A Region-and-Trajectory Movement Matching for Multiple Turn-counts at Road Intersection on Edge Device

Duong Nguyen-Ngoc Tran, Long Hoang Pham, Huy-Hung Nguyen,
Tai Huu-Phuong Tran, Hyung-Joon Jeon, Jae Wook Jeon*

Department of Electrical and Computer Engineering,
Sungkyunkwan University, Suwon, South Korea
{duongtran, phlong, huyhung91, taithp, joonjeon, jwjeon}@skku.edu

Abstract

In intelligent traffic systems, vehicle detection and counting have become an important task. The counting information is essential for reducing traffic congestion and improving traffic signal capability. Traditional methods have been focusing on counting vehicles in a single frame or consecutive frames. However, they have not yet considered the movement of interest (MOI) of the vehicles moving in different lanes and directions. This paper proposes a region-and-trajectory movement matching method that aims to detect and count vehicles for each movement on the road. First, the YOLOv5 detection model is used to detect candidate vehicles in the region of interest (ROI). Second, the SORT tracking method associates vehicles of the same instance in consecutive images to create tracked trajectories. Then, the counting method using the combination of MOI regions and predefined movement tracks. Each tracked trajectory is assigned to the corresponding movement id and is outputted to the result file. The efficiency and effectiveness of the proposed method have been evaluated and ranked 3rd on AI City Challenge 2021 Track 1 leaderboard. Further experiments showed that the method could achieve around 120 fps on an NVIDIA Quadro RTX 8000 and 20 fps on an NVIDIA Jetson Xavier AGX.

1. Introduction

In recent years, vehicle detection and counting tasks have become essential in intelligent traffic systems [22]. The counting results can estimate traffic flow in an area, reduce traffic congestion, improve traffic planning and operation. The vehicle counting in traffic cameras presents several challenging issues. First, accurate vehicle detection and tracking in crowded scenes are hard to achieve. Second, identifying vehicles' movements in intersections is difficult because of the combined, crossing, and turning lanes. Third, bad weather conditions and a variety of camera view angles further increase the complexity of the vehicle counting problem.

In order to solve these problems, several studies have been proposed which follow nearly the same pipeline. Firstly, vehicle detection is performed by either deep learning or traditional methods. Secondly, tracking is performed based on the results of the detection module, which can fail when the detection results (bounding boxes) are not accurate. Thirdly, each vehicle trajectory is matched with a predefined movement track on the road. This task is quite challenging, especially at intersections, where many movements are overlapped with each other. Finally, counting is performed for each movement when the vehicle exits the region of interest (ROI). Overall, stability and scalability are a vital concern given the real-world applications on several computational systems.

This paper proposes a region-and-trajectory movement matching method for improving the precision when associating between the tracked vehicles and the predefined movements, which in turn, increasing the accuracy of the vehicle counting task. In this method, motion of interest (MOI) regions and reference trajectories are defined for each movement. For each vehicle, the best optimal movement is identified by matching the reference trajectory and overcoming the corresponding region of each movement direction.Based on the studies [13, 18], we reduce the number of the anchor points in a reference trajectory and use a polygon shape for the MOI region instead of the rectangle zone. Using both techniques, the processing speed can be reduced using fewer anchor points while short tracklets' movements can be identified using the MOI regions. More detail of the proposed method is found in Figure 5. The proposed method has been evaluated on AI City Challenge 2021 Track 1 Dataset A, and ranked 3rd in the leaderboard.

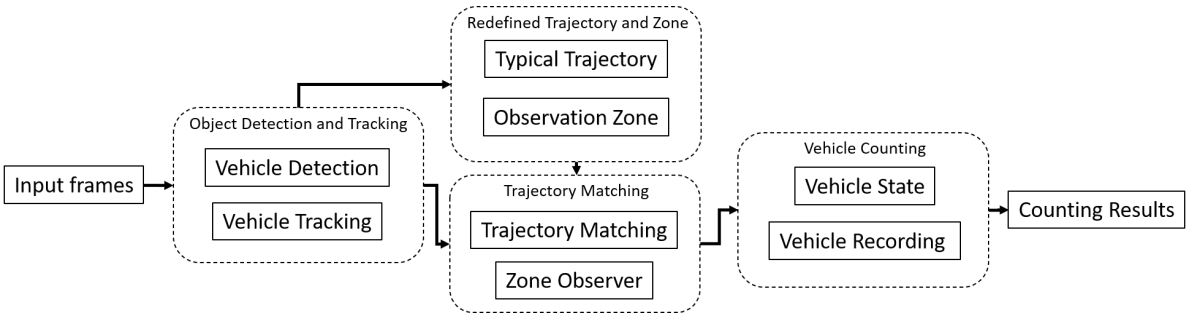


Figure 1. Framework of a region-and-trajectory movement matching method.

Also, the requirements for real-world processing are in high demand, especially on embedded computational devices. Manufacturing companies provide various application-specific integrated circuits such as field-programmable gate arrays (FPGAs) or digital signal processors (DSPs), or processing unit graphics (GPU). In this study, the proposed method has been implemented and tested on an NVIDIA GPU-based computer and NVIDIA embedded computing platforms such as Jetson NX Xavier [4], and Jetson AGX Xavier [5]. The experiments have shown promising performance with around 120 fps on an NVIDIA Quadro RTX 8000 and around 20 fps on an NVIDIA Jetson Xavier AGX.

The rest of this paper is organized as follows. In Section 2, the related works are discussed. The detail of the proposed method is described and discussed in Section 3. In Section 4, the experiments show qualitative and benchmark results of the proposed method. Conclusions are drawn in Section 5.

2. Related Work

2.1. Object Detection

Moving object detection and identification is one of the most fundamental and complex problems in computer vision. In the beginning, moving objects in the image are detected using the background subtraction method. Then, additional features of the objects are provided by applying either SIFT [19] or HOG [7] descriptor. This approach is prone to a high error rate because of the variety in objects' appearances and scale, and also noise and illumination in the image. Another approach is to used traditional classification model such as Support Vector Machines [12] or Random Forest [3]. These models require manually engineered information more information, which is hard to tweak to obtain good results.

The progress of convolution neural networks and deep neural network architectures have made more reliable solutions. It avoids manual feature extraction and uses a data-driven approach that allows machines to learn feature expressions automatically. There are two common categories of object detection, two-stage detection and one-stage detection. Two-stage frameworks divide the detection process into the region proposal and the classification stage, and the well-known models are RCNN[11], Fast RCNN [10], Faster RCNN[24]. On the other hand, one-stage detectors contain a single feed-forward fully convolutional network that directly provides the bounding boxes and the object classification. The widely used models are YOLO [23], SSD [17]. One of the biggest challenges that many object detection methods face is the dilemma between speed performance and accuracy. It finds hard to improve both of them simultaneously. Deploying these heavy object detectors on an edge device is of a major concern, which can balance both the aspects of speed and accuracy, even though there is some high power device from NVIDIA [4, 5] are made. To support and improve the accuracy of these models, many research institutions release popular datasets and benchmarks, including the datasets of PASCAL VOC Challenges[9, 8], MS-COCO Detection Challenge [16], ImageNet Large Scale Visual Recognition Challenge [25].

2.2. Multi-Object Tracking

Multiple Object Tracking (MOT) plays an essential role in video-based application systems [20]. Many existing MOTs are built as the post-processing task of detection models. The tracking could be run offline in traffic analysis or online running real-time processing simultaneously with the camera or video input frame. When running offline, the tracking uses the detection results over the entire frame sequence for the offline methods and then performs global optimizations. The standard offline methods have structure as the graph model, which can be enhanced by using minimum cost flow [27], and subgraph decomposition [26].

On the other hand, the online method follows the tracking-by-detection paradigm. This approach uses the current frame and the previous frames to link detection results while maintaining spatial and temporal consistency. To perform feature association between tracking objects and

new detection objects, Kalman Filter-based [2], and Neural Network [30, 29] have been proposed. These methods require no training and allowing for fast-speed tracking.

2.3. Vehicle Counting

The vehicle counting methods are mainly divided into two categories: density-aware approach[1] and detection-aware approach [6]. The density-aware approach learns the feature of an object to approximate the counting results. However, the drawback of this method is that they only estimate the probable number of objects instead of the exact amount. The detection-aware approach uses the detection and tracking result for counting. It can provide more precise counting results; however, requiring more complex counting logic. For instance, the counter takes attention to the trajectories' properties (for example, traveled length, direction, etc.) to keep the ones that follow the predefined trajectories and reject small tracklets. Therefore, the occluded and not detected can be solved by keeping the trajectory in a consecutive frame.

3. Methodology

3.1. Moving Object Stage

When a moving vehicle travels through the camera field of view, several stages can be defined. Stage management is considered the core of the system, which controls the vehicle's existence in the whole program. Figure 1 shows the stage as the last task of framework. Each tracking object has the related position with the region of interest: $R_m = \begin{cases} -n & out\ ROI \\ 0 & tourch\ ROI \\ n & in\ ROI \end{cases}$ where n is the distance from center of vehicle to ROI. There are six states of vehicle in system:

1. **Candidate**: The new moving object is created and waits to meet the SORT tracking requirement to become the new track.
2. **Confirm**: The moving object is considered as the moving vehicle with its own track
3. **Counting**: The moving vehicle touches the end of the ROI, which consider for counting.
4. **To be counted**: The moving vehicle wait for matching with the appropriate movement id.
5. **Counted**: The moving vehicle has been counted and outputted. It will not be counted even if being detected in later frame.
6. **Exit**: The moving vehicle has just exited the frame and is marked to be deleted.

Algorithm 1 shows the detail of the switching between each states. Each condition in algorithm contains processes the Section 3.2 and Section 3.3

Algorithm 1: State switching algorithm

Input: List of tracks
Output: Updated state of each tracks
for *each tracking object* **do**
  if $moving\_state\ is\ candidate$ **then**
    if $R_m \geq 0 \wedge meet\_track\_requirement$ **then**
      $moving\_state \leftarrow confirm$
  else if $moving\_state\ is\ confirm$ **then**
    if $R_m \leq 0$ **then**
      $moving\_state \leftarrow counting$
  else if $moving\_state\ is\ counting$ **then**
    if $R_m < 0 \vee\ object\_age \geq max\_of\_age$ **then**
      $moving\_state \leftarrow to\_be\_counted$
  else if $moving\_state\ is\ to\_be\_counted$ **then**
    Do the Weighted Vehicle Trajectory Counting (Section 3.3)
    $moving\_state \leftarrow counted$
  else if $moving\_state\ is\ counted$ **then**
    if $R_m \leq 0$ *or* $object\_age \geq max\_of\_age$ **then**
      $moving\_state \leftarrow exiting$
end

3.2. Vehicle Detection and Tracking

3.2.1 Vehicle Detection

Model Architecture. YOLOv5 [15] has several models, from the largest one is YOLOv5x to the smallest one is YOLOv5s. The AI City Challenge 2021 requires real-time processing on an edge device. Therefore, the YOLOv5s6 is chosen for the vehicle detection task. Furthermore, a pre-trained model on COCO is used and is fine-tuned on the manually annotated data in AI City 2021 Track 1 Dataset A.In addition, cameras with the same viewpoints and vehicle scales are grouped and trained together. Training and implementation of the YOLOv5s can extract good detection results on Dataset A. Also, in inference step, the model is inputted with a batch size of 16 images is used. Then, the detection results are put in to a buffer queue for the tracking task.

Object Classes of Interest. According to the challenge rules, the vehicles are detected and classified into two classes:

1. **Car**: sedan car, sport unify vehicle (SUV), van, bus, small trucks such as pickup truck, UPS mail trucks, etc.

2. **Truck**: medium trucks such as moving trucks, garbage trucks, large trucks such as a tractor-trailer, 18-wheeler, etc.

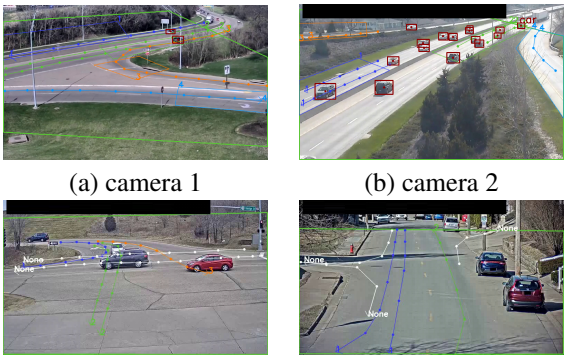
3.2.2 Online Multi-Vehicle Tracking

After a detected vehicle enters the region of interest (ROI), a unique id is assigned. The tracking module associates the currently tracking vehicles with newly detected vehicles and maintains the unique id of a vehicle when it is moving through the camera. Besides, the tracked locations of the vehicle in consecutive frames are used to define its trajectory. The trajectory is later used to associate the movement of the vehicle.

After obtaining the batch's detection results, the SORT [2] method is used as the online multi-vehicle tracking tracker. Even though other trackers, such as DeepSORT [30], are better. However, they require higher memory space while processing which can reduce the overall processing speed. Therefore, additional processing time can be saved for other tasks.

The SORT algorithm is a fast online multi-object tracking, and it can achieve real-time processing. The method associates already detected objects across different frames based on the coordinates of the detected bounding boxes. The vehicle's trajectory is relatively simple in the traffic camera, such as a simple straight line or curve, since the camera's position is high above the road. The SORT method uses efficiency algorithms such as Kalman Filter and Hungarian algorithm for the tracking components to achieve real-time processing.

The Kalman Filter is used for motion prediction. The algorithm uses the unmatched detection results to initialize the tracking state as a new target and the matched detection results for updating the existing target's tracking state. State-space in each target is defined in the dimensional state space $(u, v, s, r, \dot{u}, \dot{v}, \dot{s})$, where u and v stand for the horizontal and vertical pixel 2D location of the center of the target, and the scale s and r represent the scale (area) and the aspect ratio of the object's bounding box respectively. Standard Kalman Filter with constant velocity motion and linear observation model assign target object the tracklet k, which is used for following counting task. When assigning new detection results to exist targets, the shape of each target's bounding box is estimated by predicting its new position in the current frame. The allocation cost matrix is then calculated as the IOU (intersection) distance between each detection and all predicted bounding boxes of the existing target. The assignment is solved optimally using the Hungarian algorithm.



(a) camera 1 (b) camera 2

(c) camera 13 (d) camera 16

Figure 2. Visualization of reference trajectories and observation zone of camera 1, camera 2, camera 13, and camera 16. The color trajectory represent for each reference trajectories, and the white trajectory stand for the uncounted movement.

3.3. Region-and-Trajectory Movement Matching

After obtaining the series of tracklets result from the detection and tracking process, the tracklets have to be matched with a predefined reference movement. This section describes a region-and-trajectory movement matching method to count vehicle in each movement direction. This method has been inherited from [13, 18] and has been further extended to improve the robustness. The proposed method consists of 2 separate vehicle movement identification method, namely: reference trajectory and movement of interest region.

3.3.1 Reference Trajectory

Reference Trajectory. The rule of the challenge defines several movement directions for each camera. Based on this given information, reference trajectories are defined for each movement directions. Follows the examples in [28], the reference trajectories are modeled by accumulating and averaging several moving vehicles' trajectories in the camera. Then, the number of anchor points in the reference trajectories are subsequently reduced until the tracking robustness can no longer be maintained. The results are the set of reference trajectories with fewer points that can increase the processing time while still maintaining the tracking robustness as shown in Figure 2. A completed reference trajectories set $Traj_G = [Traj_{g_1}, Traj_{g_2}, \ldots, Traj_{g_k}]$, where trajectory $Traj_{g_i}$ stands for the i_{th} defined movements. Each reference trajectory has the set of points $Traj_i = \left[p_m^1, p_m^2, ..., p_m^n\right]$, which would be used to match of vehicle's tracklet.

In additions, further tweaks were performed for some distinct cases to further reducing the processing time. In most cases, the reference trajectories are kept from the entrance and the exit for each movement as shown in Figure

Algorithm 2: Find best match reference trajectory

Input: Tracklet $Traj_m$
Reference trajectories $Traj_G$
Distance threshold D_thr
Output: The identity of the best match reference trajectory or none of them
for *each* $Traj_g$ *of* $Traj_G$ **do**
  for *each* sub_m *of* $Traj_m$ **do**
    Compute distance $d_{sub_m,g}$ (Equation 1)
    if $d_{sub_m,g} > D_thr$ **then**
      break
    else if $moving\_state\ is\ candidate$ **then**
      $min_{sub} \leftarrow g_{id}$
  end
  if $min_{sub} < min_m$ **then**
    $min_m \leftarrow min_{sub}$
end
if $Traj_m\ overcome\ Zone_g$ **then**
  return $Traj_{min_m}$
return $None$

2a. Meanwhile, in some case, the movement directions do not intersect with each other, therefore, only the portion at the end of each reference trajectory is kept as shown in Figure 2b. The fewer anchor points, the faster processing speed. Moreover, in some cameras, "None" reference trajectories (as shown in Figure 2c, d) are defined to remove unwanted vehicle's movements as defined by the rules. Figure 5 shows all references trajectories in all cameras.

Matching with Reference Trajectory. Given the tracklets of the vehicles, each of them has $Traj_m = [p_m^{t_1}, p_m^{t_2}, ..., p_m^{t_n}]$, where $p_m^{t_i} = (x_m^{t_i}, y_m^{t_i})$ is the coordinate point of the vehicle, t_1 is the time when the vehicle in candidate state, and t_n is the time when the vehicle in the counted state. The Hausddorff distance [14] is used to find the best match between the tracklet $Traj_m$ with the reference trajectory $Traj_G$, and use Euclidean distance for each point in both set. Let $d_H(m, g_i)$ denote the Hausdorff distance between of tracklet of vehicle m and reference trajectory g_i. However, the Hausdorff distance considers two set of points as the disordered sets; Therefore, $Traj_m$ are divided into subset from the t_n to t_1 to choose the best match even if any lost track while tracking. The distance between subset of tracklet m and the reference trajectory $Traj_g$ is:

$$d_{sub_m,g} = \min_{sub_m \in \mathrm{Traj}_m} d_H(sub_m, g) \quad (1)$$

3.3.2 Movement Of Interest Region

Addition to the reference trajectories, several movement of interest (MOI) regions on the road, one for each movement direction. The MOI region's primary purpose is to capture tracklets that are too short (i.e., trajectory length) to match with any reference trajectories. The main reason for the short tracklets can be from miss-detection, fast-moving vehicles, or new vehicles breaks out from occlusion, etc. When a vehicle has been moving through the MOI region, it is assigned with a corresponding movement id.

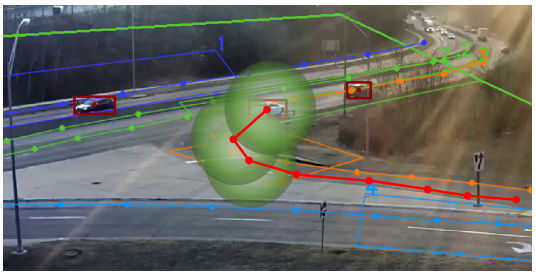


Figure 3. Visualization of region-and-trajectory movement matching for find the best match reference trajectory for the vehicle's tracklet. The red dot line represent for vehicle tracklet, the green circle with gradient color from center to outside is weight shape-based matching, other color dot lines stand for reference trajectories.

Overall, both techniques are used in vehicle movement identification. The Algorithm 2 show the detail of the overall matching method. Each vehicle's movement id is being calculated until the vehicle reaches the end of the ROI or out of the image frame. After finding the best match for tracklet, t_n is the used. Finally, the vehicle movement id and the exiting time is recorded and outputted to the result file.

4. Experiments

4.1. System Implementation

A vehicle detection and counting system has been implemented for as shown in Figure 8. The pipeline utilizes buffer queue and multi-thread processing to parallel the bottle-neck between the detection and tracking stage. In particularly, the detector is performed on a separated thread and the detected bounding boxes are buffered in a queue. The detected bounding boxes are grouped together by the frame id in order to maintain the temporal consistency. The tracker, running on another thread, gets the detection results and performs it own task as soon as the buffer queue is available. The detector runs on GPU while both tracker and counter run on CPU. The whole system has been implemented and tested on a NVIDIA Quadro RTX 8000 with 48GB memory and two-thread Intel i9-9900X 3.50GHz.

4.2. Datasets

The AI City 2021 Track-1 dataset data set contains 31 video clips captured from 20 single camera views; some cameras were captured in different weather (including rain, snow) and light conditions (including dawn, clear sky).

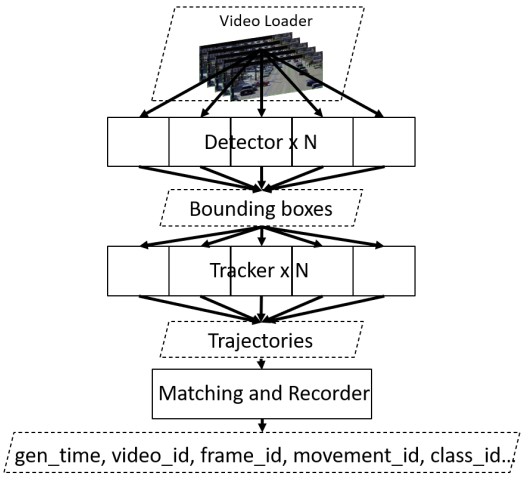


Figure 4. System Pipeline.

Each camera view comes with detailed documentation describing the region of interest and movement of interest. The next section provides experimental results for data set A and our final score of the leaderboard on Dataset A. The detection model is materialized on the AI City 2021 Track 1 Dataset A using a pre-learned model for COCO. A total of 15500 frames is fully annotated from all the videos in the dataset, and only cars and trucks are labeled. The detail of the label can be found in Section 3.2.1

4.3. Evaluation Metrics

According the rules of challenge, there is no additional data for train or self-testing, we only provide the result of final leaderboard. The following section shows the evaluation metrics for this challenge using efficiency score and effectiveness score with along parameter: $S_1 = \alpha S_{efficiency} + \beta S_{effectiveness}$, where $\alpha = 0.3$, $\beta = 0.7$. $S_{efficiency}$ corresponds to the processing time in whole videos. $S_{effectiveness}$ reply on a weighted average of normalized weighted root mean square error scores *nwRMSE* across all generate time, videos, movements, and vehicle classes in the test set, with proportional weights based on the number of vehicles of the given class in the movement.

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

The efficiency score bases on the total execution time T as

$$S_{1,efficiency} = \max\left(0, 1 - \frac{T \times base\_factor}{1.1 \times video\_time}\right) \quad (3)$$

As can be seen on Equation 3 from previous year challenge [21] and this year challenge, the factor of denominator replace by "1.1" instead of "5" and the result must include the generate time, which means this year, they focus more on the time processing. Moreover, one of the requirements is that the framework can run on an edge device, the time consuming and memory management are put higher rank.

Rank	Team ID	Score
1	37	0.9467
2	5	0.9459
3	**8(Ours)**	**0.9263**
4	19	0.9249
5	118	0.9235

Table 1. Top 5 overall score in the AI City 2021 Track 1 learderboard.

mwRMSE	S1_Effectiveness	S1_Efficiency	S1
3.6259	0.9287	0.9209	0.9263

Table 2. The detailed evaluation of the proposed method.

Device	Batch size	Queue	FPS
COMPUTER	32	10	120
Jetson Xavier AGX	16	5	20
Jetson Xavier NX	8	5	12

Table 3. Speed performance.

4.4. Quantitative Result

The Table 1 shows the comparisons of overall scores. After checking, we still have failures caused by certain reasons: The number of missing detection of a truck is still high because the ratio appears vehicle is much different. The hard weather condition, such as rain, makes blur in camera capture image, which cause the detector hardly get the good result. The overlap with vehicle results in the miss track in tracker session.

4.5. Speed Performance

The main goal of the AI City Challenge 2021 Track 1 is to achieve real-time processing on edge devices. Therefore, the whole system has been tested on two NVIDIA development boards: the NVIDIA Jetson Xavier NX [4] and the NVIDIA Jetson AGX Xavier [5]. Because the processing unit and memory of each device is different from the computer, the models' hyperparameters have to be adjusted to optimize for each device. The Table 3 shows detail of parameters and the speed result.

5. Conclusions

In this paper, a region-and-trajectory movement matching method have been proposed. The method has been shown to successfully define the corrected movement di-

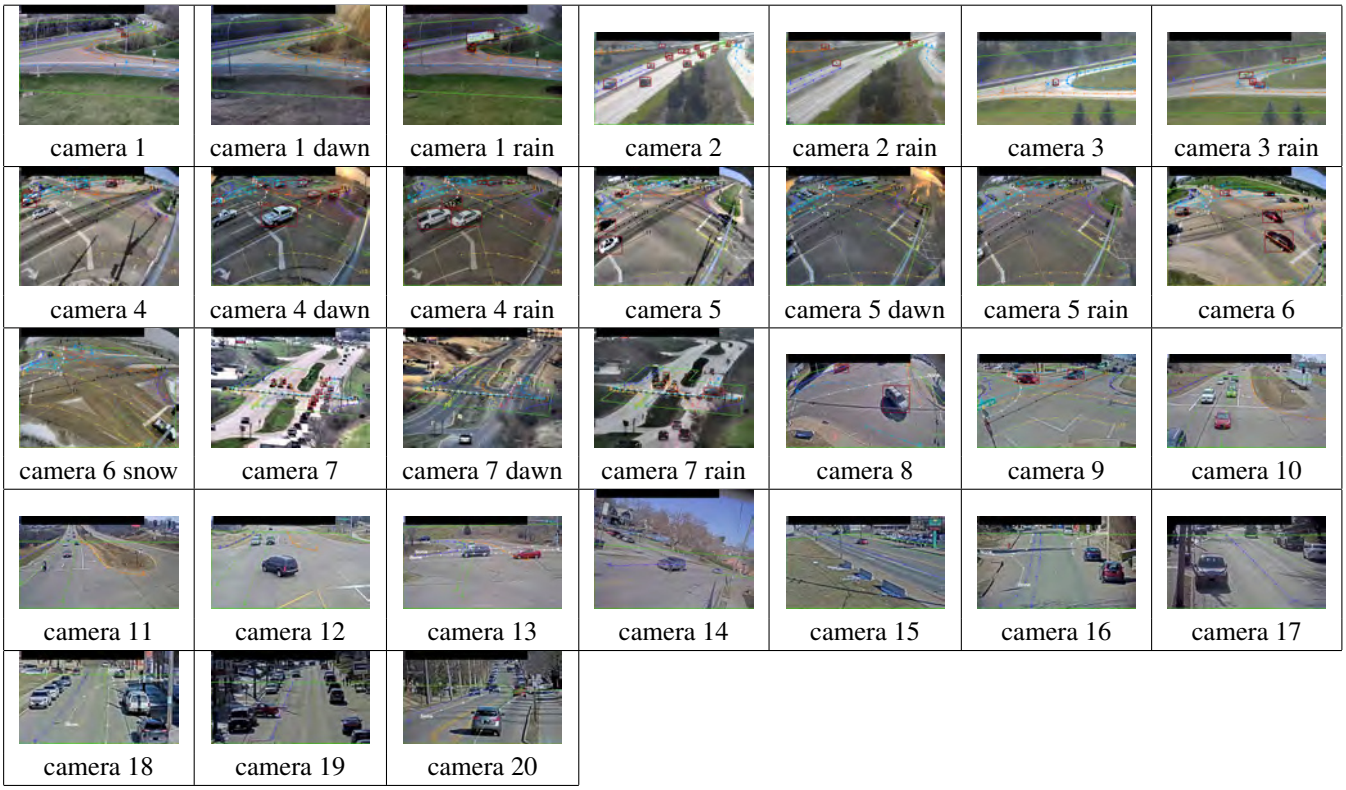


Figure 5. Visualization of results on sample frames from different scenes and different conditions, the color trajectories and number represent for the movement of interest defined at the beginning.

rection in a vehicle counting task. The proposed method's performance shows its effectiveness and efficiency in determining a vehicle's route in different camera views. The system achieved third place on AI City Challenge 2021 Track 1 Dataset A (Track 1: Number of vehicles by the class at multiple intersections). For future work, we will improve each of the components even further, both in speed and robustness. For example, we will address typical detection errors, e.g., false positives due to dynamic or reflective background and lack of detection is too large or small vehicles. Moreover, we learn how to build better models for the route that can adapt to the new scenes and weather conditions.

6. Acknowledgments

This work was supported by Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MIST) (2021-0-01364, An intelligent system for 24/7 real-time traffic surveillance on edge devices)

References

[1] Javier Barandiaran, Berta Murguia, and Fernando Boto. Real-time people counting using multiple lines. In *2008 Ninth International Workshop on Image Analysis for Multimedia Interactive Services*, pages 159–162. IEEE. 3

[2] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468, 2016. 3, 4

[3] Leo Breiman. Random forests. 45(1):5–32. 2

[4] Nvidia Coorporation. Jetson xavier nx developer kit, 2020. [Online; accessed 01-01-2021]. 2, 6

[5] Nvidia Coorporation. Jetson agx xavier developer kit. https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit, 2021. [Online; accessed 01-01-2021]. 2, 6

[6] Zhe Dai, Huansheng Song, Xuan Wang, Yong Fang, Xu Yun, Zhaoyang Zhang, and Huaiyu Li. Video-based vehicle counting framework. 7:64460–64470. 3

[7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE. 2

[8] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, Jan. 2015. 2

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) chal-

lenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010. 2

[10] Ross Girshick. Fast r-CNN. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448. IEEE. 2

[11] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587. IEEE. 2

[12] M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support vector machines. 13(4):18–28. 2

[13] Hung-Min Hsu, Tsung-Wei Huang, Gaoang Wang, Jiarui Cai, Zhichao Lei, and Jenq-Neng Hwang. Multi-Camera Tracking of Vehicles based on Deep Features Re-ID and Trajectory-Based Camera Link Models. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 416–424, Long Beach, CA, USA, June 2019. 1, 4

[14] D.P. Huttenlocher, G.A. Klanderman, and W.J. Rucklidge. Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, Sept. 1993. 5

[15] Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode, ChristopherSTAN, Liu Changyu, Laughing, Tkianai, YxNONG, Adam Hogan, Lorenzomammana, AlexWang, Ayush Chaurasia, Laurentiu Diaconu, Marc, Wanghaoyang, Ml5ah, Doug, Durgesh, Francisco Ingham, Frederik, Guilhen, Adrien Colmagro, Hu Ye, Jacobsolawetz, Jake Poznanski, Jiacong Fang, Junghoon Kim, Khiem Doan, and Lijun Yu. ultralytics/yolov5: v4.0 - nn.SiLU() activations, Weights Biases logging, PyTorch Hub integration, Jan. 2021. 3

[16] 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 David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, volume 8693, pages 740–755. Springer International Publishing. Series Title: Lecture Notes in Computer Science. 2

[17] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot MultiBox detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, volume 9905, pages 21–37. Springer International Publishing. Series Title: Lecture Notes in Computer Science. 2

[18] Zhongji Liu, Wei Zhang, Xu Gao, Hao Meng, Xiao Tan, Xiaoxing Zhu, Zhan Xue, Xiaoqing Ye, Hongwu Zhang, Shilei Wen, and Errui Ding. Robust Movement-Specific Vehicle Counting at Crowded Intersections. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2617–2625, Seattle, WA, USA, June 2020. IEEE. 1, 4

[19] David G. Lowe. Distinctive image features from scale-invariant keypoints. 60(2):91–110. 2

[20] Brendan Tran Morris, Cuong Tran, George Scora, Mohan Manubhai Trivedi, and Matthew J. Barth. Real-time video-based traffic measurement and visualization system for energy/emissions. 13(4):1667–1678. 2

[21] Milind Naphade, Shuo Wang, David C. Anastasiu, Zheng Tang, Ming-Ching Chang, Xiaodong Yang, Liang Zheng, Anuj Sharma, Rama Chellappa, and Pranamesh Chakraborty. The 4th ai city challenge. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, page 2665–2674, June 2020. 6

[22] Long Hoang Pham, Hung Ngoc Phan, Nhat Minh Chung, Tuan-Anh Vu, and Synh Viet-Uyen Ha. A robust multi-class vehicle detection and classification algorithm for traffic surveillance system. In *2020 RIVF International Conference on Computing and Communication Technologies (RIVF)*, pages 1–6. IEEE. 1

[23] Joseph Redmon and Ali Farhadi. YOLO9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525. IEEE. 2

[24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 91–99, Cambridge, MA, USA, 2015. MIT Press. 2

[25] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 2

[26] Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. Subgraph decomposition for multi-target tracking. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5033–5041. IEEE. 2

[27] Xinchao Wang, Engin Turetken, Francois Fleuret, and Pascal Fua. Tracking interacting objects using intertwined flows. 38(11):2312–2326. 2

[28] Zhihui Wang, Bing Bai, Yujun Xie, Tengfei Xing, Bineng Zhong, Qinqin Zhou, Yiping Meng, Bin Xu, Zhichao Song, Pengfei Xu, Runbo Hu, and Hua Chai. Robust and Fast Vehicle Turn-counts at Intersections via an Integrated Solution from Detection, Tracking and Trajectory Modeling. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2598–2606, Seattle, WA, USA, June 2020. IEEE. 4

[29] Zhongdao Wang, Liang Zheng, Yixuan Liu, Yali Li, and Shengjin Wang. Towards real-time multi-object tracking. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, volume 12356, pages 107–122. Springer International Publishing. Series Title: Lecture Notes in Computer Science. 3

[30] 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. 3, 4