road rules. We generate equally-distributed scanlines, parallel lines that are nearly-perpendicular to the road direction, with reference to the line connecting the corresponding ends of multiple white dotted lines. We detect the time instance when a vehicle runs across a scanline, calculate the timespan that vehicle spends to move between two consecutive scanlines, and infer the speed of that vehicle. In fact, our method can be used for the calibration task even when there is no vehicle moving on the street and without annotated data on vehicle's position or velocity.

We conduct experiments on the dataset of Track 1 – Traffic Flow Analysis of NVIDIA AI City Challenge 2018. Experimental results show that our method for vehicle detection with multiple adaptive detectors achieves the perfect detect rate of 100%, higher than using only one detector as in traditional methods, such as Yolo, Faster R-CNN, SSD, etc. Furthermore, our idea of using multiple adaptive detectors can also be applied for different problems, such as face detection at different distances, resolutions, and poses.

Experiments show that our method can determine the speed of a vehicle with high accuracy (the difference of less than 7 mph) for locations with complete and regular landmark patterns, such as in free ways. However, our method might be affected in urban areas, especially at cross roads, where the landmark patterns might be incomplete or irregular. In our experiments, the difference at a cross roads can be up to 11.9049 mph. On average, the velocity root mean square error of our method for the dataset of Track 1 in NVIDIA AI City Challenge 2018 is 8.9144 mph.

The content of this paper is as follows. In Section 2, we briefly review existing methods and approaches for object detection and vehicle tracking. We propose our method for traffic flow analysis with multiple adaptive vehicle detectors and landmark-based scanlines in Section 3. Experimental results on the dataset of Track 1 - AI City Challenge 2018 are presented and discussed in Section 4. Conclusions and future works are in Section 5.

2. Related Work

In this section, we first present briefly several tasks on traffic analysis recently in Section 2.1, then we discuss about the common method for object detection in the general cases (Section 2.2) and the case of small instances (Section 2.3).

2.1. Traffic Analytic

With the abundant availability of video data from surveillance cameras, it is an urgent need to develop algorithms to process such data for valuable insight understanding for better traffic management and other smart services.

One of the fundamental tasks for traffic video analysis is object detection and localization for scene layout awareness [23] or vehicle localization [11] in traffic surveillance. Detected vehicles can then be classified into known vehicle types [13, 22] or further processed for car fluent recognition [14].

Vehicle velocity estimation is one of the traditional problems in traffic surveillance [2, 8].This problem can be solved with classical techniques, such as object tracking and optical flow, and can take advantage of new achievements in deep networks [12]. Extra information from LIDAR or other sensors and annotated data may be exploited to train networks for location and velocity estimation.

Vehicle re-identification is also a challenging task for traffic video analysis. X. Liu et. al. propose a large scale benchmark dataset facilitate vehicle re-identification research [16]. The dataset is collected with 20 cameras that record unconstrained traffic scene and over 40,000 bounding boxes of 619 vehicles are labeled. There are at least 2 cameras the capture a single vehicle in different viewpoints, illuminations, and resolutions so that the dataset provide high recurrence rate for vehicle re-identification.

Y. Shen et. al. introduce an incorporating additional spatio-temporal information for solving the challenging re-identification task [21]. A chain MRF model with a deeply learned potential function generates candidate visual-spatio-temporal path when a pair of vehicle images with their spatio-temporal information is given. A Siamese-CNN takes the candidate path as well as the pairwise queries to generate their similarity score.

2.2. Object Detection Networks

The application of Deep Learning in Computer Vision using Convolutional Neural Network (CNN) approach has improved significantly and achieved good results in various tasks, especially in object detection. R-CNN [6] is one of the pioneers and there are many methods have been developed based on R-CNN, such as Fast R-CNN[5] or Faster R-CNN [20], which can be considered as a state-of-the-art architecture in Object Detection. However, the representations in R-CNN and its related methods cost lots of time to run on an image completely. To overcome this 'drawback', YOLO [18, 19] and SSD [15] use local information to predict object, instead of extracting Region-of-Interesting

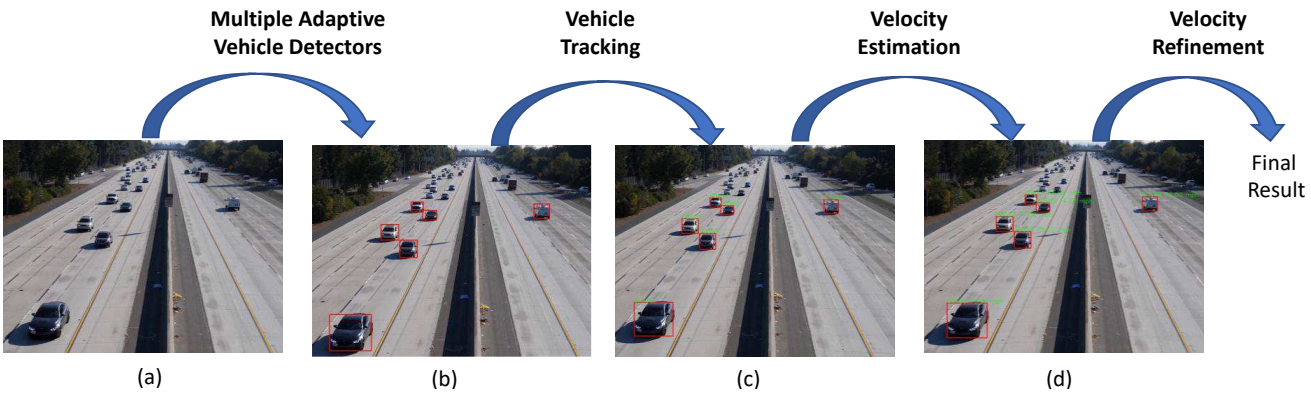


Figure 1. Overview of the proposed method with four main parts. (a) input frame, (b) frame with detected bounding boxes, (c) frame with bounding boxes and vehicle IDs, (d) frame with velocity value for each vehicle.

before moving to classifier like Faster R-CNN to save time and can run in real-time condition.

Because Faster R-CNN is the state-of-the-art method for object detection, we adopt this method in our proposed method for multiple adaptive vehicle detectors (see Section 3.1).

The key features of Faster R-CNN are (1) CNN base which extracts high level features of an image and (2) Region Proposal Network (RPN) that shares full-image above convolutional features with the detection network, thus enabling nearly cost-free region proposals [20]. An RPN is a fully convolutional network that takes an image as an input and the output is a set of rectangle object proposal, each with an object score. The RPN is trained end-to-end to generate high-quality region proposals, which are used by Fast R-CNN for detection.

Most of 'normal' CNN architecture are built with numerous stacked layers which leads to the network might learn unreferenced information and loose some important feature when it becomes deeper. To tackle that problem, ResNet is a potential idea to improve 'original' Faster R-CNN [7]. We choose ResNet-101 variation because its top-1 and top-5 errors are nearly the same with ResNet-152 on ImageNet[3].

2.3. Small Object Detection

Although existing methods may be used directly to detect objects in different scales and levels of details, recent works focus on creating dedicated solutions for small object detection to adapt the detection process for the domains of tiny visual instances.

Christian Eggert et al. propose a method for small object detection in Faster R-CNN [4] by theoretically examine the problem of small objects at the proposal stage. Derive a relationship which describes the minimum object size which can reasonably be proposed and provide a heuristic for choosing appropriate anchor scales. They perform detailed experiments which capture the behavior of both the proposal and the classification stage as a function of object size using features from different feature maps. Deeper layers are potentially able to deliver features of higher quality which means that individual activations are more specific to input stimuli than earlier layers. They show that in the case of small objects, features from earlier layers are able to deliver a performance which is on par with and -can even exceed the performance of features from deeper layers.

Duy Nguyen et al. focus on evaluating real-time model for small object detection [17], specifically, YOLO and SSD. The paper shows that if we focus on detecting small objects in real time and trade off a little about accuracy, YOLO554 is the best choice. On the other hand, if we concentrate on accuracy and do not care about the other aspects we can choose models whose sizes of an input are close to the average size of the real image. If objects are in 10% - 20% of an image, SSD get better results than versions of YOLO, especially SSD512. If objects are less than 10% of an image, the YOLO get better than SSD.

For tiny face detection, Peiyun Hu and Deva Ramanan [9] explore three aspects of the problem in the context of finding small faces: the role of scale invariance, image resolution, and contextual reasoning. Most recognition approaches aim to be scale-invariant, but it is really not suitable for small face. They take a different approach and train separate detectors for different scales. To maintain efficiency, detectors are trained in a multi-task fashion: they make use of features extracted from multiple layers of single (deep) feature hierarchy. In addition, they show that context is crucial, and define templates that make use of massively-large receptive fields (where 99% of the template extends beyond the object of interest). Finally, they explore the role of scale in pre-trained deep networks, providing ways to extrapolate networks tuned for limited scales to rather extreme ranges.

3. Proposed Method

Figure 1 illustrates the overview of our proposed method with four main parts. We first detect all vehicles in a video clip using multiple vehicle detectors to adapt to different regions in a camera's view (Section 3.1). Then we track the trajectory of a vehicle by linking detected bounding boxes in consecutive frames based on the overlapped region of bounding boxes (Section 3.2). We estimate the velocity of a vehicle exploiting the landmarks on a road using multiple scanlines, parallel lines that are nearly perpendicular to the main moving direction of a lane in that road (Section 3.3). Finally, we propose several techniques for velocity refinement (Section 3.4). Both our code and models are available online at https://github.com/HCMUS-Smart-Environment-Group/AICityChallenge2018.

3.1. Vehicle Detection with Multiple Adaptive Object Detectors

3.1.1 Multiple Adaptive Object Detectors

We adopt Faster R-CNN with ResNet-101 for vehicle detection. Although Faster R-CNN is an ideal method for multi-scale object detection, it is still not good enough for detecting small objects[17]. Therefore, we propose to use multiple object detectors to efficiently detect vehicles in different areas of a camera's view. Based on the mean velocity direction and the mean size of a vehicle's bounding box in each region of the frame, we determine the appropriate detectors from our collection of vehicle detectors that should be applied in that region. By this way, we can both increase the recall and reduce false positive vehicles.

Specifically, we create a detector set $\mathbb{D}$ with 3 Faster R-CNN detectors: (1) a detector D_1 for the front and back views of a vehicle, (2) a detector D_2 for the side view of a vehicle, and (3) a detector D_3 for a tiny vehicle which is very far from the camera.

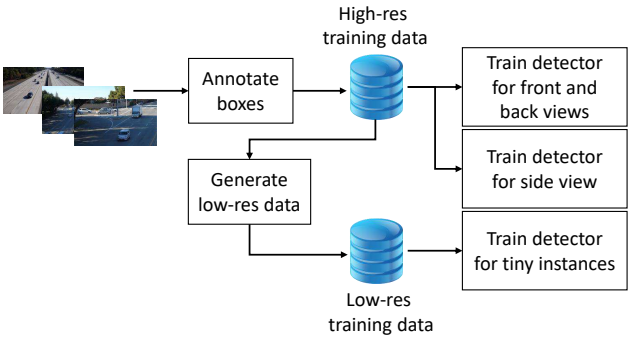


Figure 2. Process to train multiple vehicle detectors.

To train detectors in $\mathbb{D}$, we select only 1000 frames, approximately 2% frames in all video clips, for manual annotation. Instead of detecting all vehicles in selected frames for training, we only annotate, on average, 10 most visible vehicles per frame. We define 2 groups of vehicle instances: front or back views, and side view. In total, we have about 7000 bounding boxes for front and back views, and about 3000 bounding boxes for side views.

We do not create training samples for tiny instances of vehicles directly from training frames because such task is difficult, unnecessary, and the boundary for such instance may not be precisely annotated. Instead, we generate training samples for tiny instances but applying a low pass filter to the high resolution (and high quality) training samples (see Figure 2).

For vehicle detection process, in our preliminary implementation, we forward each frame through all detectors in $\mathbb{D}$, extract all bounding boxes, and apply non-maximum suppression to get the final detection result. As each detector may be appropriate just for certain areas in a camera's view, applying a detector to an inappropriate region may produce false positive vehicles.

3.1.2 Region-based Adaptive Set of Detectors

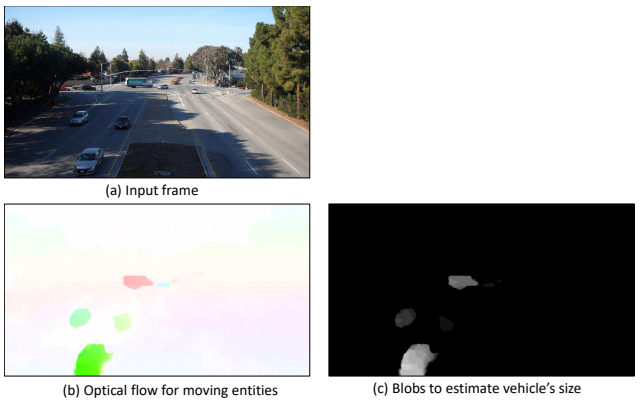


Figure 3. Velocity direction estimation and vehicle's estimation using FlowNet. (a) input frame, (b) optical flow of vehicles - the color represents the flow direction, (c) blobs to evaluate the size of each vehicle.

As the camera is fixed, we propose a heuristic to evaluate the fitness of a detector $D_i \in \mathbb{D}$ with certain area in a frame based on the profile of the detector and that of each region in the camera's view.

The profile for each detector D_i is defined from its training data, including the distribution of velocity direction and the distribution of vehicle's size. Each distribution can be represented by its mean and standard deviation. For each training sample of a vehicle, we use not only its bounding box but also the instant flow map, generated by FlowNet [10], at the corresponding frame. The velocity direction of each training sample is defined as the mean of flow map directions.

To split the camera's view into appropriate regions, we first determine the main moving direction at each pixel as the mean of (non-zero) velocity direction at that pixel over multiple frames. Then we evaluate the average blob size of moving vehicle at each pixel over time/ Finally, we can define the profile at each pixel or even a certain region in the camera's view by the distribution of velocity direction and the distribution of blob size. Figure 3 illustrates our usage of FlowNet to determine the main velocity direction and blobs of moving vehicles for a given frame.

Using our proposed heuristic, we can divide a frame into multiple regions, each corresponds to a subset of $\mathbb{D}$ that should be applied to that region.

3.2. Vehicle Tracking with Overlapped Bounding Boxes in Consecutive Frames

Because the existence of a vehicle is continuous frame-by-frame, we use a simple technique to track vehicles based on the previous frame. For each bounding box B_1 in the current frame, we find a bounding box B_2 at the previous frame with the highest IoU rate (Intersection of Union, (1)).

$$IoU\,(B_1, B_2) = \frac{\|B_1 \cap B_2\|}{\|B_1 \cup B_2\|} \tag{1}$$

If the IoU score is greater than a certain threshold, we link the two bounding boxes B_1 and B_2. By this way, we can create a sequence of bounding boxes of a vehicle in a video clip. We assign the same vehicle ID for all tracked bounding boxes of one vehicle in a video clip. Figure 4 visualizes the trajectories of certain vehicles with our simple tracking technique. Experiments show that this method is sufficient to follow a vehicle in traffic video from a surveillance camera, even when it changes its moving direction, such as turning left, right, or U-turn.

(a) Moving with straight path (b) Moving with curve path

Figure 4. Trajectories of tracked vehicles. (a) vehicles move in straight lines (b) a vehicle moves forward then turns right.

3.3. Velocity Estimation with Scanlines based on Landmarks

Estimating the distance in real world between two pixels in an image is the essential step of speed estimation and this calibration task can be done only once.

In this section, we present our proposed method for approximate calibration based on U.S. road rules for broken white lines painted in roads. According to US Department of Transportation, in each broken white line, each segment has a standard length of 10 feet, and the gap between two consecutive segments is 30 feet (see Figure 5a). Thus the distance between the first point of a 10-feet segment 1 and the first point of segment 2 in the same dash line is expected to be 40 feet.

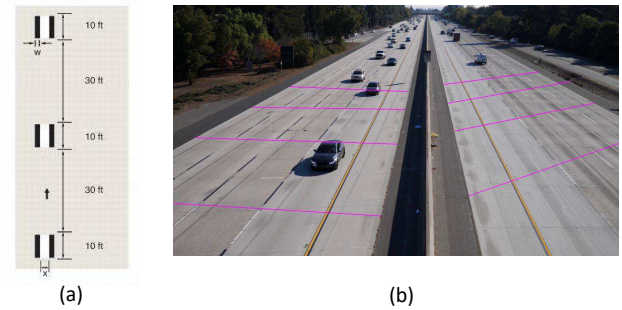


Figure 5. Landmarks and virtual scanlines on road surface. (a) landmarks with broken line marking pattern (b) virtual scanlines that connect corresponding ends of multiple broken line segments.

As dash lines in the same driving direction are parallel, the 4 first points of the 4 white line segments in 2 adjacent dash lines usually form a rectangle, or at least a parallelogram. Therefore, we propose to draw a virtual scanline passing through the first points of 2 white line segments of 2 adjacent parallel dash lines, and the distance d between the 2 consecutive scanlines is 40 feet (see Figure 5b)

There are 3 cases to estimate the velocity of a vehicle:

1. The vehicle crosses through at least 2 scanlines
2. The vehicle crosses only 1 scanline
3. The vehicle doest not cross any scanline.

The first case is the main scenario we use for speed estimation. Let f_i and f_{i+1} be the frame index when a vehicle goes across the i^{th} and $(i+1)^{th}$ scanlines, respectively. Figure 6 illustrates our strategy to determine the time instant when a vehicle first crosses a scanline. If the vehicle runs toward the camera, its front is not occluded and crosses the scanline before other parts in the vehicle. Then we choose the frame when the bounding box of a vehicle first hits the scanline. If the vehicle runs away from the camera, its back is visible and crosses the scanline after all other parts in the vehicle. In this situation, we choose the frame when the bounding box of a vehicle last hits the scanline.

The timespan (in seconds) for the vehicle to run the distance d is 40 feet between two consecutive scanlines is $t = \frac{(f_2 - f_1)}{\text{fps}}$ where fps is the frame rate of the video clip. The average velocity of the vehicle between the two scanlines can be determined with a simple formula $v = d/t$. In the case that the vehicle hits only 1 scanline, we suppose its velocity $v = 0$. For the last case, we cannot determine the velocity when there is no collision between the bounding

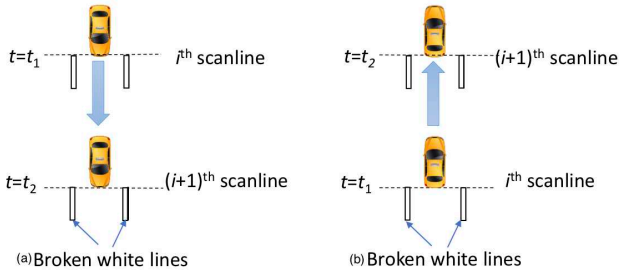


Figure 6. Strategy to determine when a vehicle runs across a virtual scanline. (a) the vehicle runs toward the camera, its front is visible and hits first with the scanline (b) the vehicle runs far away from the camera, its back is visible and hits last with the scanline

box of a vehicle and any scanline, so we set its velocity v to infinity.

3.4. Velocity Refinements

Using our proposed method for velocity estimation with landmark-based scanlines, we can successfully determine the velocity of a vehicle when it actually runs continuously between consecutive virtual scanlines. This assumption is true for vehicles in freeways. However, in urban areas, especially at crossroads, a vehicle should slow down its speed when it approaches an intersection, stop at a red traffic light, and accelerate its speed when the traffic light turns green. Therefore, it is necessary to detect when a vehicle stops to better estimate its speed.

For each vehicle in a frame, we calculate the color similarity in its bounding box between the current and previous frames using the cosine similarity. If the visual data in its bounding box is nearly the same in two consecutive frames, the vehicle is considered to be in the idle state (or with neglected movement). By this way, we can determine the state (Moving or Stopped) of a vehicle in each frame. We also smooth the sequence of states using a smoothing window with the size of 30 frames (equivalent to 1 second).

After smoothing the sequence of moving states of a vehicle, we assign the velocity of that vehicle to be 0 (mph) when it is in Stopped state. We use linear interpolation to estimate the velocity for the vehicle when it slows down or accelerates its speed.

4. Evaluation on NVIDIA AI City Challenge 2018

In this section, we first present the dataset and metrics used to evaluate our proposed method. Then we study the improvement of our method with multiple adaptive vehicle detectors over existing methods, such as YOLO, Faster R-CNN (Section 4.2). Finally, we evaluate the velocity error of our method in two main environmental contexts: freeways without crossroads and traffic lights, and roads in urban areas (see Section 4.3).

4.1. Dataset

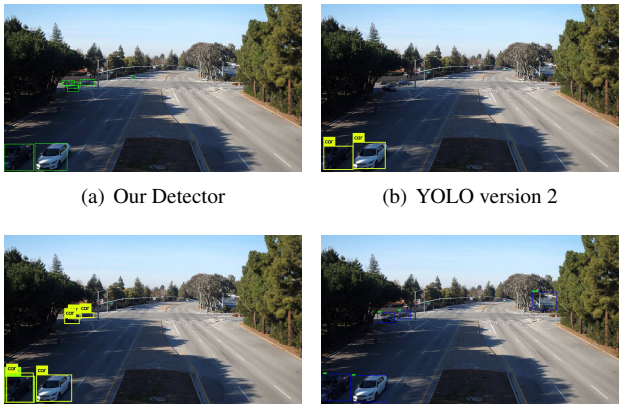
We conduct the experiments with our proposed method on the dataset of Track 1 - Traffic flow Analysis in NVIDIA AI City Challenge 2018. This dataset includes 27 video clips recorded from fixed cameras at 4 different locations in California, U.S.A. The duration for each clip is 1 minute with 1800 frames in total.

For the first two locations (with 16 clips), traffic flow is recorded on I-280 highway. There are only two opposite driving directions along the vertical axis of the video clip. As there is no intersection or traffic lights, each vehicle runs at a nearly-constant speed without any direction change, such as turning left, right, or U-turn. These two locations are ideal case for our proposed method to estimate velocity with landmark-based scanlines (see Section 3.3).

For the last two locations (with 11 clips), the video clips show the traffic data at intersections with traffic lights in urban areas. A vehicle may slow down or accelerate its speed, stop, or change it moving direction. This context raises difficult yet practical issues for velocity estimation, and demonstrates the importance and efficiency of the velocity refinement step in our proposed method.

The detection rate (DR) is used to measure the accuracy of vehicle detection. DR is the ratio of detected vehicles and the total number of vehicles. Vehicle is considered as detected if the intersection-over-union (IOU) score from 0.5 to 1.0 between its predicted bounding box and its ground truth bounding box in at least 30% of its ground truth frames. The speed estimate error is the root mean square error (RMSE) of the ground truth vehicle speed and predicted speed for all correctly localized ground-truth vehicles.

4.2. Detection Rate



(a) Our Detector (b) YOLO version 2

(c) YOLO version 3 (d) Original Faster RCNN

Figure 7. Comparison on vehicle detection between our proposed method with other existing methods, including YOLO v2, YOLO v3 and Original Faster R-CNN

Figure 7 illustrates an example of vehicle detection with our proposed method and some state-or-the-art object detection methods, including YOLO v2 and v3, and original Faster R-CNN. YOLO and the origin Faster R-CNN can detect most of the cars in a frame. However, they still miss small objects due to the long distance as discussed in Section 3.1. The key idea, which is also the main difference of our methods comparing to others, is that we train and use multiple vehicle detectors to adapt to the profile of a specific region in the camera's view. In Figure 7, our detectors can detect very small instances of vehicles, while such objects are usually ignored in other methods.

If we use only one detector as in traditional approach, the detect rate (DR) on the evaluation dataset is 88.89%. In this case, we miss some vehicles that are far away from the camera. When we use all three detectors, our method achieves up to the absolute score 100% detection rate on Track 1 NVIDIA AI City Challenge 2018.

4.3. Evaluation of Velocity

Table 1 shows the detection rate and velocity error of our proposed method on the evaluation dataset. The detection rates in all cases are always 100%.

As we mention in Section 4.1, the first two locations with 15/26 video clips are the ideal cases for our method to estimate velocity. All vehicles continuously runs without any significant velocity or direction changes. For this case, our method achieves the root mean square error (RSME) of 6.9762 mph.

For the second case, we consider the crossroads in urban areas with 11/27 clips. Our method can successfully predict velocity when a vehicle continuously runs on a street and crosses virtual scanlines. However, near the intersection of two roads, the landmark patterns may not be complete, and our method may not evaluate directly the velocity from the last scanline to the traffic light, and in the middle of the crossroad. Therefore, the RMSE for this case is 11.9049 mph, and the error mostly appears near the intersection of two roads during the slow down or accelerate periods of a vehicle.

We consider all the four locations with 26 video clips in the last two cases. Using the refinement techniques (see Section 3.4, we achieve the RMSE of 8.9144 mph (for case 4), while the RMSE without refinement is 9.3800 mph (for case 3).

Case	Locations	#Clips	DR	RSME
1	1, 2	15	1.0	6.9762
2	3, 4	11	1.0	11.9049
3	1, 2, 3, 4	26	1.0	9.3800
4	1, 2, 3, 4	26	1.0	8.9144

Table 1. Evaluation of velocity in different locations.

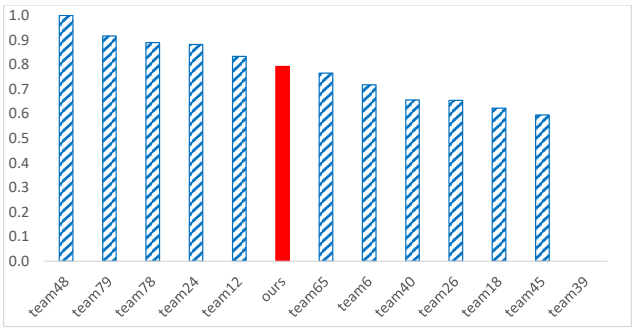


Figure 8. Comparison between different teams in Traffic Flow Analysis task.

Figure 8 shows the comparison between our method with others in the Traffic Flow Analysis task of NVIDIA AI City Challenge 2018. This task uses the score $S1 = DR \times (1 - NRMSE)$ where DR is the detection rate and NRMSE is the normalized root mean square error. Our method achieves the $S1 = 0.7924$ and ranks 6th in the track.

5. Conclusion and Future Works

In this paper, inspired by the idea of tiny object detector, we propose a object detection scheme with multiple adaptive detectors to detect objects in different poses and sizes. For vehicle detection, we train and use 3 specific detectors using Faster R-CNN with Resnet-101 corresponding to 3 view types: (1) the front and back views of a vehicle, (2) the side view of a vehicle and (3) a tiny instance of a vehicle which is very far away from the camera.

We also propose a heuristic to select appropriate set of trained detectors for each region in the camera's view based on the the fitness of their profiles. In the context for vehicle detection, the profile for a particular detector includes the distribution of vehicle's size and the distribution of velocity direction of training data that are used to trained that detector. The profile for each pixel or region in the camera's view includes the distribution of moving blob size and the distribution of object flow's direction from the optical flow map. By this way, we can improve both precision and recall for object detection as we only apply detectors that are appropriate to certain area in the camera's view.

For velocity estimation, we propose to create virtual scanlines from the landmarks on almost every road, i.e. broken white lines on a road's surface. We exploit the standard length of a white line segment and the standard gap distance between two consecutive segments in a single white dash line to calibrate the mapping between pixels in an image to real-world distance. As our proposed method for velocity estimation does not required augmented data with annotation (such as the location of velocity), our method can be applied easily in most cases.

Although our detection rate for vehicles is 100%, our detectors might lose some bounding boxes, thus some vehicles hold more than one ID. Besides, we need further refinements to estimate vehicle's velocity near the intersection of roads in urban areas.

References

[1] X. Z. A. Kanaci and S. Gong. Vehicle re-identification by fine-grained cross-level deep learning. In *5th Activity Monitoring by Multiple Distributed Sensing Workshop, British Machine Vision Conference*, pages 1–6, July 2017.

[2] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik. A real-time computer vision system for vehicle tracking and traffic surveillance. *Transportation Research Part C: Emerging Technologies*, 6(4):271 – 288, 1998.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[4] C. Eggert, S. Brehm, A. Winschel, D. Zecha, and R. Lienhart. A closer look: Small object detection in faster r-cnn. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pages 421–426, July 2017.

[5] R. Girshick. Fast r-cnn. In *International Conference on Computer Vision (ICCV)*, 2015.

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition*, 2014.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[8] J.-W. Hsieh, S.-H. Yu, Y.-S. Chen, and W.-F. Hu. Automatic traffic surveillance system for vehicle tracking and classification. *Trans. Intell. Transport. Sys.*, 7(2):175–187, Sept. 2006.

[9] P. Hu and D. Ramanan. Finding tiny faces. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[10] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[11] H. Jung, M.-K. Choi, J. Jung, J.-H. Lee, S. Kwon, and W. Young Jung. Resnet-based vehicle classification and localization in traffic surveillance systems. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[12] M. Kampelmühler, M. G. Müller, and C. Feichtenhofer. Camera-based vehicle velocity estimation from monocular video. *CoRR*, abs/1802.07094, 2018.

[13] P.-K. Kim and K.-T. Lim. Vehicle type classification using bagging and convolutional neural network on multi view surveillance image. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[14] B. Li, T. Wu, C. Xiong, and S.-C. Zhu. Recognizing car fluents from video. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

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

[16] X. Liu, W. Liu, H. Ma, and H. Fu. Large-scale vehicle re-identification in urban surveillance videos. In *2016 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, July 2016.

[17] P. Pham, D. Nguyen, T. Do, T. D. Ngo, and D.-D. Le. Evaluation of deep models for real-time small object detection. In D. Liu, S. Xie, Y. Li, D. Zhao, and E.-S. M. El-Alfy, editors, *Neural Information Processing*, pages 516–526, Cham, 2017. Springer International Publishing.

[18] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, July 2017.

[19] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.

[20] 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 (NIPS)*, 2015.

[21] Y. Shen, T. Xiao, H. Li, S. Yi, and X. Wang. Learning deep neural networks for vehicle re-id with visual-spatio-temporal path proposals. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1918–1927, Oct 2017.

[22] J. Sochor, J. pahel, and A. Herout. Boxcars: Improving fine-grained recognition of vehicles using 3-d bounding boxes in traffic surveillance. *IEEE Transactions on Intelligent Transportation Systems*, PP(99):1–12, 2018.

[23] T. Wang, X. He, S. Su, and Y. Guan. Efficient scene layout aware object detection for traffic surveillance. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.