Vehicle re-identification and trajectory reconstruction using multiple moving cameras in the CARLA driving simulator

SUBMISSION DATE / POSTED DATE

07-11-2022 / 10-11-2022

CITATION

Kumar, Ashutosh; Kashiyama, Takehiro; Maeda, Hiroya; Zhang, Fan; Omata, Hiroshi; Sekimoto, Yoshihide (2022): Vehicle re-identification and trajectory reconstruction using multiple moving cameras in the CARLA driving simulator. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.21513768.v1

DOI

10.36227/techrxiv.21513768.v1

Vehicle re-identification and trajectory reconstruction using multiple moving cameras in the CARLA driving simulator

Ashutosh Kumar
Institute of Industrial Science
The University of Tokyo, Japan
ashutosh@iis.u-tokyo.ac.jp

Takehiro Kashiyama
Faculty of Economics
Osaka University of Economics, Japan
t.kashiyama@osaka-ue.ac.jp

Hiroya Maeda
UrbanX Technologies, Inc.
Tokyo, Japan
hiroya_maeda@urbanx-tech.com

Fan Zhang
Department of Civil and Environmental Engineering
The Hong Kong University of Science and Technology, Hong Kong
cefzhang@ust.hk

Hiroshi Omata
Center for Spatial Information Science
The University of Tokyo, Japan
homata@iis.u-tokyo.ac.jp

Yoshihide Sekimoto
Center for Spatial Information Science
The University of Tokyo, Japan
sekimoto@csis.u-tokyo.ac.jp

***Abstract*—Analyzing vehicle movement trajectories is essential for understanding urban mobility and traffic flow patterns. Obtaining a reliable estimate of vehicle trajectory is challenging as it requires the vehicle to be observed and re-identified at different locations and times. Recently, a scalable citywide traffic flow estimation method has been proposed utilizing moving cameras on vehicle dashboards. As moving cameras constantly interact with their surroundings, they provide valuable information about the urban environment that can be used to estimate vehicle trajectories. This study extends the recently proposed method for traffic flow estimation by using cameras mounted on multiple moving observers to reconstruct the trajectory of detected vehicles. We develop the CARLA ReID dataset, which includes more than 50,000 images taken from 85 cameras for over 700 different vehicle models, and train a re-identification network to identify the same vehicle by multiple observers. Utilizing our proposed methodology, we conduct extensive research to estimate trajectories of vehicles in a driving simulator CARLA and evaluate the accuracy of reconstructed trajectories using Symmetrized Segment-Path Distance (SSPD) and *Hausdorff* Distance metrics. Our proposed method achieves a mean error of 5.13 meters evaluated using the SSPD metric for ten driving experiments in CARLA. Findings from this study will provide valuable insights for conducting traffic flow research in a simulation environment, which is otherwise challenging and costly in practice.**

***Index Terms*—CARLA, ITS, OSNet, ReID, Vehicle trajectory**

I. INTRODUCTION

Reconstruction of vehicle trajectory is crucial to understanding traffic mobility patterns as they contain rich spatial and temporal information about the traffic flow. When the same vehicle is observed at different places and times, it is possible to reconstruct its rough trajectory. Since there can be several vehicles in the traffic stream, the same vehicle needs to be identified by multiple observers. Camera sensors such as surveillance cameras observe traffic flow at different locations and can be used to re-identify vehicles for trajectory reconstruction research and applications.

Various research has been conducted to re-identify vehicles and reconstruct their trajectories. The study conducted in [1] estimates vehicle trajectories at signalized intersections under the Connected and Automated Vehicle (CAVs) environment. Research conducted in [2] developed a framework *VeTrac* that uses surveillance cameras to trace vehicle movements and reconstruct the trajectory. Further, they use graph convolution to maintain identity consistency across multiple cameras and a self-training process to align the trajectories on the road network. Another research [3] introduced a two-step process based on wavelet analysis for filtering errors and reconstructing trajectories.

The use of static cameras in previous studies has limited the ability to observe vehicles beyond the location where the cameras were installed. Many previous research studies lack generalizability and scalability because they have a number of parameters. Additionally, it may not be possible for everyone to reconstruct vehicle trajectories through large-scale surveillance cameras due to data procurement issues.

In recent studies [4], [5], the authors propose a novel framework using vehicle detection, vehicle tracking, vehicle localization, and map-matching on the OpenStreetMap road network for estimating traffic flow parameters from moving cameras mounted on the dashboard of a car. The authors suggest using crowd-sourcing for citywide traffic flow estimation, making the method highly scalable. Another study [6]

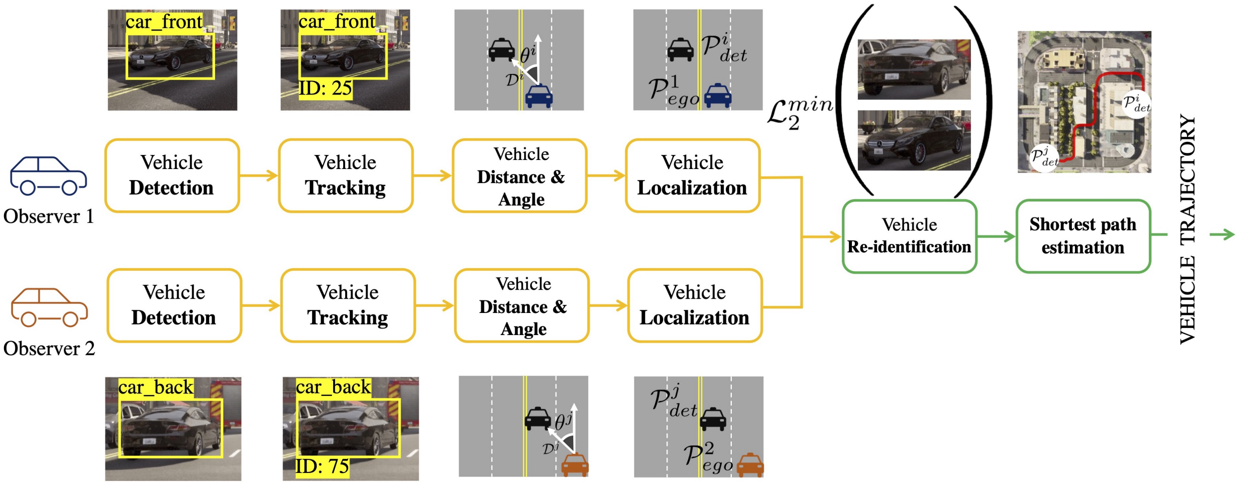


Fig. 1. The framework for the reconstruction of trajectory using two observers. The vehicles are detected/tracked and their distance and angle from the ego/observer vehicle is calculated to localize (e.g., $\mathcal{P}^i_{det}$) them on the map. A vehicle re-identification neural network finds the closest matching vehicles using $\mathcal{L}_2$ loss on the feature vectors. Re-identified detected vehicles' position is used to reconstruct the trajectory using shortest path algorithm.

extends the methodology introduced in [4] to estimate traffic flow in a driving simulator CARLA to verify the efficacy of the proposed algorithm [4] under a variety of environmental conditions.

Conducting driving experiments in the real world for trajectory reconstruction research using computer vision techniques is extremely challenging since we need to set up cameras at different locations and also have multiple target vehicles for generalization. Alternatively, a driving simulator such as CARLA provides a realistic virtual environment to conduct various experiments. Similar to the research conducted in [6] for traffic flow estimation using multiple moving cameras in a driving simulator, we use the CARLA driving simulator to develop a fully computer vision-based approach to reconstruct vehicle trajectory, as shown in Fig. 1. Specifically, we:

- Develop a large-scale vehicle re-identification (ReID) dataset in the CARLA driving simulator with more than 50,000 images to train a ReID model and validate its performance on synthetic and real-world [7] data sets.
- Use YOLOv7 [8] object detection network with SORT [9] to estimate distances and angles of the detected vehicles for localization and map-matching.
- Re-identify the same vehicles from multiple moving observers using the vehicle re-identification network with OSNet [10] backbone.
- Estimate the trajectory of the re-identified vehicles using the shortest path algorithm utilizing distance heuristic and assess the similarity between the estimated trajectory and ground truth using two metrics – Symmetrized Segment-Path Distance and *Hausdorff* distance.

II. METHODOLOGY

In the following subsections, we describe the necessity and details of the CARLA driving simulator, development of the CARLA ReID dataset for vehicle re-identification, vehicle detection and tracking, vehicle localization, vehicle re-identification, trajectory estimation techniques, evaluation metrics, and driving experiment details.

A. *CARLA driving simulator*

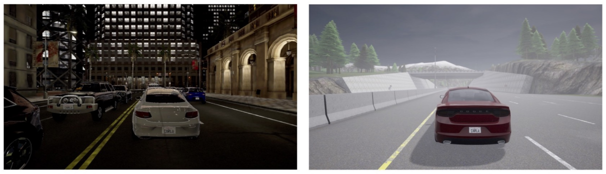
Conducting driving experiments in the real world for autonomous driving research and applications is challenging and expensive. A driving simulator simulates a variety of environments and generates dynamic camera images at any time and location. In this research, we use the CARLA driving simulator (0.9.13) [11], which has a client-server-based framework to simulate driving experiments for various autonomous driving research and applications. CARLA's simulation engine runs on Unreal Engine 4 to create highly realistic, physics-based environments. CARLA offers various environments known as *Town* for conducting experiments, such as urban towns, rural areas, highways, etc. It also provides various vehicle class models along with several customization options for the simulation environment. Lights, weather, traffic, camera, etc., inside the simulation environment can be controlled through the client using Python API, as shown in Fig 2.

B. *CARLA ReID dataset*

Conducting driving experiments in the real world for vehicle re-identification data collection is challenging, expensive, and time-consuming. For preparing a vehicle re-identification dataset, every unique vehicle needs to be identified and labeled manually from multiple camera sources. Research conducted in [7] captured vehicle images from up to 20 surveillance cameras in an unconstrained traffic scenario to develop the VeRi dataset. Another study [12] developed a large-scale vehicle re-identification dataset from synthetic vehicle models prepared in Unity called the VehicleX dataset. Though it reduces the manual labeling of the images, nevertheless, they have only vehicle images from varying perspectives without any context, like a traffic stream on the road.

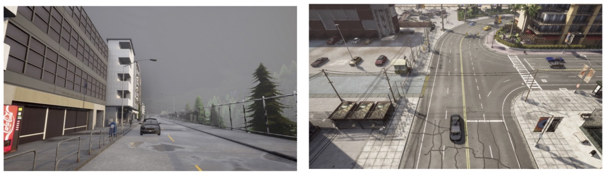
Town01 Town10

a) Multiple simulation environments



Night time Haze

b) Dynamic light and weather conditions



Front view Top view

c) Customizable camera location

Fig. 2. CARLA (0.9.13) has nine different simulation environments. The light and weather can be customized or kept dynamic with adjustable rate. The camera attached to the ego vehicle can also be moved anywhere in the simulation world for different applications.

In this study, we leverage the CARLA driving simulator and prepare a large-scale synthetic vehicle re-identification dataset called the CARLA ReID dataset, consisting of four vehicle classes such as cars, trucks, motorcycles, and bicycles. CARLA offers 39 models for all classes of vehicles, such as Audi A2, Mini Cooper, Tesla Model 3, Tesla Cybertruck, etc. As mentioned previously, the camera in the CARLA driving simulator can be customized and placed at any location in the simulation world. We consider 85 fixed locations of cameras to observe the same vehicles. The 85 locations of cameras are chosen manually at various distances around the vehicle. For each vehicle, 85 images are captured and stored with labels such as vehicle type, vehicle ID, and camera ID.

New models of vehicles can be prepared and imported, however, CARLA has the option to change the vehicle color for 35 out of 39 vehicle models, which can be used to create new vehicles. In such cases, the vehicle model remains the same; however, a new unique vehicle (vehicle ID) is created. We consider various colors, such as black, light-gray, silver, red, green, etc., to create additional vehicle IDs for the compatible 35 vehicle models, as shown in Fig. 3.

55,196 images are generated from 700 different vehicle models, i.e., 700 distinct vehicle IDs, with 650 vehicles containing 50,949 images for training, 50 vehicles containing 424 images for query, and 3,823 images for gallery. A vehicle re-identification model tries to match each vehicle ID in the query set to the corresponding vehicle ID from a different camera in the gallery set, as explained later in Section II-E.

C. *Vehicle detection and tracking*

1) Vehicle detection using YOLOv7: For the detection of vehicles inside the CARLA driving simulator, we use the vehicle orientation dataset [4] to train a YOLOv7 model [8] to detect both vehicle class and orientation. The main reason to choose YOLOv7 to train the vehicle detection neural network is that YOLOv7 is significantly lightweight (75% reduction in parameters with 1.5% higher AP for the same base model) compared to its real predecessor YOLOv4 [13], [14] while achieving improved benchmark results on the COCO dataset [15].

We use the base model of YOLOv7 with pre-trained weights on the COCO dataset with an input size of 640×640. We first train with the real-world images from the vehicle orientation dataset [4] for 100 epochs with a learning rate of 0.001 on four Tesla A100 GPUs [16] and then use the synthetic vehicle orientation dataset [6] to fine-tune the next ten epochs with a reduced learning rate of 0.0001 to prevent large changes in the parameters. It should be noted that the vehicle orientation dataset contains 15 classes of vehicles, while the synthetic vehicle orientation dataset has 12 classes of vehicles due to the absence of bus class; thus, bus_front, bus_back, and bus_side classes are not present. Thus, we keep the output size of 15 for fine-tuning weights. It is also important to note here that during training, we do not consider image augmentation techniques such as image flipping in any direction because such augmentation will produce images with incorrect labels. For all subsequent experiments, we use the weight that achieves the best mAP on the validation set of the synthetic vehicle orientation dataset.

2) Vehicle tracking using SORT: When a vehicle is detected in a frame, it needs to be identified by assigning a unique ID and tracked over consecutive frames, as object detection models such as YOLOv7 detect vehicles independently across consecutive frames. Identifying identical vehicles in the image frames is necessary to consider a detected vehicle only once for re-identification/matching. If vehicles are not tracked, multiple images of the same vehicle from consecutive frames will exist in the search space for vehicle re-identification.

In this research, we consider an online lightweight tracker SORT for tracking vehicles. SORT tracks objects by linear velocity model using Kalman filter and association of detections to tracks using Hungarian algorithm considering Intersection over Union (IoU) metric. There exists an improved version of SORT with a CNN-based feature descriptor known as DeepSORT [17], which is good at tracking lost objects after their reappearance in the image. In our problem, no lost objects need to be tracked; thus, a simple Kalman filter-based tracking algorithm is sufficient.

As tracker state of the original SORT algorithm considers only single object class, similar to the research conducted in [4], we modify the state of the tracker to include the detected vehicle class (η), as shown in Eq. 1.

$$\Psi = [\epsilon, a, r, \dot{\epsilon}, \dot{a}, \eta]^T \tag{1}$$

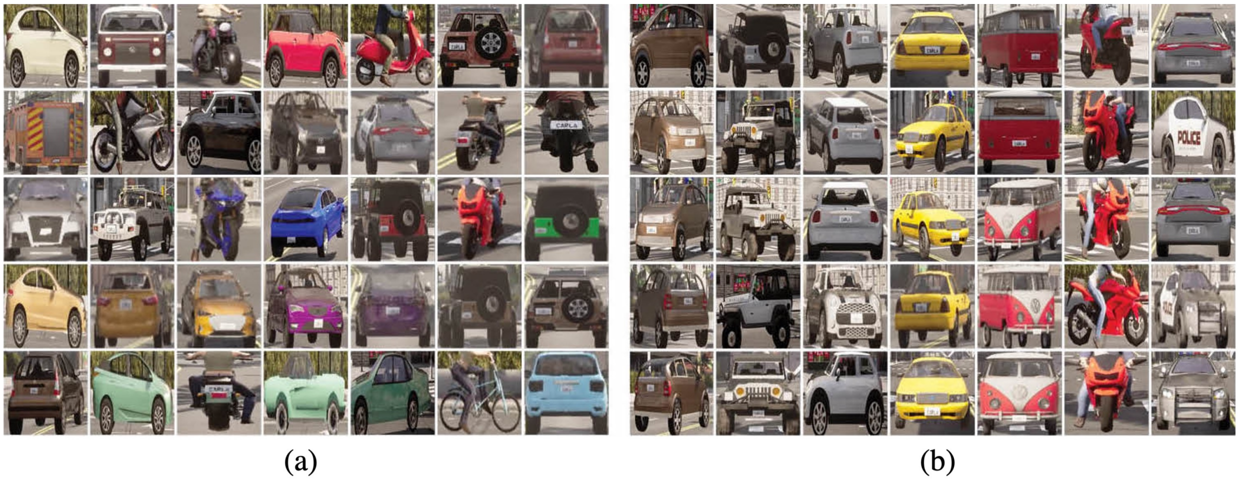

(a) (b)

Fig. 3. Sample images from the CARLA ReID dataset, resized to the same dimension for visualization. Fig. **(a)** shows the variation of vehicle models and colors in the CARLA ReID dataset. CARLA ReID dataset contains four classes of vehicles such as car, truck, motorcycle, and bicycle. Fig. **(b)** shows the camera angle variation for observing the same vehicle. Each vehicle is observed by 85 cameras located at varying distances surrounding the target vehicle.

In Eq. 1, ϵ, a, r, and η represent center of the bounding box coordinates, area of the bounding box, aspect ratio (assumed constant), and vehicle class, respectively. The change in the center of the bounding box coordinates and its area is represented using $\dot{\epsilon}$ and $\dot{a}$, respectively.

D. *Localization of detected vehicles*

To localize the detected vehicles on the map, we need to know their position, which can be estimated using the ego vehicle's position and the distance and angle of the detected vehicles with respect to the ego vehicle.

1) Distance of the detected vehicles: To estimate the distance of the detected vehicles from the ego vehicle, we use the photogrammetry approach presented in [18], which states that the ratio of object size in the real-world ($H_{real,m}$) and its size in image coordinates ($H_{image,m}$), considering same units (say meters), is equal to the distance of the object from the camera sensor ($\mathcal{D}$) and focal length (f), as shown in Eq. 2. In Eq. 2, we consider the height dimension of the object/image/camera sensor to estimate the distance of the object from the camera sensor.

$$\frac{H_{real,m}}{H_{image,m}} = \frac{\mathcal{D}}{f} \tag{2}$$

The dimension of the object in image reference frame by digital cameras is measured in terms of pixels $H_{image,px}$, which is related to the metric unit $H_{image,m}$ using sensor size (μ_h) and image resolution height (I_H) as $H_{image,m} = \frac{\mu_h \times H_{image,px}}{I_H}$. Replacing $H_{image,m}$ in Eq. 2, we can obtain the distance of the object in meters, as shown in Eq. 3.

$$\mathcal{D} = \frac{H_{real,m} \times f \times I_H}{\mu_h \times H_{image,px}} \tag{3}$$

In Eq. 3, $H_{real,m}$ is considered fixed for different classes of vehicles. For cars, trucks, bicycles/motorcycles, the average height is considered to be 1.6 meters, 3.5 meters, and 1.45 meters, respectively. $H_{image,px}$ is known through the height of the bounding box coordinates of the detected vehicles from the object detection network YOLOv7. I_H is also a constant height of the image resolution. The value of real-world camera's focal length (f) and sensor size (μ_h) is known through sensor specifications. However, in CARLA, these values are unknown, and thus we obtain the focal length by considering its relationship with the camera field of view (FOV) and image resolution width (I_W), as shown in Equation 4.

$$f = \frac{I_W}{2 \times \tan(FOV \times \frac{\pi}{360})} \tag{4}$$

In the CARLA driving simulator, we consider an FOV of 90 degrees and an image resolution width of 1,280 pixels for carrying out experiments, which is used to calculate f in Eq. 4 as 640-pixel units. The only parameter unknown in Eq. 3 is the sensor height μ_h of the camera, which is estimated considering it as the only variable in Eq. 3 using 51 known ground truth distances of different vehicles. The average value of μ_h from 51 observations is calculated to be 599.46-pixel units [6], which we use in Eq. 3 to calculate the distance of detected vehicles.

2) Angle of detected vehicles: To estimate the angle (θ) that the detected vehicles make with respect to the ego vehicle, we make use of the camera matrix (K), which is used to project 3D coordinates in the real world onto the image plane. Conversely, the inverse of the camera matrix (K^{-1}) can be used to project image coordinates to 3D rays in the real world. While the real-world cameras need to be calibrated to obtain the camera matrix, the camera sensor in the CARLA driving simulator is considered ideal and distortion-free and hence can be obtained directly using focal length (f_x, f_y) and optical center ($I_W/2$, $I_H/2$) as shown in Eq. 5.

$$K = \begin{bmatrix} f_x & 0 & I_W/2 \\ 0 & f_y & I_H/2 \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

Considering the inverse of the camera matrix, we obtain one ray passing through the image center ($I_W/2$, $I_H/2$), which represents a ray along the forward direction of the ego vehicle and the other ray through the centroid of the detected vehicles (c_x, c_y), as shown in Eq. 6 and Eq. 7, respectively.

$$\overrightarrow{r_I} = K^{-1} \cdot [\frac{I_W}{2} \quad \frac{I_H}{2} \quad 1]^T \tag{6}$$

$$\overrightarrow{r_c} = K^{-1} \cdot [c_x \quad c_y \quad 1]^T \tag{7}$$

The angle θ between the ego vehicle's direction and detected vehicle is obtained using the cosine law for $\overrightarrow{r_I}$ and $\overrightarrow{r_c}$, as shown in Eq. 8.

$$\theta = \arccos \frac{\overrightarrow{r_I} \cdot \overrightarrow{r_c}}{\|\overrightarrow{r_I}\| \|\overrightarrow{r_c}\|} \tag{8}$$

3) Vehicle localization and map-matching: Once the distances and angles of the detected vehicles with respect to the ego vehicle are estimated, their positions can be determined using the ego vehicle's location. In CARLA, a coordinate (x, y, z) is represented using the Cartesian coordinate system, as shown in Fig. 4. If the position of the ego vehicle is (x_{ego}, y_{ego}, z_{ego}), the position of the detected vehicle (x^i_{det}, y^i_{det}, z^i_{det}) at a distance $\mathcal{D}^i$ and angle θ^i, considering positive direction of axes shown in Fig. 4 can be calculated, as shown in Eq. 9.

$$\begin{aligned} x^i_{det} &= x_{ego} + \mathcal{D}^i \cos\theta^i \\ y^i_{det} &= y_{ego} + \mathcal{D}^i \sin\theta^i \\ z^i_{det} &= z_{ego} \end{aligned} \tag{9}$$

In Eq. 9, we consider the z-coordinate of the detected and ego vehicle as the same since the road surface is almost flat within the camera's FOV.

Once the detected vehicle's position is estimated, the position can be map-matched on the road network to obtain a *waypoint* ($\mathcal{W}$). In CARLA, a waypoint $\mathcal{W}$ consists of four parameters, which are road ID, section ID, lane ID, and distance (s) from the beginning of the road section, as shown in Fig. 4. It should be noted that in CARLA, the sign of lane ID for one direction is opposite to another direction. As shown in Fig. 4, the lane IDs on the side of the ego vehicle are positive compared to the opposite direction lanes. It is important to know the waypoint information during the reconstruction of the trajectory, as explained in Section II-F.

E. Vehicle re-identification

To identify the same vehicle across several vehicles detected by multiple observers, it is necessary to carry out vehicle re-identification. Given a query vehicle and a gallery of several vehicles, the vehicle re-identification problem refers to identifying the query vehicle from the vehicle gallery.

The backbone of the vehicle re-identification networks commonly use CNNs such as ResNet [19]. However, such CNNs are generic and developed for object recognition tasks. Recently, specific networks for re-identification problems have been developed. OSNet [10] is an Omni-scale feature learning network specifically designed for the re-identification of objects such as persons, vehicles, etc. Re-identification networks with a CNN backbone can be trained using the softmax loss function or a ReID specific loss function – triplet loss [20]. During our preliminary experiment on the VeRi dataset, however, we find the performance of softmax to be better than the triplet loss, and thus we only consider softmax loss for training ReID networks for further experiments.

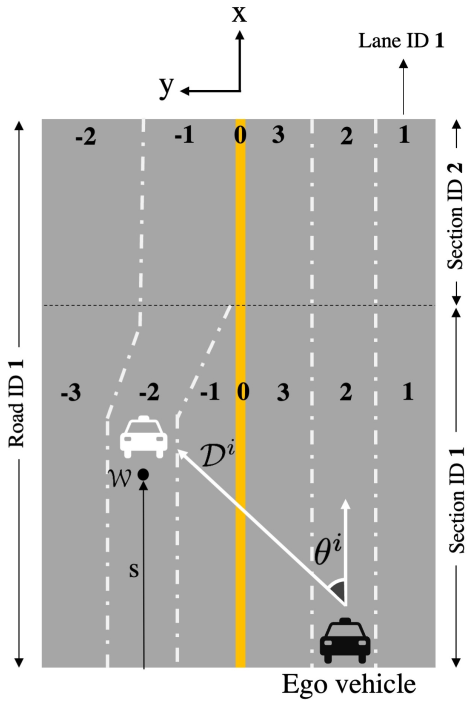


Fig. 4. The waypoint $\mathcal{W}$ of the estimated position (x^i_{det}, y^i_{det}, z^i_{det}) of the detected vehicle is obtained from a hash combination of road ID, section ID, lane ID, and distance s from the beginning of the road section.

We use the VeRi dataset and the CARLA ReID dataset to train vehicle re-identification network using six backbones – ResNet-50, ResNet-152, InceptionV4, OSNet_ain_x1, OSNet_ibn_x1, and OSNet_x1 with softmax loss function for 60 epochs. The hyperparameters of the model are shown in Table I. Finally, we select the model with the best Cumulative Matching Characteristics (CMC) curve top-1 accuracy, which is defined as shown in Eq. 10. The ranking in the CMC curve is estimated using the $\mathcal{L}_2$ loss between the query vehicle image and vehicles in gallery images.

$$Acc^k = \begin{cases} 1, & \text{query in the top-k ranked gallery vehicles} \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

Further, for the model trained on the CARLA ReID dataset, we also evaluate its accuracy on the real-world images of the validation set of the VeRi dataset to understand the performance of the model trained on synthetic images on real-world images.

During the driving experiment in the CARLA driving simulator, we re-identify the vehicle only when the distance

TABLE I
HYPERPARAMETERS OF THE REID MODEL WITH DIFFERENT BACKBONES.

Hyperparameter	Value
Learning rate	0.0003
Input size	256×256
Loss function	Softmax
Optimizer	Adam [21]
Output feature vector	512

between the ego vehicle and the detected vehicle is less than 15 meters. This is done to improve the accuracy of the re-identification since the vehicle size would be larger compared to if it was re-identified at a farther distance. This consideration improves not only the re-identification but also the estimated position of the vehicles as they are much closer to the ego vehicle [6]. Further, it also reduces the tracking failure reducing the redundant images in the search space during re-identification.

F. Trajectory reconstruction

Once a target vehicle is re-identified at different locations by the observer/ego vehicles, its rough trajectory can be estimated considering its obtained positions, as mentioned in Section II-D. Apart from the position of detected vehicles, we also store attributes such as the timestamp of detection/re-identification, vehicle orientation, and position of the ego vehicle. If there are n observed locations for a target vehicle, we estimate its complete trajectory by $n-1$ pairs of trajectory based on the timestamp of detection/re-identification. For estimating the trajectory between two positions, we consider the shortest path between those two locations using the A* search algorithm with distance as a heuristic using NetworkX [22]. *calc_traj* function in Algorithm 1 shows feature extraction of detected vehicles for re-identification and trajectory reconstruction.

Further, it is important to note here that the estimated position using Eq. 9 may sometimes contain some errors. For example, if for one of the estimated positions, the detected vehicle is on the opposite lane of the ego vehicle and due to errors, say in the distance, the estimated position lies in the same direction lanes as the ego vehicle, the estimated path between the two positions can be quite large. As shown in Fig. 4, lanes in opposite directions have different signs, which we use to solve the occurrences of estimated position on the wrong lane, as shown in Algorithm 1. For example, if the orientation of the detected vehicle is *front*, it should lie on the opposite lane of the ego vehicle. However, if the lane ID is the same as the ego vehicle, we find a new waypoint by multiplying lane ID with -1 without changing the road ID, section ID, or distance from the beginning of the road. This creates a new waypoint in the opposite direction of the ego vehicle, from which we extract the position for trajectory calculation. A similar procedure is repeated for vehicles with *back* orientation such that the direction of the ego vehicle and the detected vehicle is the same, as shown in the *correct_position* function of Algorithm 1.

Algorithm 1: Re-identification & trajectory estimation

```
Function correct_position($veh$, $ego$):
        // Corrects detected vehicle direction
    if $veh.road \cap ego.road$ then
        if ($veh.orient \cap$ 'front') $\cap$ ($veh.lane \oplus ego.lane > 0$) then
            // Opposite direction to ego vehicle
            $veh$.lane $\leftarrow$ $veh$.lane $\times$ -1
        end if
        if ($veh.orient \cap$ 'back') $\cap$ ($veh.lane \oplus ego.lane < 0$) then
            // Same direction as ego vehicle
            $veh$.lane $\leftarrow$ $veh$.lane $\times$ -1
        end if
    end if
    return ($veh.x$, $veh.y$)
End Function
Function calc_traj($ego1$, $ego2$, $veh\_t1$, [] $veh2$):
    $veh1\_t1\_f$ $\leftarrow$ extract_feature($veh\_t1.image$)
    // Extract feature for target vehicle
    [ ] $veh2\_f$ $\leftarrow$ extract_feature($veh2.images$)
        // Search space for re-identification
    $veh\_t1$, $veh\_t2$ $\leftarrow$ $\mathcal{L}_2^{min}$(veh_t1_f, [ ] veh2_f)
        // Re-identify the vehicle at time t2
    $veh\_t1\_pos$ $\leftarrow$ $correct\_position$($veh\_t1$, $ego1$)
    $veh\_t2\_pos$ $\leftarrow$ $correct\_position$($veh\_t2$, $ego2$)
        // Correct direction for front, back
    $\mathcal{P}$ $\leftarrow$ shortest_path($veh\_t1\_pos$, $veh\_t2\_pos$)
    // $\mathcal{P}$ $\rightarrow$ Corrected trajectory
End Function
```

The switching of the lane by multiplication with -1 only works when the road ID is the same, as shown in Fig. 4. This is due to the OpenDrive map [23] standard that comprises the CARLA road network, which does not guarantee that the consecutive road sections follow the same signs for lane ID. Thus if the right side of a road section has a positive lane ID, the next road section will not have the same case and is arbitrary.

For the evaluation of the reconstructed trajectory compared to the ground truth path, we consider the following two metrics:

Symmetrized Segment-Path Distance (SSPD): SSPD [24] is a shape-based distance, which considers the entire trajectory and is less affected by noise, and it is calculated using Segment-Path-Distance (SPD). Consider $\mathcal{D}_{xt}$ as the distance from a point x to a trajectory $\mathcal{T}$ defined as the minimum of distances between this point (x) and all segments $\mathcal{S}$ that compose $\mathcal{T}$. The SPD between two trajectories $\mathcal{T}^1$ and $\mathcal{T}^2$ is the mean of all distances from points that compose $\mathcal{T}^1$ to the trajectory $\mathcal{T}^2$.

$$\mathcal{D}_{SPD}(\mathcal{T}^1, \mathcal{T}^2) = \frac{1}{n_1} \sum_{i_1=1}^{n_1} \mathcal{D}_{xt}(x_{i1}^1, \mathcal{T}^2) \qquad (11)$$

where $\mathcal{D}_{xt}(x_{i1}^1, \mathcal{T}^2) = min_{i2\in[0,..,n_2-1]} \mathcal{D}_{xs}(x_{i1}^1, s_{i2}^2)$ The

SPD, however, is not symmetric, which means that $\mathcal{D}_{SPD}(\mathcal{T}^1,\mathcal{T}^2)=0$ if $\mathcal{T}^1$ is a very small trajectory of $\mathcal{T}^2$. However, in this case $\mathcal{D}_{SPD}(\mathcal{T}^2,\mathcal{T}^1)$ will be a very large number. The SSPD is then defined using SPD as shown in Eq. 12.

$$\mathcal{D}_{SSPD}(\mathcal{T}^1,\mathcal{T}^2)=\frac{\mathcal{D}_{SPD}(\mathcal{T}^1,\mathcal{T}^2)+\mathcal{D}_{SPD}(\mathcal{T}^2,\mathcal{T}^1)}{2} \tag{12}$$

Distance between points can be considered using both the Spherical or Euclidean systems. In this study, we consider the Euclidean system to calculate distances between points for calculating the SSPD metric.

Hausdorff Distance: *Hausdorff* Distance (HD) [25], [26] measures the dissimilarity between two point sets and is used primarily for pattern matching and evaluating the quality of clusters. The directed *Hausdorff* distance $\hat{\mathcal{H}}$ between two trajectories $\mathcal{T}^1$ and $\mathcal{T}^2$ is defined as the maximum of distances between each point $x\in\mathcal{T}^1$ to its nearest neighbor $y\in\mathcal{T}^2$ as shown in Eq. 13.

$$\hat{\mathcal{H}}(\mathcal{T}^1,\mathcal{T}^2)\ = max_{x\in\mathcal{T}^1}[min_{y\in\mathcal{T}^2}(\|x,y\|)] \tag{13}$$

Similar to the SSPD metric, for estimating the distance between points $\|x,y\|$, we consider the Euclidean distance function. The directed *Hausdorff* distance is not symmetric as $\hat{\mathcal{H}}(\mathcal{T}^1,\mathcal{T}^2)\neq\hat{\mathcal{H}}(\mathcal{T}^2,\mathcal{T}^1)$. The *Hausdorff* Distance (HD) $\mathcal{H}$ is defined as the maximum of the directed *Hausdorff* distance in both directions, and hence is symmetric, as shown in Eq. 14.

$$\mathcal{H}(\mathcal{T}^1,\mathcal{T}^2)\ = max[\hat{\mathcal{H}}(\mathcal{T}^1,\mathcal{T}^2),\hat{\mathcal{H}}(\mathcal{T}^2,\mathcal{T}^1)] \tag{14}$$

The main difference between SSPD and HD is the way distance is calculated between the two trajectories. If there is a point on one of the two trajectories, HD measures the longest distance, which one must travel from the point to the other trajectory, while SSPD measures the mean distance. For this reason, the HD metric is always greater than SSPD.

G. *Driving experiment details*

As part of the driving experiment, we drive through two different virtual cities – Town01 and Town10 – of the CARLA driving simulator during daytime normal weather conditions. Town01 has a basic town layout with several T junctions, while Town10 is an urban town with the most realistic textures among all towns, with a wide variety of surrounding terrain. Approximately, Town01 has a size of 387×329 m^2 with 160 road sections, and Town10 has a size of 217×204 m^2 with 200 road sections. We conduct five driving experiments in Town01 and the rest five in Town10. The simulation of each driving experiment is finished when the target vehicle whose trajectory is to be reconstructed reaches its final destination.

For simplification of results for visualization, we consider that the traffic is composed of three types of vehicles – observers, target vehicle, and self-driving agent vehicles. For experiments, we consider up to four observers, one target vehicle and up to 20 self driving agents in the simulation environment. The self-driving agents are kept to 20 to avoid traffic congestion or accidents due to unpredictable behavior of the agents due to dynamic conditions. The purpose of self-driving agents is to provide sample search space for target vehicle re-identification. All observer vehicles are equipped with a camera sensor, and they detect vehicles, track vehicles, estimate the distance and angle of the detected vehicles to localize them, and put the detected vehicle's image to a database for re-identification by another program. The driver of the target vehicle is instructed to go from an initial location to a final location, during which it will be detected by observer vehicles. As the target vehicle passes nearby observer vehicles, it is detected/tracked, and the localized position of both the observer vehicle and target vehicle, vehicle class, and timestamp are sent to a database. At the end of the experiment, we read attributes of the detected/tracked vehicles and re-identify the same vehicles at different timestamps by multiple observers, and reconstruct the trajectory, as described in Section II-E and Section II-F.

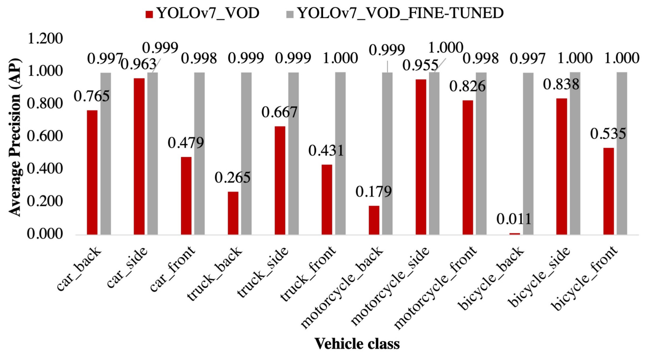


Fig. 5. Figure shows the comparison of APs for vehicles present in the validation set of the synthetic vehicle orientation dataset. YOLOv7_VOD refers to the model solely using real-world images in the vehicle orientation dataset for 47 epochs, while YOLOv7_VOD_FINE-TUNED refers to the model fine-tuned with images in the synthetic vehicle orientation dataset.

III. RESULTS

A. *Vehicle detection*

We evaluate the performance of YOLOv7 for vehicle detection on the synthetic vehicle images prepared in the CARLA driving simulator. As mentioned before in Section II-C1, we train YOLOv7 on the vehicle orientation dataset for 100 epochs and select the weights with the highest mAP on the validation set of the vehicle orientation dataset, which corresponds to epoch 47 and continue to fine-tune with the synthetic vehicle orientation dataset with reduced learning rate. The AP of the vehicle detection before and after fine-tuning is shown in Fig. 5. From Fig. 5, we notice that the performance of the YOLOv7 model increases significantly after fine-tuning the weights trained on the vehicle orientation dataset with the images from the synthetic vehicle orientation dataset.

B. *Vehicle re-identification*

We train the vehicle re-identification models with six CNN backbones – ResNet-50, ResNet-152, InceptionV4, OSNet_ain_x1, OSNet_ibn_x1, and OSNet_x1 – with softmax loss function for 60 epochs using the VeRi dataset and the

TABLE II
TOP-1 ACCURACY AND ITS CORRESPONDING MAP OF THE MODELS TRAINED ON THE CARLA REID DATASET.

Backbone	Top-1 accuracy	mAP
ResNet-50	99.50%	98.70%
ResNet-152	99.50%	99.00%
InceptionV4	100.00%	97.10%
OSNet_ibn_x1	100.00%	98.30%
OSNet_ain_x1	100.00%	99.30%
OSNet_x1	100.00%	99.30%

TABLE III
UNMATCHED AND TOP-1 ACCURACY OF THE MODELS ON THE VERI VALIDATION DATASET CONTAINING 1,678 QUERY IMAGES.

Backbone	VeRi dataset		CARLA ReID dataset	
	Unmatched	Top-1	Unmatched	Top-1
ResNet-50	55	96.72%	568	66.15%
ResNet-152	55	96.72%	563	66.45%
InceptionV4	100	94.04%	574	65.79%
OSNet_ibn_x1	6	**99.64%**	171	89.81%
OSNet_ain_x1	9	99.46%	169	89.93%
OSNet_x1	11	99.34%	164	**90.23%**

CARLA ReID dataset. For the models trained on the CARLA ReID dataset, we evaluate the top-1 accuracy on its validation set, as shown in Table II. The validation set of the CARLA ReID dataset contains 424 query images and 3,823 gallery images. From Table II, we find that all the six models have a very high top-1 accuracy. ReID-specific OSNet models achieve perfect accuracy (100%) in the vehicle matching on the query images of the CARLA ReID dataset.

To further assess the performance of the CARLA ReID dataset on the re-identification of vehicles in the real world, we also calculate top-1 accuracy on the validation set of the VeRi dataset. The validation set of the VeRi dataset contains 1,678 vehicles in query images and 11,579 vehicles in gallery images. Table III shows the top-1 accuracy of models trained on the VeRi dataset and the CARLA ReID dataset on the validation set of the VeRi dataset.

From Table III, it is evident that models with generic CNN backbones, such as ResNet-50, ResNet-152, and InceptionV4, trained on the CARLA ReID dataset have a drastic reduction in the matching accuracy for real-world images in the VeRi dataset. OSNet models trained on the CARLA ReID dataset significantly outperform generic CNN backbone-based models and have comparable performance with the models trained on the VeRi dataset. Comparing results from Table II and Table III, we select OSNet_x1 model trained on the CARLA ReID dataset in our experiment for the reconstruction of vehicle trajectory.

C. *Vehicle trajectories*

We present the results of trajectory reconstruction for various trips in Fig. 6. In Fig. 6, we overlay the ground truth trajectory (shown in red) and estimated trajectory (shown in yellow) on the top view of each town. Fig. 6(a) – (e) and Fig. 6(f) – (j) show the experiment results for Town10 and Town01, respectively. The number of observers, trip distance and the accuracy of trajectory reconstruction using SSPD and HD metrics are also presented.

From Fig. 6, we notice that the vehicle trajectory can be reconstructed with comparable accuracy for a range of vehicle movements. For simple movements with small route choices, such as Fig. 6(a) and Fig. 6(f), the accuracy is high. In some cases, such as Fig. 6(c), Fig. 6(d), Fig. 6(e), and Fig. 6(j), we find that some parts of the estimated trajectory do not correspond or overlap with the ground truth. Overall, we notice that the SSPD or HD metric are smaller for reconstructed trajectory in Town01 compared to Town10 for similar movement patterns.

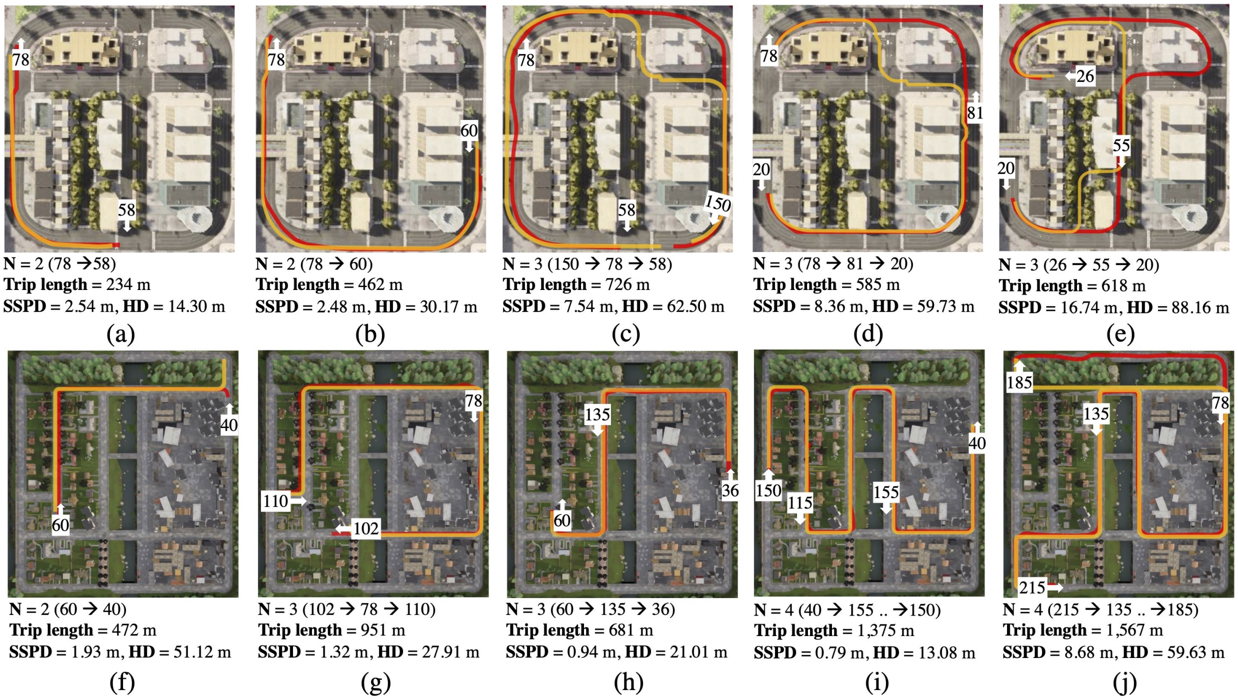
IV. DISCUSSIONS

Vehicle detection using YOLOv7 on the validation set of the synthetic vehicle orientation dataset shows good accuracy using real-world images. After fine-tuning, the accuracy is significantly improved and reaches almost perfect accuracy. Compared to the vehicle detection AP on the validation set of the synthetic vehicle orientation dataset evaluated using the YOLOv4 model in [6], YOLOv7 significantly outperforms its predecessor. The high accuracy can be attributed to the limited number of vehicle models present in the synthetic vehicle orientation dataset, which accounts for the similar distribution of the training and validation sets.

The accuracy of various backbones for vehicle re-identification clearly shows that OSNet models outperform generic CNN backbones even with deeper layers such as ResNet-152. This is not so pronounced on the validation set of the CARLA ReID dataset since generic CNN models such as InceptionV4, ResNet-152, etc., achieve similar accuracy. However, the evaluation of the models trained on the CARLA ReID dataset on the VeRi dataset shows the superiority of OSNet. Particularly, OSNet_x1 achieves the best performance among other variants of OSNet models. Despite the CARLA ReID dataset containing only synthetic images, it can well match real-world vehicles with more than 90% top-1 accuracy.

We consider vehicles for re-identification only when they are less than 15 meters from the observer vehicle. This is due to two reasons. The first reason is that YOLOv7 can detect vehicles when they are quite far, e.g., more than 50 meters. Such far detected vehicles are very small in size (width and height are in the range of 10 pixels to 50 pixels), and hence it leads to incorrect re-identification due to less information since the input size of images to the OSNet re-identification model is 256×256. By contrast, a vehicle at distances smaller than 15 meters has sizes ranging from 70 pixels to 500 pixels. The second reason is due to the error in larger estimated distances as presented in [6], which shows the RMSE for estimated distances up to 60 meters to be 3.95 meters, and it reduces to 2.33 meters for distances smaller than 20 meters. A larger error in the distance may localize the vehicle on the wrong side of the road, thereby increasing errors in trajectory reconstruction.

Results of trajectory reconstruction show that the estimated trajectory overlaps well with the ground truth trajectory. In the case of Fig. 6(c), Fig. 6(d), and Fig. 6(e), the estimated



N = 2 (78 → 58)
Trip length = 234 m
SSPD = 2.54 m, **HD** = 14.30 m
(a)

N = 2 (78 → 60)
Trip length = 462 m
SSPD = 2.48 m, **HD** = 30.17 m
(b)

N = 3 (150 → 78 → 58)
Trip length = 726 m
SSPD = 7.54 m, **HD** = 62.50 m
(c)

N = 3 (78 → 81 → 20)
Trip length = 585 m
SSPD = 8.36 m, **HD** = 59.73 m
(d)

N = 3 (26 → 55 → 20)
Trip length = 618 m
SSPD = 16.74 m, **HD** = 88.16 m
(e)

N = 2 (60 → 40)
Trip length = 472 m
SSPD = 1.93 m, **HD** = 51.12 m
(f)

N = 3 (102 → 78 → 110)
Trip length = 951 m
SSPD = 1.32 m, **HD** = 27.91 m
(g)

N = 3 (60 → 135 → 36)
Trip length = 681 m
SSPD = 0.94 m, **HD** = 21.01 m
(h)

N = 4 (40 → 155 .. → 150)
Trip length = 1,375 m
SSPD = 0.79 m, **HD** = 13.08 m
(i)

N = 4 (215 → 135 .. → 185)
Trip length = 1,567 m
SSPD = 8.68 m, **HD** = 59.63 m
(j)

Fig. 6. Comparison of ground truth path and estimated trajectory of vehicles. The ground truth path is shown by the red line, while the yellow line shows the estimated path. The observer vehicle's location and ID are also shown with the arrow pointing towards the *front* direction of the vehicle.

trajectories do not completely overlap due to the incorrect perception of the human driver about the shortest path for traversing a set of positions. Moreover, we observe a slight curve in the ground truth movement trajectory between two points caused by the bad steering skills of the human driving inside the simulator. In contrast, the agent makes a more informed decision to move, which results in a mostly straight trajectory. The curved movement is more prominent in Town10 (Fig. 6(a) to Fig. 6(e)) because Town10 has two-lane roads compared to one lane in Town01, which gives more degree of freedom for the driver, causing slightly curved trajectories.

The error in estimated distances can lead to the wrong localization of re-identified vehicles since we assume a constant height for different classes of vehicles in Eq. 3. In some of the vehicle models, the vehicle characteristics are much farther than their respective vehicle classes, e.g., the Mercedes Sprinter is much taller than the average height of cars, resulting in distance/localization errors. As mentioned in Section II-F, this can be solved for *front* and *back* orientation vehicles by finding a new waypoint in the opposite lane for the same road ID; the same is not true for vehicles with *side* orientation since the road ID is mostly not the same. This can be seen in the reconstructed trajectory in Fig. 6(j), where the orientation of the target vehicle near observer 185 is *side* and has a different road ID, which causes the final localized position to lie on the same side as the observer vehicle, and hence the estimated trajectory using shortest path algorithm takes the first left turn after observer 78. Such problems can be solved if the information about the surrounding road section from the road network is also taken into account.

Table IV shows the sources of various errors and their impact on the reconstructed trajectory. In Table IV, Type 1 error, which has a low impact on accuracy, refers to the errors due to humans, while Type 2 error refers to potential errors that occur in the trajectory reconstruction framework. Among Type 2 errors, vehicle detection and tracking failure have almost no impact because even if the vehicles are failed to be detected/tracked among some frames, they can be detected later as they approach closer to the observer vehicle. Even for distances smaller than 15 meters, tracking failure only increases the search space for vehicle re-identification without affecting the accuracy as their localized position is almost the same. Localization error's impact is considered medium when the detected vehicle orientation is either *front* or *back*, which can be fixed by switching lane direction if the road section is the same. However, for *side* orientation, we need additional information about the road section, which is not considered in this study. Traffic density also has a low impact since vehicle detection is carried out for each vehicle when they are close to the observer vehicles. The traffic density does increase the search space for re-identification; however, we do not notice significant errors since OSNet can re-identify target vehicles from thousands of vehicles in gallery images with high accuracy. The impact of traffic density may increase if there are various identical vehicles in the vicinity of the observer vehicle. There is a small impact in error due the number of observers if the detected positions are close as they do not have several route choices; however, the impact will

TABLE IV
ERROR TYPES WITH THE SEVERITY LEVEL.

Error Type	Description	Impact
Type 1	Shortest-path perception	Medium
Type 1	Driving style	Low
Type 2	Vehicle detection/tracking failure	Low
Type 2	Localization error	Medium to high
Type 2	Traffic density	Low
Type 2	Number of observers	Low to high

increase with larger distances with multiple route choices.

V. CONCLUSIONS

In this study, we present a novel method for estimating vehicle trajectory from multiple cameras mounted on moving vehicles. We train a YOLOv7 model to detect vehicles and track them using the SORT algorithm. The distances and angles of detected vehicles are estimated for localization and map-matching, and they are re-identified at different locations using a ReID model with OSNet backbone. We use the re-identified vehicle's detected positions to estimate the trajectory using the shortest path algorithm and verify the accuracy of the reconstructed trajectory using SSPD and HD metrics for various driving experiments. The CARLA ReID dataset, developed in this study, accurately identifies vehicles in real-world images in the VeRi dataset, with 90.23% top-1 accuracy.

This study does not consider trajectory reconstruction in a traffic stream consisting of vehicles of the same model and color with high density. In the continuation of our research, we will conduct experiments to develop algorithms for traffic streams consisting of high-density identical vehicle models and also work on developing and optimizing a real-time framework for communication between different ego vehicles for re-identification and trajectory reconstruction. As moving cameras become more prevalent in vehicles and crowdsourcing becomes more accessible, we anticipate our developed techniques to have a promising future for vehicle trajectory reconstruction.

ACKNOWLEDGMENT

This study used high-performance Tesla A100 GPU cluster nodes provided by the Information Technology Center, The University of Tokyo. This work has also been supported by JST SPRING, Grant Number JPMJSP2108.

REFERENCES

[1] X. Chen, J. Yin, K. Tang, Y. Tian, and J. Sun, "Vehicle trajectory reconstruction at signalized intersections under connected and automated vehicle environment," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[2] P. Tong, M. Li, M. Li, J. Huang, and X. Hua, "Large-scale vehicle trajectory reconstruction with camera sensing network," in *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, 2021, pp. 188–200.

[3] M. R. Fard, A. S. Mohaymany, and M. Shahri, "A new methodology for vehicle trajectory reconstruction based on wavelet analysis," *Transportation Research Part C: Emerging Technologies*, vol. 74, pp. 150–167, 2017.

[4] A. Kumar, T. Kashiyama, H. Maeda, and Y. Sekimoto, "Citywide reconstruction of cross-sectional traffic flow from moving camera videos," in *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021, pp. 1670–1678.

[5] A. Kumar, T. Kashiyama, H. Maeda, H. Omata, and Y. Sekimoto, "Real-time citywide reconstruction of traffic flow from moving cameras on lightweight edge devices," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 192, pp. 115–129, 2022.

[6] A. Kumar, T. Kashiyama, H. Maeda, H. Omata, and Y. Sekimoto, "Citywide reconstruction of traffic flow using the vehicle-mounted moving camera in the carla driving simulator," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2022, pp. 2292–2299.

[7] X. Liu, W. Liu, H. Ma, and H. Fu, "Large-scale vehicle re-identification in urban surveillance videos," in *2016 IEEE international conference on multimedia and expo (ICME)*. IEEE, 2016, pp. 1–6.

[8] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696*, 2022.

[9] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE international conference on image processing (ICIP)*. IEEE, 2016, pp. 3464–3468.

[10] K. Zhou, Y. Yang, A. Cavallaro, and T. Xiang, "Omni-scale feature learning for person re-identification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3702–3712.

[11] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[12] Y. Yao, L. Zheng, X. Yang, M. Naphade, and T. Gedeon, "Simulating content consistent vehicle datasets with attribute descent," in *ECCV*, 2020.

[13] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[14] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Scaled-yolov4: Scaling cross stage partial network," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 13 024–13 033.

[15] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[16] T. Suzumura, A. Sugiki, H. Takizawa, A. Imakura, H. Nakamura, K. Taura, T. Kudoh, T. Hanawa, Y. Sekiya, H. Kobayashi *et al.*, "mdx: A cloud platform for supporting data science and cross-disciplinary research collaborations," *arXiv preprint arXiv:2203.14188*, 2022.

[17] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE international conference on image processing (ICIP)*. IEEE, 2017, pp. 3645–3649.

[18] D. Kendal, "Measuring distances using digital cameras," *Australian Senior Mathematics Journal*, vol. 21, no. 2, pp. 24–28, 2007.

[19] 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.

[20] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International workshop on similarity-based pattern recognition*. Springer, 2015, pp. 84–92.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[22] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using networkx," Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2008.

[23] M. Dupuis, M. Strobl, and H. Grezlikowski, "Opendrive 2010 and beyond–status and future of the de facto standard for the description of road networks," in *Proc. of the Driving Simulation Conference Europe*, 2010, pp. 231–242.

[24] P. Besse, B. Guillouet, J.-M. Loubes, and R. François, "Review and perspective for distance based trajectory clustering," *arXiv preprint arXiv:1508.04904*, 2015.

[25] F. Hausdorff, *Grundzüge der mengenlehre*. von Veit, 1914, vol. 7.

[26] A. A. Taha and A. Hanbury, "An efficient algorithm for calculating the exact hausdorff distance," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 11, pp. 2153–2163, 2015.