Real-time Traffic Pattern Collection and Analysis Model for Intelligent Traffic Intersection

Unnikrishnan Kizhakkemadam Sreekumar, Revathy Devaraj, Qi Li, Kaikai Liu
Computer Engineering Department
San Jose State University (SJSU)
San Jose, CA, USA
Email: {unnikrishnan.kizhakkemadamsreekumar, revathy.devaraj, qi.li, kaikai.liu}@sjsu.edu

***Abstract*—The traffic congestion hits most big cities in the world - threatening long delays and serious reductions in air quality. City and local government officials continue to face challenges in optimizing crowd flow, synchronizing traffic and mitigating threats or dangerous situations. One of the major challenges faced by city planners and traffic engineers is developing a robust traffic controller that eliminates traffic congestion and imbalanced traffic flow at intersections. Ensuring that traffic moves smoothly and minimizing the waiting time in intersections requires automated vehicle detection techniques for controlling the traffic light automatically, which are still challenging problems.**

In this paper, we propose an intelligent traffic pattern collection and analysis model, named TPCAM, based on traffic cameras to help in smooth vehicular movement on junctions and set to reduce the traffic congestion. Our traffic detection and pattern analysis model aims at detecting and calculating the traffic flux of vehicles and pedestrians at intersections in real-time. Our system can utilize one camera to capture all the traffic flows in one intersection instead of multiple cameras, which will reduce the infrastructure requirement and potential for easy deployment. We propose a new deep learning model based on YOLOv2 and adapt the model for the traffic detection scenarios. To reduce the network burdens and eliminate the deployment of network backbone at the intersections, we propose to process the traffic video data at the network edge without transmitting the big data back to the cloud. To improve the processing frame rate at the edge, we further propose *deep object tracking* algorithm leveraging *adaptive multi-modal models* and make it robust to object occlusions and varying lighting conditions. Based on the deep learning based detection and tracking, we can achieve pseudo-30FPS via *adaptive key frame selection*.

***Index Terms*—traffic surveillance, real-time traffic data, edge devices, deep learning, multiple object tracking.**

I. Introduction

With 180,000 people moving into cities every day, rapid urbanization places a significant strain on transportation infrastructure and traffic management for most cities. Santa Clara in Silicon Valley has the second longest commute in the nation, by some estimates. At present, the traffic signals are controlled with fixed timers. The commuters on the red signals have to stop compulsorily, whether there is traffic or no traffic. Real-time, robust and reliable traffic surveillance and analytics are urgent requirements to improve urban traffic control systems. Many intelligent systems have been proposed to take steps such as optimizing timings at traffic signals in order to achieve the best possible flow [1]. For example, the virtual loop cameras have been proposed to constantly read the density of traffic at the intersections and adjust the green/red light duration accordingly to ensure there is more of red light on directions which have lesser vehicular density and more of green for routes having a greater number of vehicles. AGT and Cisco's Smart Intersection system leverages Internet-of-Things (IoT) edge analytics to improve road safety and traffic management. Siemens Intelligent Traffic portfolio is made up of the ACS Lite and TACTICS 3 systems. CMU with Pittsburgh is developing smart artificial-intelligence-fueled traffic signals that adapt to changing traffic conditions on the fly. Their startup Surtrac is commercializing the technology. MIT's SENSEable City Lab even developed a solution for traffic light-free intersections based on vehicle-to-vehicle (V2V) communications. Their simulation results demonstrated the efficiency over the traditional traffic light systems. Existing solutions to intelligent transportation systems include sensor-based approaches [2], for example, radar, infrared, and inductive loop detectors. However, sensor-based approaches only work for vehicle counting and not suitable for detecting vehicle types, colors, passengers, moving traces, and pedestrians.

While there are significant research efforts in techniques used to collect traffic data, right from using on-road sensors to floating vehicle data, there exists no real-time standalone system that integrates the process of traffic data collection and analysis to an Intelligent traffic controller. To address these problems, we propose an edge intelligent system that aggregate data at intersections to support various traffic detection, robust object and traffic flow tracking, optimize traffic flow, ensure pedestrian safety, and potential to integrate with existing traffic control and management systems. Specially, we propose a new deep learning model to perform object detection for various type of vehicles, train buses, and pedestrians. We propose algorithms and advanced video analytics technologies based on deep learning results to automatically detect and track the vehicles and pedestrians flows. To reduce the network burdens and eliminate the deployment of network backbone at the intersections, we propose to process the traffic video data at the network edge without transmitting the big data back to the cloud. However, the achievable frame rate on the most advanced embedded systems (NVIDIA Jetson TX2) is only 3 FPS, which is too low to capture the full moving

978-1-5386-7238-9/18/$31.00 ©2018 IEEE
DOI 10.1109/EDGE.2018.00028

trace of the vehicles. To improve the frame rate, we further propose *deep object tracking* algorithm leveraging *adaptive multi-modal models* and make it robust to object occlusions and varying lighting conditions. Based on the deep learning based detection and tracking, we can achieve pseudo-30FPS via *adaptive key frame selection*.

II. System Design

Our camera feed module will take the input from existing CCTV cameras or other off-the-shelf cameras. We perform video transcoding to ensure our processing module get the right format of the video frame. For example, the CCTV camera deployed in the Manhattan, New York utilized the asf format. Using open source video transcoding codecs, we can read from any video format and convert them into the mp4 format with H.264 coding. Our camera feed module also continuously supplies video frames with frame-based flow control to the following computing modules. For example, a 30 fps camera generates thirty frames in one second. Many computing modules cannot process the frames at the rate of 30 fps. To ensure the real-time performance, our camera feed module will automatically adjust the frame rate (f_d) to feed the computing module. To capture the whole object movement within the CCTV camera view and do not miss the target, we need to achieve a minimum frame rate (f_{min}) for the field of view of the camera. To estimate the f_{min}, we can divide the intersection into basic grids, for example, we can use the lane width (12 feet) as the size of the square grid. With an average of 10 lanes on each side of the intersection, we assume that the intersection field of view shall be 240 feet wide. Assuming an average vehicle speed of 50mph (miles per hour) at intersections and we need to capture all lane-entries and lane-exits, we found out that a vehicle/object of interest can cross 73 feet in one second when the field of view is 240 feet. As shown in the Fig. 1, one second moving is equivalent to six grids and thus we can capture $f_{min} = 6$ frames in one second to effectively track vehicles.

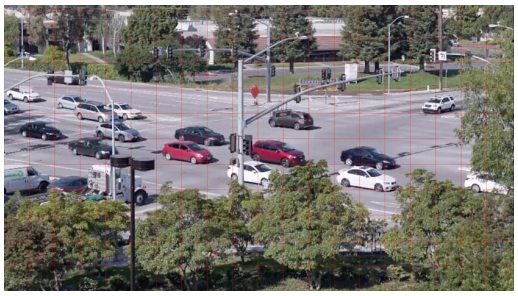

Fig. 1. Grid division of a standard traffic intersection field of view.

To get the lane information, we mark the intersection view and group multiple lanes into a single route via polygonal drawn covering the lane zone. Each route is a pair of lanes since our intelligent traffic model only needs to get appropriate traffic flux over specific route instead of detailed lanes.

III. Object detection and classification module

A. State-of-the-Art Models

To achieve faster, rather than more accurate solutions of object detection, researchers have turned their attention to regression-based models that can generate proposals from convolutional features by simple rules, such as YOLO and YOLOv2 [3]. Google also opened TensorFlow-based object detection APIs recently and includes the implementation of several popular models, for example, SSD with MobileNet, SSD with Inception V2, R-FCN with Resnet 101, Faster RCNN with Resnet 101 [4], and Faster RCNN with Inception Resnet v2. According to a recent work discussing the speed/accuracy comparisons and trade-offs for modern object detectors [5], the choice and the justification of a proper choice of the object detection approach is crucial for different applications.

B. Reinforced YOLOv2

We needed a real-time object detection model that shall give precise results. Among existing models we computed the Average Precision (AP) score [6] to evaluate these models in the intelligent traffic scenarios, including TensorFlow, DIGITS, Caffe, and Darknet. For example, we got AP score of 0.63 for Faster-RCNN and 0.71 for YOLOv2 for one particular classifier, namely car.

Based on darknet, we added additional software layers to support object tracking and traffic flow analytics. To further improve the accuracy of YOLOv2 for traffic detection scenarios, we trained the existing pre-trained YOLOv2 model based on the multi-resolutions of NVIDIA AI City dataset [1], which contains 3 subset of traffic videos taken from 3 different locations including (1) a Silicon Valley intersection, (2) a Virginia Beach intersection, and (3) Lincoln, Nebraska with different video resolutions. The videos are recorded under diverse environmental and lighting conditions, ranging from day and night. About 150, 000 key frames extracted from 80 hours videos are manually annotated with bounding boxes around the objects of interest with corresponding labels. The labels for the datasets are: Car, SUV, SmallTruck, MediumTruck, LargeTruck, Pedestrian, Bus, Van, Group of People, Bicycle, Motorcycle, TrafficSignal-Green, TrafficSignal-Red, TrafficSignal-Yellow.

To further improve the accuracy, we utilized our developed annotation and evaluation tool to evaluate the initial object detection results. We added one UI interface that allows the user to view the visualization results, and adjust the bounding box if the automatic detection results are not correct. Our annotation and evaluation tool will record these error frames. When all error frames are identified, we will use this set of data to reinforce the training of the new YOLOv2 model. Through this approach, we can improve the accuracy gradually. The objection detection results are shown in Fig. 2 under different traffic scenarios.

Fig. 2. Object detection results under different scenarios.

C. Overall Multialgorithmic Tracking Procedure

Algorithm 1 shows our multi-algorithmic tracking procedure for various scenarios. BBD is Bounding Box Detected, BBT is Bounding Box Tracked, *TAlgo* is the desired tracking algorithm to use, *Frame0* is the old frame which is the base for tracking, *FrameCurrent* is the current frame on which we shall track objects detected in frame 0, and *MLV* is the Motion Level Value.

Algorithm 1: Multialgorithmic tracking procedure

Data: Previous Frame in the video pipeline $FramePrev$ and Current frame$FrameCurrent$
Result: Tracked bounding boxes, BBT_List in $FrameCurrent$
$gridVector$=sca5($FramePrev$, $FrameCurrent$);
BBD=BBD_List of $FramePrev$;
$OcclusionRectangles$=findOcclusionsBySegmentation();
while *BBD is non NULL* **do**
  $BBD(MLV)$=calculateMLV($gridVector$, BBD);
  if $FutureObjectPoint(x, y)$=*Kalman() is within* $OcclusionRectangles$ **then**
    if *occlusionPercent(*$OcclusionRectangles$, BBD, $FutureObjectPoint(x, y)) > 80$ **then**
      $TAlgo$=$MIL\_tracker$;
    else
      $TAlgo$=$TPCAM\_OptFlow\_Tracker$;
  else
    if $BBD(MLV) > CONFIG\_THRESHOLD$ **then**
      $TAlgo$=$BOOSTING$;
    else
      $TAlgo$=$TPCAM\_OptFlow\_Tracker$;
  BBT=track(BBD, $FrameCurrent$, $TAlgo$);
  addToList(BBT, BBT_List);
  BBD=BBD(next);
return BBT_List;

D. Error Detection and Failure Recovery

Another major challenge of the tracking is to recover from the error. We need to estimate whether the tracking methodology correctly tracked a target or not. Median Flow tracker [7] demonstrates a novel idea to detect tracking failure scenarios, while most of the other trackers fail to detect failure.Based on the existing solutions of Median Flow tracker, we further employ an effective tracking failure detection algorithm based on object detector output. We utilize the IoU (Intersection over Union) mapping mechanism to evaluate the difference between the tracked bounding box (BB) and the detected BB. We utilize IoU to give our optical flow based tracker the ability to quickly detect tracking failures than any other models in real-world scenarios when subsequent frames being tracked have limited motion of objects. We also add this methodology as a post-processing step to the tracked BB's generated from other object tracking models to further improve the accuracy. The post-processing process of detecting tracking failures is described in tracking-failure detect algorithm, where matching IoU for best fit shall calculate the best and valid Intersection over Union IoU and check for an appropriate mapping between tracked and detected bounding boxes in the *FrameCurrent*.

Final object annotation involves the object ID assignment. In order to assign a unique object ID for the tracked objects, the detection and tracker are run on the same frame and the objects are matched by computing Intersection over Union (IoU) between the two resultant bounding boxes and selecting the matches with the best IoU value, which is the value closest to 1.0. Such a closed mapping allows us to identify undetected tracking failures. Furthermore, the learnt information helps us to preserve the undetected objects which were successfully tracked.

E. Achieving Pseudo-30 FPS

To achieve the real-time performance, i.e., capable to process 30 FPS, we propose to utilize our tracking results to perform *dynamic frame selection* that process a minimum number of images and achieve pseudo-30 FPS. We propose using the multi-algorithmic approach discussed in the section **??** to use an intelligent combination of tracking algorithms driven by a self-adapting algorithm selection technique. This novel approach uses a simple scene change filter between the desired image frames. Depending on the scene change filter decision on whether there was substantial object movement between the frames, we employ Median Flow and Optical Flow techniques or KCF and TLD. Again, within these two branches, depending on whether there is any minor/major occlusion of the target object, we employ optical flow based methods appropriately. The order in which tracking algorithms are used by the branches depends on its ability to detect tracking failure and performance. The optical flow tracking mechanism we use here is equipped with tracking failure detection which is superior to other models.

IV. Evaluation

A. Evaluation Metrics and Object Detection Results

While the obtained results show us the mAP score obtained by Faster RCNN is higher than YOLO, but considering the real-time requirement of the proposed system, we utilize YOLO-based object detection models. The comparison results

TABLE I
THE COMPARISON OF THE MOTA SCORE

	MOTA	MOTP	MOTAL
adaptive frame selection	30.1	74.2	33.5
No skipping	33.0	74.4	36.0

are shown in Fig. 3 with a detailed list of the precision-recall curves for different object categories.

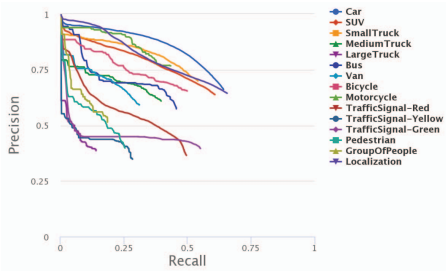

Fig. 3. Results of evaluation for different object categories.

B. The Performance Comparison via Two Different Datasets

To illustrate the robustness of our proposed TPCAM approach, we utilized two datasets (NVIDIA dataset and China dataset) to compare different algorithm in different cases.

C. The MOTA Score of Object Tracking

To evaluate our tracking method, we utilize the MOT score as the performance metric and follow the standard tracking evaluation procedure in the MOT 2015 benchmark [9]. Our achieved MOTA score is 33, which is already in the top tier of the MOTA leading board (most top algorithm's MOTA score is around 30).

D. Real-time Performance

In order to achieve real-time traffic detection and tracking at the network edge (run on NVIDIA TX2). We propose *adaptive frame selection* approaches to achieve pseudo-30 FPS. Table. I shows MOTA score of 33.0 for our tracker without skipping frames, where we achieved an average of 9.6 fps on a desktop with Intel i7 and NVIDIA 1080 GTX GPU. Whereas by skipping frames via *adaptive frame selection*, we can achieve pseudo-30 FPS. The MOT score comparison is shown in Table. I, where we got MOTA = 30.1. The comparison of the FPS is shown in Fig. 4, and we got average FPS = 33.65 when performing adaptive frame selection. With only a small performance drop ($\delta MOTA = 3.55$), we can achieve real-time traffic detection and analytics via edge devices (NVIDIA TX2).

V. CONCLUSION

In this paper, we proposed a real-time object detection and tracking approach, named TPCAM. It can achieve precise object tracking with an adaptive multi-algorithm approach. The precision achieved is superior to the existing models. This

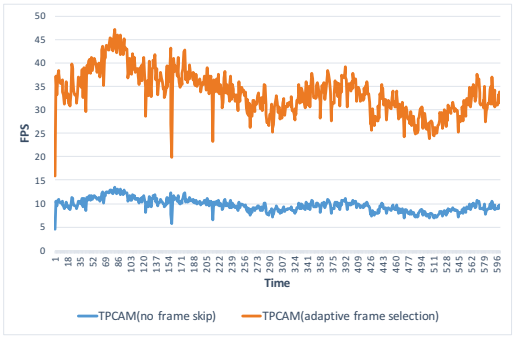

Fig. 4. Results of real-time frame rate.

approach can be easily deployed on an edge device like the NVIDIA TX2, where we could use the available processing power to adequately process the traffic video in pseudo-30 FPS via adaptive frame selection. Our detection and analytic results of vehicle and pedestrian flows will help optimize timings at traffic signals and ensure "safe time" to facilitate movement of pedestrians. In the future, our system can also catch the offenders flouting the traffic norms like jumping the signals, over-speeding, cross lanes, driving without helmet and seat-belt etc.

ACKNOWLEDGMENT

The work presented in this paper is funded by Cisco Systems and National Science Foundation under Grant No. CNS 1637371.

REFERENCES

[1] M. Naphade, D. C. Anastasiu, A. Sharma, V. Jagrlamudi, H. Jeon, K. Liu, M.-C. Chang, S. Lyu, and Z. Gao, "The nvidia ai city challenge," in *2017 IEEE SmartWorld Congress*, ser. SmartWorld'17. Piscataway, NJ, USA: IEEE, 2017.

[2] M. Alam, J. Ferreira, and J. Fonseca, "Introduction to intelligent transportation systems," in *Intelligent Transportation Systems*. Springer, 2016, pp. 1–17.

[3] J. Redmon and A. Farhadi, "YOLO9000 better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016. [Online]. Available: http://arxiv.org/abs/1612.08242

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[5] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," *arXiv preprint arXiv:1611.10012*, 2016.

[6] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[7] Z. Kalal, K. Mikolajczyk, and J. Matas, "Forward-backward error: Automatic detection of tracking failures," in *2010 20th International Conference on Pattern Recognition*, Aug 2010, pp. 2756–2759.

[8] 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*, 2015, pp. 91–99.

[9] L. Leal-Taixé, A. Milan, I. D. Reid, S. Roth, and K. Schindler, "Motchallenge 2015: Towards a benchmark for multi-target tracking," *CoRR*, vol. abs/1504.01942, 2015. [Online]. Available: http://arxiv.org/abs/1504.01942