Zero-VIRUS*: Zero-shot VehIcle Route Understanding System for Intelligent Transportation

Lijun Yu[1], Qianyu Feng[1,2], Yijun Qian[1], Wenhe Liu[1], and Alexander G. Hauptmann[1]

[1]Language Technologies Institute, Carnegie Mellon University
[2]University of Technology Sydney
lijun@cmu.edu, qianyu.feng@student.uts.edu.au
{yijunqia, wenhel}@andrew.cmu.edu, alex@cs.cmu.edu

Abstract

Nowadays, understanding the traffic statistics in real city-scale camera networks takes an important place in the intelligent transportation field. Recently, vehicle route understanding brings a new challenge to the area. It aims to measure the traffic density by identifying the route of each vehicle in traffic cameras. This year, the AI City Challenge holds a competition with real-world traffic data on vehicle route understanding, which requires both efficiency and effectiveness. In this work, we propose Zero-VIRUS, a Zero-shot VehIcle Route Understanding System, which requires no annotation for vehicle tracklets and is applicable for the changeable real-world traffic scenarios. It adopts a novel 2D field modeling of pre-defined routes to estimate the proximity and completeness of each track. The proposed system has achieved third place on Dataset A in stage 1 of the competition (Track 1: Vehicle Counts by Class at Multiple Intersections) against world-wide participants on both effectiveness and efficiency, with a record of the top place on 50% of the test set.

1. Introduction

The rapid technological change in all the industries galvanizes the development of systems that allow high-autonomous production and control. Intelligent utilities compose the missing link to smart city construction. Particularly, transportation is one of the domains remaining underdeveloped, which can largely benefit from the collection and mining of traffic information to facilitate further unified scheduling and dispatching. Pedestrians and vehicles make up the main components of traffic executing different activities. Although tremendous traffic rules have been formed to ensure efficient operation and safety, autonomous traffic systems are indispensable. To date, a large number of datasets and algorithms [5, 39, 30] are devoted to the study of pedestrians' tracking, identification, and action prediction. In contrast, vehicles are more convenient

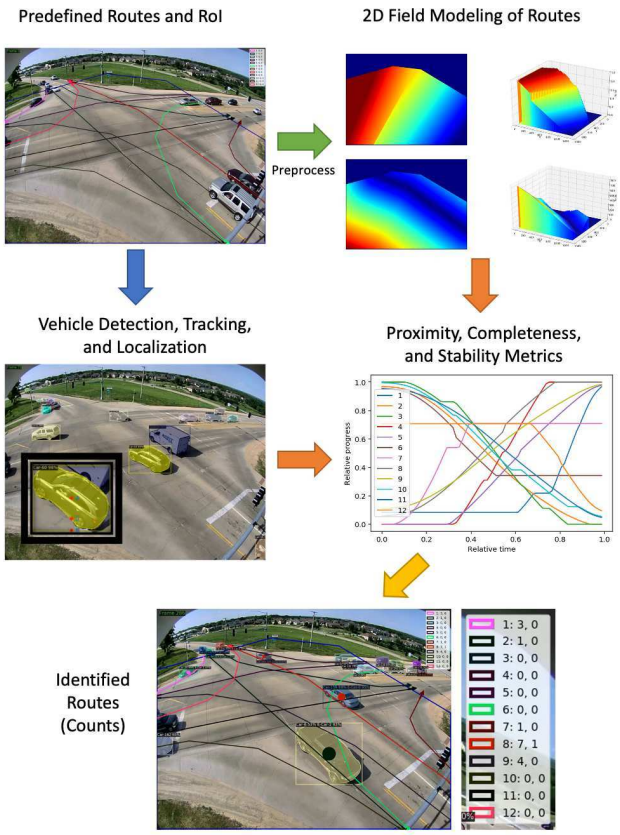

Figure 1: System Overview

*Written in the era of Coronavirus Disease 2019 (COVID-19), with a sincere hope for a better world.

to identify and uniformly dispatch than pedestrians with the sensor data recorded. Unfortunately, the lack of supervised information and high-quality models constrain those benefits.

Proposed by the AI City Challenge workshop[1], AIC20_track dataset provides videos collected from different city-scale traffic monitoring cameras. As a new topic, Vehicle Counts by Class at Multiple Intersections is proposed upon the AIC20_track dataset, where four-wheel cars and freight trucks are the target objects. With movements pre-defined for each camera scene, *e.g.*, left-turning, right-turning, and straight through the given traffic intersection, the objective is to count the number of each type of vehicle by the class of movement. In order to benefit from the practical value of the outcome from the vehicle countings, not only the effectiveness and but also the efficiency of the algorithm will be measured in the final score.

Facing the dire need of algorithms tackling the vehicle counts task, the poor quality of data comparing to the well-formed and fully annotated tracking datasets [10, 35, 22] sets a considerably high threshold. The difficulties existing in the annotation for the humongous vehicle in the low-quality camera hinder the development of vehicle tracking and counting tasks. Beyond that, it is also confronted with challenges of the variance of scenes and cameras, *e.g.*, fish eye camera, long-range PTZ camera, tilted view cast in unsettled weather. Furthermore, the key challenge lies in how to effectively fuse information of target vehicles for better tracking over proper handling of the variabilities and uncertainties of camera view.

So as to effectively count target vehicle with different movements, it is noted that the first barrier to break is the lack of annotation either for the vehicle or the infrastructure, *e.g.*, region of transport road, green belts, road signs. The proposed Zero-shot VehIcle Route Understanding System (Zero-VIRUS) is built upon tracking-by-detection paradigm and designed for arbitrary traffic cameras with the modeling of designated routes in the 2D field.

Our main contributions exist in the following aspects:

1. We propose a novel 2D field modeling of vehicle routes as a better representation for identification purposes.

2. A zero-shot vehicle route understanding system is designed for intelligent transportation, which demonstrates high effectiveness with zero annotation of vehicle tracklets.

3. Our system achieves third place on Dataset A in Stage 1 of the competition (Track 1: Vehicle Counts by Class at Multiple Intersections) and the top place on 50% of the dataset.

[1] https://www.aicitychallenge.org/

2. Related Work

Vehicle Counts by Class at Multiple Intersections requires simultaneous vehicle detection, multi-target tracking, and movement classification in real city-scale traffic camera videos. Our work can be summarized in the following aspects.

Vehicle Detection Recently, deep-learning-based methods were proposed due to its great success in many computer vision tasks [4, 19]. Object detection and tracking in videos have attracted a surge of interest lately. SSD [20], YOLOv3 [28] are designed in a single-stage style which performs the extraction of feature and location end-to-end. There are also many successful proposal-based methods [11, 29, 18, 17, 27] which perform more accurately by optimizing regressions on the generated proposals.

Multi-Object Tracking Multiple Object Tracking (MOT), or Multiple Target Tracking (MTT), plays an important role in computer vision, which is widely used in video-based systems such as in [38, 21, 1]. Most of the recent methods for MOT are CNN-based. Many approaches adopt post-processing elaborately designed for the final tasks on top of the detection models. MOT methods mostly follow the tracking-by-detection paradigm [3, 2, 23, 24], to associate detection results per-frame or tracklets into longer tracks with consistency across space and time. DeepSORT [13] is a simple feed-forward network with no online training required, which allows tracking objects at a fast speed. It also aids in tracking novel objects by learning a generic representation of the motion and appearance of objects in different frames. Toward Real-time MOT [33] proposes a shared model to perform the target localization and appearance embedding learning which takes anchor classification, bounding box regression, and embedding learning as multi-task learning. In order to achieve temporal consistency, the tracking results of the previous frames are used as a guidance [23, 13, 33, 9]. Methods [14, 16] have been proposed to leverage spatial-temporal contextual information.

Vehicle Count Vehicle count by movement class in the city-scale video is a new topic in AI City Challenge 2020. The number of vehicles in a city camera view with different movements, *e.g.*, going straight or turning, is quite valuable information for real-time control in a traffic system. A Kalman-Filter-based method [32] is proposed to produce reliable estimates of real-time measurements of detectors. Pang *et al*. [25] proposed a method for counting the number of vehicles with occlusions, based on the resolvability of each occluded vehicle. Traffic videos from real-world cameras usually come with relatively lower quality. What's

more, there is no available annotation for the vehicles to count. Zero-shot learning has been a long-standing problem in computer vision tasks [31, 15, 7, 8]. Thus, a naive combination of detector and tracker can perform poorly, while standard vision algorithms such as background modeling and optical flow can work in a limited setup. Thus, we choose to resolve this dilemma by first identifying the target vehicle routes for pre-defined movements. With the tracks of target vehicles aggregated by the tracking-by-detection paradigm, the tracks are identified and regressed to the modeled routes for the pre-defined movements.

3. Method

In this paper, we propose Zero-VIRUS, a zero-shot system to identify vehicle routes in still-camera videos for intelligent transportation purposes. With 2D-field modeling of designated routes, this system is built upon zero known vehicle tracks and designed for arbitrary traffic cameras. Zero-VIRUS consists of two stages, where we first obtain vehicle trajectories and then identify them with field modeling of pre-defined routes.

3.1. Vehicle Detection and Tracking

In a video sequence, we detect vehicles on each frame and track their locations along time. State-of-the-art object detection and tracking techniques are adapted for the traffic scenarios to produce initial trajectories. Then we enhance the trajectories through localization, smoothing, and filters to provide a more explicit representation for the latter stage.

3.1.1 Vehicle Detection

Model Architecture We adopt the instance segmentation network Mask R-CNN [12] with a ResNeXt-101 [36] feature pyramid network [17] backbone as our frame-level vehicle detection model. On a given frame I_t at time t, it outputs a series of detected vehicles $\{V_{t,i}\}$ with vehicle class $c(V_{t,i})$, confidence score $s(V_{t,i})$, bounding box $B(V_{t,i})$, and segmentation mask $M(V_{t,i})$.

Object Classes of Interest In the traffic scenarios, we mainly focus on vehicles of the following classes. The corresponding classes in the Microsoft COCO [19] dataset are listed in the parentheses.

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

2. *Truck*: medium trucks such as moving trucks, garbage trucks; large trucks such as tractor-trailers, 18-wheeler, etc. (COCO: Truck)

Our definition of *Truck* is inconsistent with COCO in that we do not include small trucks such as pickups. This is intended to be fixed partially by the inter-class non-maximum suppression and the track label assignment.

Weighted Inter-class Non-maximum Suppression Some vehicles, especially those small trucks, could result in double detections of both *Car* and *Truck*. In such cases, we would prefer to remove the *Truck* detection. Therefore, we additionally apply a weighted inter-class non-maximum suppression(NMS) [6] on the detections. Object classes are assigned different weights as

$$w_c = \begin{cases} 1, c = \textit{Car} \\ 0, c = \textit{Truck} \end{cases} \tag{1}$$

All detections are sorted in the descending order according to a weighted confidence score as

$$\hat{s}(V_{t,i}) = s(V_{t,i}) + w_{c(V_{t,i})} \tag{2}$$

Then we select the detection one by one and skip if there exists a detection with intersection over union (IoU) of at least IoU_{nms}. Samples of detected vehicles are shown in Figure 2.

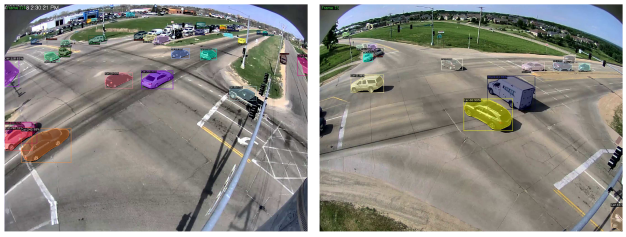

Figure 2: Sample results of vehicle detection

3.1.2 Online Multi-Vehicle Tracking

Feature Extraction To reduce computational complexity, we directly use the Mask R-CNN backbone to extract the feature representation of a detected object. The feature maps are aligned by region of interest (RoI) pooling [11] into size s_{roi} according to the bounding box. Then an average pooling into size s_{avg} is applied to obtain the final feature vector.

Online Association We adopt the association algorithm from [33] to get the vehicle ID $ID(V_{t,i})$ for each detection. Basically, new detections are assigned to existing tracklets based on feature similarity and compliance with spatial constraints. Since we are tracking *Car* and *Truck* together, a tracked vehicle is classified as *Truck* when it is detected as *Truck* for at least f_{truck} fraction of all frames.

3.1.3 Trajectory Enhancement

Localization with Segmentation Given the bounding box of a vehicle $V_{t,i}$ in the top-left bottom-right representation as

$$B(V_{t,i}) = \begin{bmatrix} x_0 & y_0 & x_1 & y_1 \end{bmatrix}_{t,i} \tag{3}$$

its location is usually defined at the center of the bounding box

$$P_A(V_{t,i}) = \begin{bmatrix} \frac{x_0+x_1}{2} & \frac{y_0+y_1}{2} \end{bmatrix}_{t,i} \tag{4}$$

or at the bottom center

$$P_B(V_{t,i}) = \begin{bmatrix} \frac{x_0+x_1}{2} & y_1 \end{bmatrix}_{t,i} \tag{5}$$

As shown in Figure 3, these two definitions both have obvious shortcomings for ground tracking of vehicles. $P_A(V_{t,i})$ is typically on the vehicle body, but far from the ground. $P_B(V_{t,i})$ is more likely on the ground, but can be off the vehicle.

Figure 3: Comparison between localization points

Therefore, we propose a segmentation based location point defined as

$$P_C(V_{t,i}) = \begin{bmatrix} \frac{x_0+x_1}{2} & \max(\frac{y_0+y_1}{2}, y_m) \end{bmatrix}_{t,i} \tag{6}$$

where

$$y_m = \arg\max_y M(V_{t,i})_{\frac{x_0+x_1}{2},y} \tag{7}$$

It selects the bottom of its segmentation mask at the center column of its bounding box, with a constraint of no higher than its center row.

As the visual size of an object changes according to its distance to the camera, we also record the diagonal of each bounding box as a scale factor along with its location.

$$S(V_{t,i}) = \| \begin{bmatrix} x_1 - x_0 & y_1 - y_0 \end{bmatrix} \|_2 \tag{8}$$

Interpolation and Smoothing For a tracked vehicle with ID x, we now have its full available history as

$$\mathbf{H}_x = \{V_{t,i} \mid ID(V_{t,i}) = x\} \tag{9}$$

Due to occlusions and detection failure, a tracked vehicle may be lost for a short period before it is re-found in later frames. This would result in gaps in $\mathbf{H}_x$ while viewing in the chronological order. Therefore, we apply linear interpolation to fill in the gaps for values of P_C and S with adjacent available frames.

Also, to deal with jittering of the detected objects, a 1D-gaussian smoothing of standard deviation σ_g is applied to the values of P_C and S after interpolation.

Movement and Region Filter Vehicles may stop somewhere for a while due to traffic lights or jams. For route identification, we would like to filter out the stopped period from the trajectory. We calculate the local average speeds of the trajectory with a sliding-window of size w_v using P_C. Then points with a speed lower than v_{min} are removed from the track.

Additionally, a region filter is applied to select the part of the trajectory within the region of interest (RoI), as shown in Figure 4. The RoI is defined as the inside of a polygon, which ensures the image border to be outside. The region filter is applied at the end to prevent loss of information for the previous operations. At this point, trajectories with a length of less than l_{min} would be discarded. After all these filters, we get an enhanced trajectory as $\mathbf{T}_x = (\mathbf{P}_x, \mathbf{S}_x)$, where $\mathbf{P}_x = \{P_C\}$ are the locations and $\mathbf{S}_x = \{S\}$ are the scale factors.

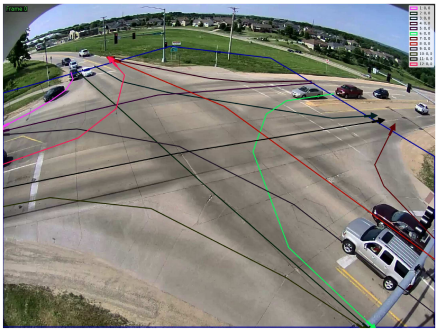

Figure 4: Region of interest (area marked with blue line) and pre-defined routes (colored lines with arrows)

3.2. Route Modeling and Identification

To realize this route identification task with zero training data while also provide a straightforward generalization to arbitrary traffic scenes, we employ 2D-field modeling of the pre-defined routes to identify detected vehicle tracks. This allows our system to be applied to any new camera solely with the route definition.

3.2.1 Route Modeling

Route In this task, we are dealing with videos from still cameras with a pre-defined RoI. A route $\mathbf{R}_i$ is described as a polyline that consists of n points.

$$\mathbf{R}_i = \begin{bmatrix} P_1^i & P_2^i & \dots & P_n^i \end{bmatrix}_i^T \tag{10}$$

where P_1^i and P_n^i should be outside the RoI, as illustrated in Figure 4. This definition ensures that a trajectory from the previous stage would be spatially consistent with its corresponding route.

Proximity Field For each point X in the view space, we would like to estimate its distance to a specific route $\mathbf{R}_i$. This is achieved by defining a 2D proximity field $\mathbf{F}_{p,\mathbf{R}_i}$, as illustrated in Figure 5.

$$\mathbf{F}_{p,\mathbf{R}_i}(X) = \min_j d(X, \overline{P_j^i P_{j+1}^i}) \tag{11}$$

where $d(X, \overline{P_j^i P_{j+1}^i})$ is the point-segment distance function. Although it is rather simple, we would contain some details here to simplify the following definition of the completeness field.

$$d(X, \overline{P_j^i P_{j+1}^i}) = \begin{cases} \|\overrightarrow{XP_j^i}\| & \alpha_j \leq 0 \\ \|\overrightarrow{XX_j^i}\| & 0 < \alpha_j < 1 \\ \|\overrightarrow{XP_{j+1}^i}\| & \alpha_j \geq 1 \end{cases} \tag{12}$$

where

$$X_j^i = P_j^i + \alpha_j^i(X)\overrightarrow{P_j^i P_{j+1}^i} \tag{13}$$

$$\alpha_j^i(X) = \frac{\overrightarrow{P_j^i X} \cdot \overrightarrow{P_j^i P_{j+1}^i}}{\|\overrightarrow{P_j^i P_{j+1}^i}\|^2} \tag{14}$$

Here $\alpha_j^i(X)$ is the relative location of the projection of X on $\overrightarrow{P_j^i P_{j+1}^i}$.

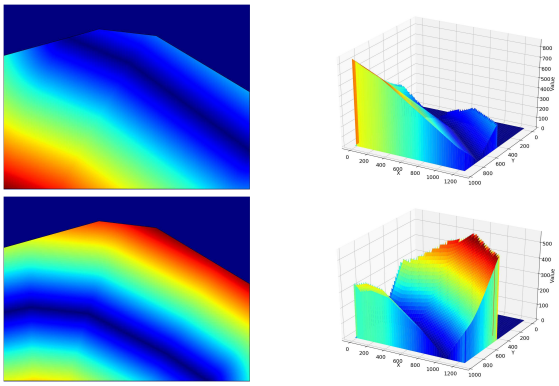

Figure 5: 2D and 3D visualizations of proximity fields for route 8 and 10 in Figure 4

Completeness Field For each point X in the view space, we would also like to estimate its relative location within a route $\mathbf{R}_i$ in the case it belongs to this route. This allows us to evaluate the completeness of a track to see where it starts and ends regarding $\mathbf{R}_i$. Like the proximity field, this is defined as the 2D completeness field $\mathbf{F}_{c,\mathbf{R}_i}$, as illustrated in Figure 6.

$$\mathbf{F}_{c,\mathbf{R}_i}(X) = \frac{\alpha_{j^*}^i(X)\|\overrightarrow{P_{j^*}^i P_{j^*+1}^i}\| + \sum_{j=1}^{j^*-1}\|\overrightarrow{P_j^i P_{j+1}^i}\|}{\sum_{j=1}^{n-1}\|\overrightarrow{P_j^i P_{j+1}^i}\|} \tag{15}$$

where j^* indicates the nearest segment.

$$j^* = \underset{j=1,2,\ldots,n-1}{\arg\min}\ d(X, \overline{P_j^i P_{j+1}^i}) \tag{16}$$

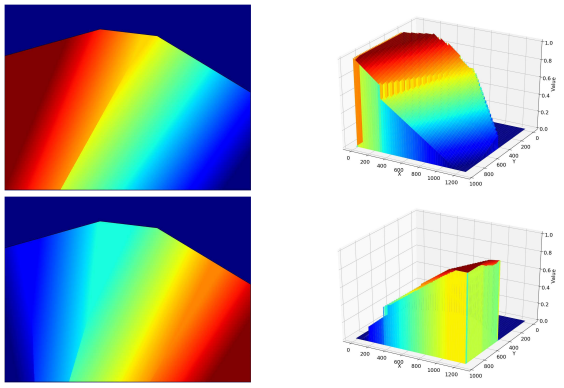

Figure 6: 2D and 3D visualizations of completeness fields for route 8 and 10 in Figure 4

3.2.2 Route Identification

Given a vehicle tracklet $\mathbf{T}_x$ and a set of routes $\{\mathbf{R}_i \mid i = 1, 2, \ldots, m\}$, our task is to determine whether $\mathbf{T}_x$ belongs to one of the routes and if so, find out which one it is. Since we are building this system with no annotation of vehicle tracklets for training, a set of metrics are proposed to solve this as a multi-class classification problem.

Proximity Metric An obvious and reasonable metric is the average distance from the track to each route. However, the absolute distance in the image varies according to the object scale. Therefore, a scale-normalized distance is calculated with the scale factors and the proximity field.

$$d(\mathbf{T}_{x,j}, \mathbf{R}_i) = \frac{\mathbf{F}_{p,\mathbf{R}_i}(\mathbf{P}_{x,j})}{\mathbf{S}_{x,j}} \tag{17}$$

For the normalization purpose, the final proximity metric is defined as

$$M_p(\mathbf{T}_x, \mathbf{R}_i) = \sigma(a - b\frac{1}{n}\sum_{j=1}^{n} d(\mathbf{T}_{x,j}, \mathbf{R}_i)) \tag{18}$$

where σ is the Sigmoid function and a, b are additional parameters.

Completeness Metric It is clearly not enough to only rely on the proximity metric, as it cannot tell the direction a vehicle is moving towards. In fact, a stable vehicle at any point within a route would yield a small distance. Therefore, another important metric is about how the vehicle goes through the route step by step. With the completeness field $\mathbf{F}_{c,\mathbf{R}_i}$, we can track the progress of the vehicle along a specific route, as shown in Figure 7. A linear model is used to approximate the change of the completeness as

$$\mathbf{F}_{c,\mathbf{R}_i}(\mathbf{P}_{x,j}) = c_{x,i}\frac{j}{n} + d_{x,i} \tag{19}$$

The slope $c_{x,i}$ is solved with least squares. As its ideal value should be 1, the completeness metric is defined as

$$M_c(\mathbf{T}_x, \mathbf{R}_i) = \min(c_{x,i}, \frac{1}{c_{x,i}}) \tag{20}$$

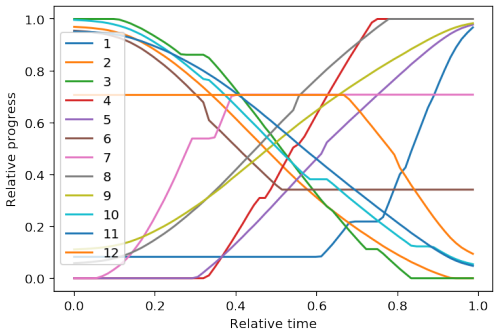

Figure 7: Completeness graph for a vehicle going along track 9 in Figure 4

Stability Metric The stability metric is designed to estimate if the vehicle is going along the route at a constant distance from it. It uses another linear model to approximate the change of the scale-normalized distance.

$$d(\mathbf{T}_{x,j}, \mathbf{R}_i) = e_{x,i}\frac{j}{n} + f_{x,i} \tag{21}$$

The slope $e_{x,i}$ is also solved with least squares. In an ideal track where the vehicle moves in a fixed lane along the route, the distance should remain unchanged, yielding a zero slope. On the other hand, a non-zero value indicates the vehicle is approaching or leaving the route. The stability metric is defined similarly to a normal distribution as

$$M_s(\mathbf{T}_x, \mathbf{R}_i) = \exp(-\frac{1}{2}e_{x,i}^2) \tag{22}$$

Aggregation and Classification With all three metrics defined above, the final confidence score of a track belonging to a route can be simply aggregated as

$$\begin{aligned} S(\mathbf{T}_x, \mathbf{R}_i) = &\min(1, \max(0, w_p M_p(\mathbf{T}_x, \mathbf{R}_i))) \\ &+ \min(1, \max(0, w_c M_c(\mathbf{T}_x, \mathbf{R}_i))) \\ &+ \min(1, \max(0, w_s M_s(\mathbf{T}_x, \mathbf{R}_i))) \end{aligned} \tag{23}$$

If $\max_i S(\mathbf{T}_x, \mathbf{R}_i)$ is above a threshold S_{min}, the prediction for the track is

$$C(\mathbf{T}_x, \mathbf{R}) = \arg\max_i S(\mathbf{T}_x, \mathbf{R}_i) \tag{24}$$

Otherwise, it will be discarded.

4. Experiments

4.1. System Implementation

We adopted the Mask R-CNN model from Facebook Detectron2 [34] with pre-trained weights on Microsoft COCO [19]. Due to efficiency concerns, we followed the practice in [37] to build a pipeline system for detection, tracking, and classification, as shown in Figure 8. It utilizes frame-level parallelism and out-of-order-execution mechanisms for the bottle-neck detection stage. According to our test, each Nvidia Geforce RTX 2080Ti GPU can process 9 frames each second. This design makes our system easily scalable, with support for up to 8 GPUs. We use a common code base with Track 3 [26] and it is publicly available[2] .

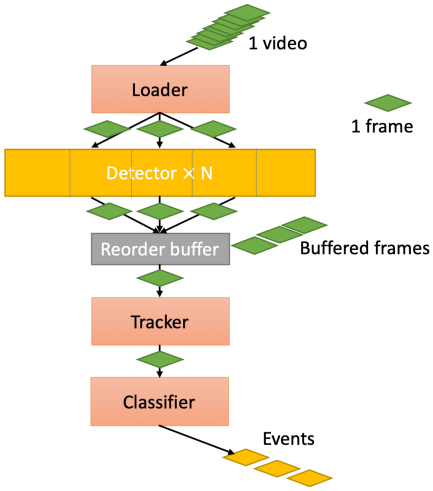

Figure 8: System pipeline

4.2. Dataset and Settings

We evaluated our proposed system on Track 1 of the 2020 AI City Challenge. The dataset split A contains videos of 5 hours from 20 unique cameras in different light and

[2] https://github.com/Lijun-Yu/AICity2020-track1

weather conditions. Since the dataset only provides diagrams of the routes with simple descriptions, we manually labeled the polygon for each route. Apart from the routes and region of interest in each camera view, no additional information is used.

For the parameters in our system, the values listed in Table 1 are used for a video with a frame rate of z fps.

Table 1: Parameter Values

Name	Value	Name	Value
IoU_{nms}	0.8	f_{truck}	0.8
σ_g	$0.3z$	w_v	z
v_{min}	10	l_{min}	$0.3z$
a	4	b	5
w_p	1	w_c	1.25
w_s	1	S_{min}	0.3

4.3. Official Metrics

As no additional data is available or allowed for self-testing, we will only provide the experiment results on the official metrics from the leaderboard. The official metrics combine efficiency score and effectiveness score as

$$S_1 = \alpha S_{1,efficiency} + \beta S_{1,effectiveness} \tag{25}$$

where $\alpha = 0.3, \beta = 0.7$. The efficiency score is based on the total execution time T as

$$S_{1,efficiency} = \max(0, 1 - \frac{T \times base\_factor}{5 \times video\_time}) \tag{26}$$

The effectiveness score is computed as a weighted average of normalized weighted root mean square error $nwRMSE$ across all videos, movements, and vehicle classes, with proportional weights based on the number of vehicles of the given class in the movement.

$$wRMSE = \sqrt{\sum_{i=1}^{k} \frac{i}{\sum_{j=1}^{k} j} (\hat{x}_i - x_i)^2} \tag{27}$$

Before the leaderboard finalizes, it was using 50% of the test to evaluate. The snapshot of the leaderboard at that time is shown in Table 2, where our system ranked the top one.

However, after the finalization, our performance dropped to the 3rd place, with a score of 0.9292. As it is unlikely that our model with very few parameters would overfit the dataset, a potential explanation of this drop is the un-balanced split of the test data.

4.4. Qualitative Results

In Figure 9, we provide qualitative visualizations of each camera under different conditions. Video demonstrations of all cameras are available online[3]. As we can see from the figures, vehicles are typically assigned to the correct route. However, there are still failures caused by certain reasons:

1. A vehicle gets occluded by other vehicles or surroundings, such as trees.
2. The image is blurred due to weather conditions, such as rainy or snowy days.
3. Mis-classification of vehicle types between car and truck. This is currently believed to be the key drawback as we are using inconsistent definitions of trucks.

Table 2: Leaderboard snapshot before finalization (using 50% test data)

Rank	TeamID	Score
1	**Ours**	**0.9444**
2	99	0.9415
3	110	0.9381

Table 3: Finalized leaderboard (using all test data)

Rank	TeamID	Score
1	99	0.9389
2	110	0.9346
3	**Ours**	**0.9292**
4	26	0.8936
5	22	0.8852
6	74	0.8829
7	6	0.8540
8	119	0.8254
9	80	0.8064
10	65	0.7933

4.5. Ablation Study

We explore the effectiveness of the three metrics we proposed for route identification. Due to lack of labeled data, the experiments are performed on a 60-second clip, which has the only ground truth in the dataset. As shown in Table 4, each of the three metrics contribute to the final performance.

5. Conclusion

In this paper, we proposed a zero-short vehicle route understanding system, Zero-VIRUS. It has been proved to successfully identify vehicle routes in the vehicle counting task via a novel 2D field modeling. The performance of the proposed approach shows its effectiveness and robustness in

[3] https://drive.google.com/drive/folders/1s3TPykPa3JTaPOHUVOQF8S4iUi3SduAN?usp=sharing

Figure 9: Visualization of results on sample frames from different scenes and different conditions, where colored dots indicate identified routes

Table 4: Effectiveness of metrics on a 60-second Clip

Metrics	Effectiveness Score
M_p	0.8903
M_p, M_c	0.9455
M_p, M_c, M_s	**0.9554**

identifying the vehicle route in variant camera views. The proposed system has achieved the third place on Dataset A in stage 1 of the competition (Track 1: Vehicle Counts by Class at Multiple Intersections). Future works are suggested on improving the efficiency of the proposed solution as performing real-time and also learning to build better models for the route that can be adaptive to new scenes and new vehicle actions.

6. Acknowledgement

This project is funded in part by Carnegie Mellon University's Mobility21 National University Transportation Center, which is sponsored by the US Department of Transportation. This research is supported in part by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/Interior Business Center (DOI/IBC) contract number D17PC00340. This research is supported in part through the financial assistance award 60NANB17D156 from U.S. Department of Commerce, National Institute of Standards and Technology.

References

[1] Xiaojun Chang, Wenhe Liu, Po-Yao Huang, Changlin Li, Fengda Zhu, Mingfei Han, Mingjie Li, Mengyuan Ma, Siyi Hu, Guoliang Kang, et al. Mmvg-inf-etrol trecvid 2019: Activities in extended video. 2

[2] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Eco: Efficient convolution operators for tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6638–6646, 2017. 2

[3] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. Learning spatially regularized correlation filters for visual tracking. In *Proceedings of the IEEE international conference on computer vision*, pages 4310–4318, 2015. 2

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 2

[5] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: A benchmark. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 304–311. IEEE, 2009. 1

[6] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2009. 3

[7] Qianyu Feng, Guoliang Kang, Hehe Fan, and Yi Yang. Attract or distract: Exploit the margin of open set. In *The IEEE International Conference on Computer Vision*, October 2019. 3

[8] Qianyu Feng, Yu Wu, Hehe Fan, Chenggang Yan, Mingliang Xu, and Yi Yang. Cascaded revision network for novel object captioning. *IEEE Transactions on Circuits and Systems for Video Technology*, 2020. 3

[9] Qianyu Feng, Zongxin Yang, Peike Li, Yunchao Wei, and Yi Yang. Dual embedding learning for video instance segmentation. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019. 2

[10] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2012. 2

[11] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 2, 3

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 3

[13] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *Proceedings of the IEEE european Conference Computer Vision*, 2016. 2

[14] Boris S Kerner, Hubert Rehborn, Mario Aleksic, and Andreas Haug. Recognition and tracking of spatial–temporal congested traffic patterns on freeways. *Transportation Research Part C: Emerging Technologies*, 12(5):369–400, 2004. 2

[15] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot visual object categorization. *IEEE transactions on pattern analysis and machine intelligence*, 36(3):453–465, 2013. 3

[16] Feng Li, Cheng Tian, Wangmeng Zuo, Lei Zhang, and Ming-Hsuan Yang. Learning spatial-temporal regularized correlation filters for visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4904–4913, 2018. 2

[17] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 2, 3

[18] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 2

[19] 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 *Proceedings of the IEEE european conference on computer vision*, pages 740–755. Springer, 2014. 2, 3, 6

[20] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Proceedings of the IEEE european conference on computer vision*, pages 21–37. Springer, 2016. 2

[21] Wenhe Liu, Guoliang Kang, Po-Yao Huang, Xiaojun Chang, Lijun Yu, Yijun Qian, Junwei Liang, Liangke Gui, Jing Wen, Peng Chen, and Alexander G. Hauptmann. Argus: Efficient activity detection system for extended video analysis. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision Workshops*, pages 126–133, 2020. 2

[22] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016. 2

[23] Hyeonseob Nam, Mooyeol Baek, and Bohyung Han. Modeling and propagating cnns in a tree structure for visual tracking. *arXiv preprint arXiv:1608.07242*, 2016. 2

[24] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4293–4302, 2016. 2

[25] Clement Chun Cheong Pang, William Wai Leung Lam, and Nelson Hon Ching Yung. A method for vehicle count in the presence of multiple-vehicle occlusions in traffic images. *IEEE Transactions on Intelligent Transportation Systems*, 8(3):441–459, 2007. 2

[26] Yijun Qian, Lijun Yu, Wenhe Liu, and Alexander G. Hauptmann. Electricity: An efficient multi-camera vehicle tracking system for intelligent city. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2020. 6

[27] Yijun Qian, Lijun Yu, Wenhe Liu, Guoliang Kang, and Alexander G Hauptmann. Adaptive feature aggregation for video object detection. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision Workshops*, pages 143–147, 2020. 2

[28] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 2

[29] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 2

[30] Mikel Rodriguez, Ivan Laptev, Josef Sivic, and Jean-Yves Audibert. Density-aware person detection and tracking in crowds. In *International Conference on Computer Vision*, pages 2423–2430. IEEE, 2011. 1

[31] Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pages 935–943, 2013. 3

[32] Georgios Vigos, Markos Papageorgiou, and Yibing Wang. Real-time estimation of vehicle-count within signalized links. *Transportation Research Part C: Emerging Technologies*, 16(1):18–35, 2008. 2

[33] Zhongdao Wang, Liang Zheng, Yixuan Liu, and Shengjin Wang. Towards real-time multi-object tracking. *arXiv preprint arXiv:1909.12605*, 2019. 2, 3

[34] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 6

[35] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2411–2418, 2013. 2

[36] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 3

[37] Lijun Yu, Peng Chen, Wenhe Liu, Guoliang Kang, and Alexander G Hauptmann. Training-free monocular 3d event detection system for traffic surveillance. In *IEEE International Conference on Big Data*, pages 3838–3843. IEEE, 2019. 6

[38] Lijun Yu, Dawei Zhang, Xiangqun Chen, and Alexander Hauptmann. Traffic danger recognition with surveillance cameras without training data. In *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE, 2018. 2

[39] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *Proceedings of the IEEE international conference on computer vision*, pages 1116–1124, 2015. 1