Scalable and Real-time Multi-Camera Vehicle Detection, Re-Identification, and Tracking

Pirazh Khorramshahi[1], Vineet Shenoy[1], Rama Chellappa[1]

[1]Artificial Intelligence for Engineering and Medicine Lab, Johns Hopkins University, Baltimore, MD

{pkhorra1, vshenoy4, rchella4}@jhu.edu

***Abstract*—Multi-camera vehicle tracking is one of the most complicated tasks in Computer Vision as it involves distinct tasks including Vehicle Detection, Tracking, and Re-identification. Despite the challenges, multi-camera vehicle tracking has immense potential in transportation applications in deciphering the value of traffic sensors data and providing time-critical information. Several recent works have addressed this problem. However, most of the effort has gone towards improving accuracy on benchmark datasets while disregarding the overwhelming amount of computational power and time needed to carry out this task on its edge and thus making it prohibitive for large-scale and real-time deployment. Therefore, in this work we shed light on practical issues that should be addressed for the design of a multi-camera tracking system to provide actionable and timely insights. Moreover, we propose a real-time city-scale multi-camera vehicle tracking system that compares favorably to computationally intensive alternatives in terms of performance. To show its effectiveness, in addition to integration into the Regional Integrated Transportation Information (RITIS) system[1], we participated in the 2021 NVIDIA AI City multi-camera tracking challenge and our method is ranked among the top five performers.**

***Index Terms*—Vehicle Detection, Re-Identification, Single Camera Tracking, Multi Camera Tracking, Real-Time, Scalable, Domain Adaptation.**

I. Introduction

Multi-Camera Tracking (MCT) is the task of tracking an unknown number of objects across a number of mounted cameras that may or may not have overlap in their fields-of-view. This makes MCT to be one of the most complicated tasks in Computer Vision as it involves several fundamental vision tasks, namely Object Detection, Tracking, and Re-identification. Due to recent advancements in Computer Vision, thanks to Deep Learning and Deep Convolutional Neural Networks (DCNN) in particular, interests in high performance MCT systems have been rapidly growing. The underlying reason for this rapid growth is that MCT has great applications in intelligent transportation systems.

In terms of transportation, there are great opportunities for optimizing traffic flow, reducing air pollution and carbon footprints by minimizing the travel duration, enhancing the safety of vehicle, bicycle and pedestrian traffic, and ultimately reducing the costs incurred by departments of transportation (DoT). According to the World Health Organization (WHO), about 1.3 million people die on the worlds' roads and 20-50 millions are injured every year [1]. These numbers can be significantly reduced upon providing real-time alerts and actionable insights from traffic cameras. In addition, based on a report from Washington State DoT [2], the cost of a traffic signal is between $\$250,000 - \$500,000$ without the consideration of routine maintenance. The cost increases in case of establishing coordination with other traffic signals which involves human operators in the loop. However, such costs can be greatly reduced by adopting a robust and real-time multi-camera tracking system that can process many cameras installed at different traffic intersections without any explicit coordination. Also in terms of surveillance applications, a MCT system can locate unauthorized vehicles and identify dangerous maneuvers and behaviours which contribute to increased safety and security.

In this paper, we focus on developing an MCT system tailored for vehicles that can operate in real-time. As mentioned above, the MCT system involves three distinct vision tasks. 1- **Vehicle Detection:** Detection is responsible for localizing vehicles of various types at different locations and scales within the camera view. High quality detections are critical to the success of the MCT system as it impacts the performance of all the downstream modules. 2- **Single Camera Vehicle Tracking**: Upon receiving detections, a multi-object tracker attempts to associate the detected bounding boxes belonging to individual identities simultaneously and predict their future locations while being robust to occlusion and variations in velocity of vehicles. Note that there are methods to perform multi-object detection and tracking via a single model [3], [4]. However in our work we find that having separate modules for the two tasks helps us to identify potential issues and optimize each to the MCT task. 3- **Vehicle Re-identification:** Re-identification aims to obtain discriminative appearance embeddings from the tracked vehicles in each camera so that we can associate different single camera tracks corresponding to individual identities. The extracted visual embeddings should be robust to variations in orientation and lighting conditions that may be different from camera to camera. Once single camera tracks and their corresponding representations have been computed for all the cameras, a clustering algorithm is needed to associate single camera tracks to unique identities based on the computed visual features and spatio-temporal information that comes naturally with each single camera track. Since the number of true identities is not known beforehand,

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

the clustering algorithm should be independent of the number of vehicle identities to perform this task.

There have been several works that address MCT for vehicles [5]–[8]. However, all these works only attempt to maximize the multi-camera tracking accuracy on benchmark datasets without any consideration for the inference time and computational complexity. As a result, they require significant amount of computation time and resources to process a number of videos with the duration of only few minutes. In contrast, in this work, we present an MCT system that can run in real-time and provide timely results for any downstream goals. To demonstrate the effectiveness of our method, we integrate our MCT system, as a prototype, in the RITIS system which is a data-driven platform from the University of Maryland for transportation analysis, monitoring, and data visualization. RITIS has access to the real-time traffic camera feeds that are provided by state and local departments of transportation traffic centers to the University of Maryland via electronic data feeds from traffic management centers. Our MCT system provides real-time multi-camera capability which is useful for a variety of transportation applications. In addition, we evaluated our system on the 2021 NVIDIA AI City Multi-Camera Vehicle Tracking challenge and were ranked among the top five competitors.

The rest of the paper is organized as follows. In section II, we review recent works on object detection, multi-object tracking, and vehicle re-identification. The detailed architecture of our multi-camera tracking pipeline is discussed in section III. Next, in section IV, we discuss the implementation details, validation data and its statistics, and evaluate our approach on real-time traffic data as well the Multi-Camera Vehicle Tracking challenge of the 2021 AI City challenge to demonstrate its effectiveness and validate our design choices. Finally we conclude in section V.

II. Related Work

Here we review most relevant works on vehicle detection, tracking, and re-identification as these are the pillars of the multi-camera vehicle tracking task.

Object Detection: Yolo [9] and it variants [10]–[12], Single Shot [13], RetinaNet [14], Faster R-CNN [15] and Mask R-CNN [16] are popular choices of object detectors to be employed in variety of applications. Many previous works use these models as off-the-shelf detectors trained on the large-scale object detection COCO dataset [17]. Moreover, RetinaNet, Faster R-CNN and Mask R-CNN are particularly popular as they are regularly maintained by Detectron library [18] which significantly facilitates their adoption by researchers. More recently, EfficientDet [19] that has a weighted bi-directional feature pyramid network to allow for easy and fast multi-scale feature fusion has been developed. Many works fine-tune these detectors on external vehicle data such as the UA-DETRAC [20] dataset. One extension of this work is SpotNet [21] which uses attention mechanisms to locate roads and driving surfaces, and limiting detections to these surfaces. Similarly, the authors in [22] propose FG-BR Net which focuses on objects in the foreground. In the first of this two-stage method, high quality regions of interest (RoI) proposal are fed to the detector by suppressing background features while amplifying feature activations in foreground objects. The second stage, assuming that there are errors in the first stage, refines the first stage proposals with pairwise non-weighted local background fusions. Some methods from face detection have found their way into vehicle detection. The authors in [23] first use a region proposal network to extract RoI proposals, which is followed by an adaptively-generated Gaussian kernel which extracts local features. The output of this stage is fed to an LSTM [24] module to encode global context, and subsequently to a classifier and bounding box regression module. Concepts tested on pedestrian-detection datasets have also found their way into vehicle detections, with the authors in [25] proposing a convolutional neural network that predicts centers and scales of bounding boxes. This method obviates the need for anchor boxes and avoids computationally-heavy post processing steps usually involved with key-point pairing-based detectors.

Multi-Target Single Camera Tracking: Most tracking methods first detect and then associate objects. The local methods [26]–[28] consider only two frames at a time, but do not perform well when occlusion, pose variations, and camera motion are present. In contrast *global* methods consider multiple frames concurrently and solve network flow problems [29]–[31]. One of the more recent trackers is the Deep Affinity Model [32], which performs detection and data association concurrently. As input, two frames of a video (not necessarily consecutive) are fed through two networks with shared parameters and object features are extracted.Another recent local method is CenterTrack [3], which represents all objects as a point at the center of a bounding box. To associate detections across frames, the distance between the object center in the previous frame and the predicted offset in the current frame is calculated. A new object identifier is created when there is no previous object center within a certain radius of the current object center. Similar to CenterTrack, [4] associates detections across frames through an offset under the assumption that motion between frames is small; the authors regress the bounding box location from the previous frame to the current frame using the regression head of a Faster R-CNN detector [15] and the deep features in the current frame. If the predicted bounding box after regression has a high Intersection-over-Union (IoU) with an incoming bounding box, then the track has been estimated successfully. However we found these end-to-end models to be burdensome to modify and adapt to new data domains in addition of being computationally expensive for a multi-camera tracking system. A successful example of an online tracking model which relies on pre-computed detections, is SORT [33] and its enhanced version with deep associations, namely DeepSORT [34]. These trackers benefit from a linear state-space model that approximates dynamics of targets and are suitable choices for real-time applications.

Vehicle Re-Identification: Most of the recent successful works in vehicle re-identification have benefited from attention models to compute robust representations [35]–[42] for distinguishing vehicles identities and extracting minute

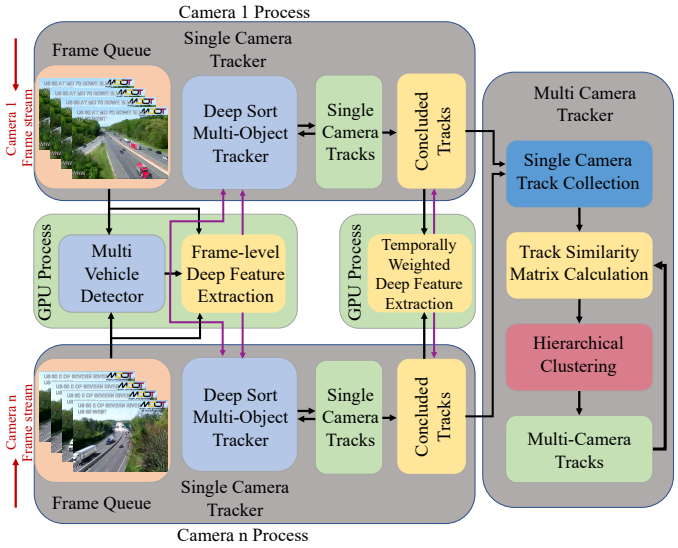


Fig. 1: Multi-Camera Vehicle Tracking in Real-Time Pipeline. For each camera in the network, a process gathers frames and sends them to a GPU process for detection and frame-level feature extraction. Next, the bounding boxes and corresponding deep features are sent back to the respective camera process for single camera tracking. Once a single camera track is concluded, all the frame-level deep features are summarized into a single representation. Finally, concluded tracks are collected and their similarity is measured to updates multi-camera tracks.

details in vehicle images. Authors in [41], [42] incorporated attention maps by localizing pre-defined key-points to enhance re-identification in a supervised manner. Typically, extracted visual representation of vehicles are biased toward the orientation in which images are captured. To alleviate this issue, authors of [39] and [40] propose to learn view-aware aligned features and to disentangle the orientation from visual features respectively. In addition, [43] proposes a heterogeneous relational model to extract region-specific features and incorporate them based on their relation into a unified representation. To overcome the demand to collect expensive annotations for learning distinguished vehicle parts, [36], [37] propose to learn salient regions of vehicles which encode identity-dependant information, in a self-supervised manner via the task of residual learning. [35] attempts to achieve the same goal via the pretext task of image rotation and degree prediction to encode geometric features along with the global appearance. Finally, authors in [44] emphasize the importance of video-based approach as opposed to the image-based approach for vehicle re-identification and introduced the VVeRI-901 dataset to contribute to this research direction.

III. Method

The overview of our proposed pipeline for real-time multi-camera vehicle tracking is shown in Figure 1. It consists of several modules which are explained in the following sections.

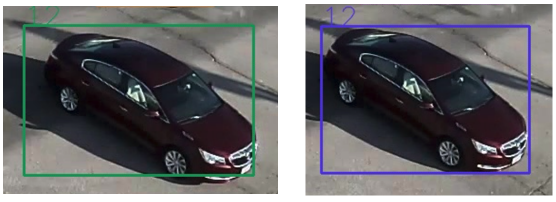
A. Camera Process

In our system, we designed a camera-specific process for each of the n cameras that are in the network so that we can process each camera in parallel and in real-time. These processes are responsible for receiving frames from each video stream and storing them in the frame queues. Upon receiving a frame it is sent to the GPU process so that vehicles can be localized and their corresponding frame-level deep appearance features can be calculated. Subsequently, they are returned to their respective camera process where the single camera tracker initiates new tracks or continues tracking previously tracked vehicles depending on the matching criteria which will be explained in section III-A1.

1) Single Camera Tracker: As we aim to design a real-time MCT system, we need to make sure all the involved components are fast and efficient while maintaining high accuracy. In addition, it is paramount that the single camera tracker has a small number of ID switches as the number of comparisons in the similarity matrix computation (required for solving multi-camera tracking) grows quadratically. Therefore, we choose DeepSort [34] as our single camera tracker. DeepSort is the successor of SORT [33] which is a very lightweight and simple multi-object tracking based on the Intersection over Union (IoU) criteria. To make tracking more robust and resilient to ID switches, DeepSort incorporates appearance information. This tracker approximates the dynamics of each target vehicle with a linear state space model. In the original implementation, the state space is defined as the following vector:

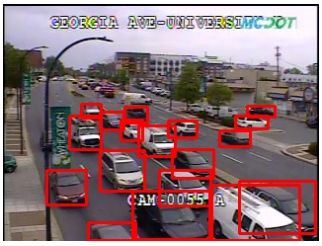
$$[u, v, r, s, \dot{u}, \dot{v}, \dot{r}, \dot{s}]^T$$

where u, v, r, and s are the bounding box's center horizontal and vertical coordinates, aspect ratio, and height respectively. In addition, their time derivatives are also included as the state space variables. However, we modify u, v to be the center point of the bottom edge of a bounding box as this point is closer to the surface of the road compared to the center of the bounding box and results in smaller distortion due to the missing depth information. DeepSort propagates the state space distribution to the current time step using a Kalman filter prediction step and obtains the predicted observation vector $[\hat{u}, \hat{v}, \hat{r}, \hat{s}]$ for a track's state at current time. However, we modify this so that whenever there is a matched detection to a predicted track's state, we use the information from the matched detection as the track's current state. Figure 2 shows

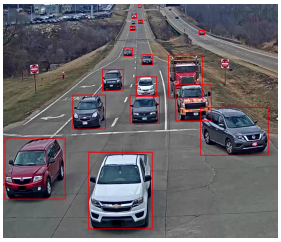


(a) Original (b) Modified

Fig. 2: Impact of replacing the Kalman filter's predicted state variables for the current frame with the observations from the matched detection. This improves the tightness of bounding boxes around vehicle tracks.



(a) RITIS Platform



(b) AI City Challenge

Fig. 3: Vehicle detection Results on sample frames after filtering out low-confidence boxes and applying NMS.

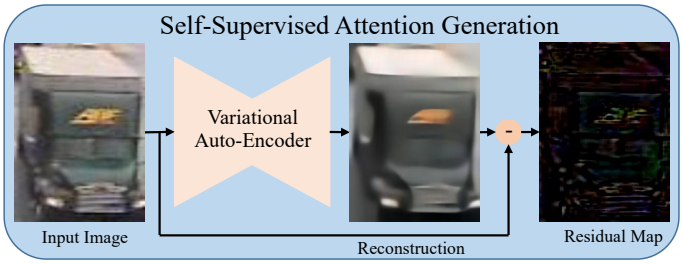


Fig. 4: The input vehicle image is coarsely reconstructed to remove the small-scale details. By subtracting the reconstruction from the input image the residual map is obtained that captures high-level details such as logos and grill design as shown above and can serve as an attention map to excite intermediate feature maps in a CNN.

the impact of this modification. This modifications is mainly due to the fact that bounding box locations from the detection module are generally more accurate compared to predictions from the tracker as it has assumptions on vehicles motion. As a vehicle track is concluded, *i.e.* the target vehicle exits the scene, all the extracted frame-level deep appearance features for its life span are temporally sorted and sent to a GPU process to be temporally weighted and summed. Therefore, a single representation vector is obtained to represent that vehicle track. The concluded track, its appearance representation, and meta-information are collected by the multi-camera tracker that is responsible for identifying multi-camera vehicle tracks.

B. GPU Process

Deep learning technology owes its success to the development of Graphical Processing Units (GPU). Almost all the recent successful Computer Vision methods benefit from GPUs. Our approach follows this trend and we run our vehicle detection and re-identification models in GPU accessible processes.

1) Multi-Vehicle Detection: To localize vehicles with high certainty, CNN-based object detectors are desirable candidates as they have achieved state-of-the-art results across different benchmarks and run efficiently on GPU cards. In this work, we choose Faster R-CNN [15], RetinaNet [14], and EfficientDet [19] to account for different scenarios, image resolution, and latency. Given a video frame I, the vehicle detector returns a list of detections in the form of

$$[x_1, y_1, x_2, y_2, \alpha, \beta]$$

where (x_1, y_1) and (x_2, y_2) show the top-left and bottom-right points of the bounding box encompassing a vehicle while α and β represent the detection confidence and class label of the detected box, respectively. Note that a minimum value of confidence score α_{min} is used to filter out unreliable detections. In addition, Non-Maximal Suppression (NMS) ensures that duplicate detections (with high IoU) for a single vehicle are merged into a single box. However, in extreme cases of occlusion this may remove the box for the occluded vehicle. While this may seem to have a negative impact and interrupts the tracking process, it precludes the extraction of erroneous features for highly occluded vehicles. Also as argued in [34], Deep Sort can recover single camera tracks that are broken up to a certain time window. Figure 3 shows detection results for frames randomly sampled from RITIS and AI City challenge data. It is worth mentioning that, on GPU cards, object detectors have the batch processing capability. At each time step, the incoming frames from all camera processes are put into a batch and detection results for all current frames of different cameras are computed at once.

2) Vehicle Re-Identification: Extracting discriminative deep features is a critical piece of a multi-camera tracking system. Small inter-class variation (different vehicles can appear very similar and share the same make, model and color) and large intra-class variation (a vehicles appearance can drastically change under different viewpoints) make vehicle re-identification challenging. Therefore, to increase the discrimination power, attention-based vehicle re-id models have been developed to incorporate details from local regions that are unique to vehicle identities. However, most of these models increased re-id accuracy at the expense of increased computational resources and inference time. Therefore, in this work, we employ the Excited Vehicle Re-identification (EVER) [37], a state-of-the art vehicle re-identification model and a top performer in 2020 and 2021 AI City Challenge [45], [46], that benefits from the self-supervised attention generation mechanism introduced in [36] without adding any overhead to the inference time. During training, intermediate layers of EVER are excited via the residual maps generated by a conditional variational autoencoder network in a self-supervised manner as proposed in [36]. The residual maps serve as pseudo-saliency maps highlighting small-scale details in vehicle images. Figure 4 shows an example of how such residual maps are generated. As training progresses, the intensity of excitation γ reduces as cosine function of training epoch, *i.e.* $\gamma(m) = 0.5 \times \left(1 + \cos(\frac{\pi m}{M})\right)$ where m, and M are current epoch and total number of training epochs. As a result, once training is finished, the inference only involves a single forward pass of a ResNet_IBN-a [47] which is quite efficient and fast, a desirable property for a real-time multi-camera tracking system. Note that we use the ResNet_IBN-a as the backbone architecture for EVER model since it is shown to be a superior candidate for the task of re-identification [48].

In contrast to the typical task of vehicle re-identification

Fig. 5: A Vehicle track images ordered in time from left to right.

which aims to compare a query image against gallery images, in a real-world multi-camera tracking scenario, the task involves comparing a query track against gallery tracks. This requires computing a representation that summarizes the extracted frame-level features of a vehicle in an effective manner. Figure 5 shows frames of a track and how they change as the vehicle passes by the camera. It shows that simply averaging all the extracted frame-level features can potentially ignore those frame-level representations that carry discriminative information on vehicle identity. For instance, when a vehicle is far from a camera, only large-scale information can be captured compared to the time it is close to camera. Since vehicle's displacement is smaller relative to when it is closer to the camera, most of the extracted frame-level features for the vehicle track only contain large-scale information and small-scale information that is critical for successful re-id is ignored. Therefore, it is important to learn how to effectively fuse frame-level information into a single representation also know as video-based re-identification. To solve this issue, our multi-camera tracking system is equipped with a Temporally Weighted Deep Feature Extractor module that is inspired by the work of [49] to compute a single representation vector for the entire length of a vehicle track. Figure 6 shows the mechanism of extracting frame-level deep features and how they are temporally weighted and averaged to obtain a single representation for the entire life span of a vehicle track. In addition, this summarization significantly simplifies the subsequent feature matching to obtain multi-camera vehicle tracks and is less memory demanding.

To train EVER, we adopt both Cross Entropy and Triplet [50] loss functions which is a common approach to train re-identification models. To make a training batch, we randomly sample K vehicle tracks with unique identities out of all the training tracks and for each selected track L frames are randomly selected and temporally ordered, *i.e.* batch size is $K * L$. In addition, all the selected frames are resized to a fixed width W and height H. As a result the computed batch is a tensor of shape $(K*L)*3*H*W$. Here we use the batch-hard sampling method for triplet loss formulated as below:

$$\mathcal{L}_t = \frac{1}{B}\sum_{i=1}^{B}\sum_{a\in b_i}\left[\gamma + \max_{p\in\mathcal{P}(a)}||f_a - f_p||_2 - \min_{n\in\mathcal{N}(a)}||f_a - f_n||_2\right]_+ \quad (1)$$

In Eq. 1, B, b_i, a, γ, $\mathcal{P}(a)$ and $\mathcal{N}(a)$ are the total number of batches, i^{th} batch, anchor sample, distance margin threshold, positive and negative sample sets corresponding to a given anchor respectively. Moreover, f_a, f_p, f_n are the extracted features for anchor, positive and negative samples. In addition, the Cross entropy loss with label smoothing technique [51] is used to alleviate the issue of over-fitting. Note that to effectively apply both cross entropy and triplet objectives to the extracted features, Batch Normalization Neck (BNNECK) [52] is employed. The Cross entropy loss is calculated as follows:

$$\mathcal{L}_c = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C}\left[y_j^i \log \hat{y}_j^i + (1-y_j^i)\log(1-\hat{y}_j^i)\right] \quad (2)$$

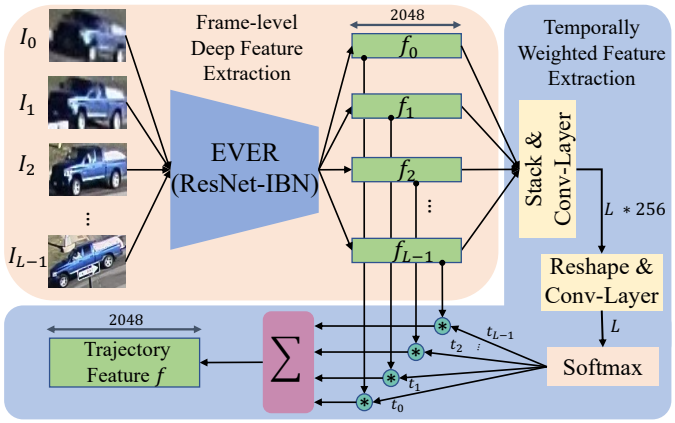
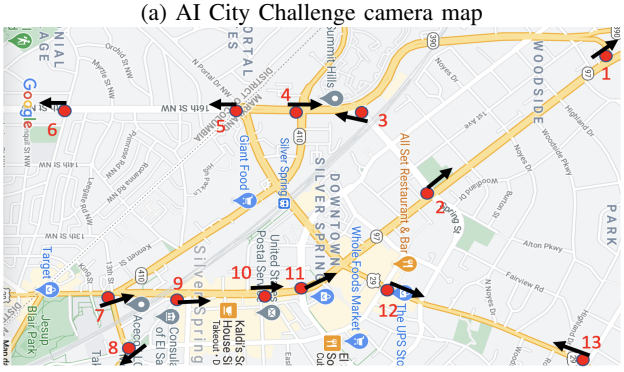


Fig. 6: Computing vehicle track representation. In Frame-level Deep Feature Extraction stage, 2048-dimensional features are computed in the GPU process. Once a single camera track of length L is concluded, temporally-ordered frame-level features are sent to another GPU process to be weighted and averaged. In this step, all the L frame-level features are stacked and passed through two convolutional layers with non-linearities to obtain L scalar values corresponding to L frames. Finally, the L frame-level features are weighted by the softmax of L scalar values and then summed to output the 2048-dimensional trajectory representation f.

(a) AI City Challenge camera map



(b) Sample RITIS system camera map

Fig. 7: Traffic cameras topology can help us reduce search to adjacent cameras only.

Where $\hat{y}_j^i = \log \frac{e^{(W_j^T f_i + b_j)}}{\left(\sum_{j=1}^{C} e^{W_j^T f_i + b_j}\right)}$ is the computed logit corresponding to class j for the extracted feature f_i of the i^{th} training sample after applying the softmax layer. Furthermore, in Eq. 2, W_j, b_j are the classifier's weight vector and bias associated with j^{th} class respectively, and N and C represent the total number of samples and classes in the training dataset. Since we use label smoothing, $y_j^i = 1 - \frac{C-1}{C}\epsilon$ if $j = c$, otherwise $y_j^i = \frac{\epsilon}{C}$ where c is the true label of i^{th} sample and $\epsilon \in [0, 1]$ is a hyper-parameter.

Camera Bias Mitigation: Another issue in multi-camera re-identification is that the orientation and background bias usually infiltrate to the computed vehicle embeddings from each camera [53] due to the limited variability. To alleviate this problem, once the model is trained, similar to [54], first we average all the vehicle representations $\{f_i^c\}_{i=1}^N$ captured in a particular camera c, resulting in a camera embedding $g_c = \frac{\sum_{i=1}^N f_i^c}{N}$. As many representations participate in this averaging, identity-dependent information can be ignored while camera-dependent information is retained. Therefore, we can reduce the impact of the camera background and orientation bias on the re-identification and cross-camera comparison by subtracting a portion λ of camera embedding g_c from vehicle representations of that camera:

$$f_i = f_i^c - \lambda g_c \tag{3}$$

where λ is a hyper-parameter. Finally, we employ the re-ranking method of [55], a common post-processing technique to enhance the distance matrix and re-identification results concurrently.

C. Multi-Camera Tracker

The multi-camera tracker is responsible for collecting the terminated single camera tracks from all the cameras and associate them into multi-camera tracks. To do so, this module has distinct sub-modules that are described below.

1) Single Camera Track Collection: As each single camera is processed in parallel and concluded single camera tracks are obtained independently, a supervisory module is required to collect the concluded tracks from all the camera processes at designated time steps depending on the query frequency in the multi-camera tracking system. Afterwards, the collected single camera tracks and existing multi-camera tracks are passed on to the next stage to compute pairwise similarities.

2) Track Similarity Matrix Calculation: To create associations among the different vehicle tracks, we need to measure pairwise similarity of tracks from the perspectives of extracted deep features and spatio-temporal information. The re-identification module is expected to extract embeddings to be similar using either Cosine or Euclidean distance for the same identities while embeddings corresponding to different identities to be dissimilar. However, due to the limited variability of vehicle types and models, lack of highly discriminating features and inherent camera view and orientation bias in embeddings, mere consideration of embedding similarity may result in erroneous multi-camera tracks. Therefore, spatio-temporal information is a valuable source of information needed to refine the similarity results. To this end, the following traffic rules are considered for the incorporation of spatio-temporal information and measuring the similarity of each pair of tracks:

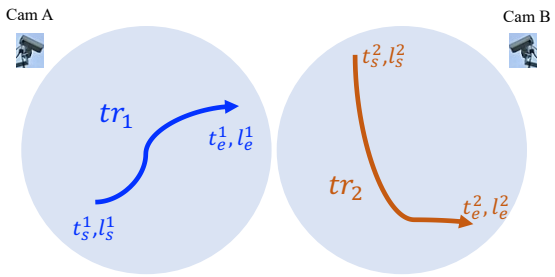


Fig. 8: In case tr_1 and tr_2 with $t_s^2 > t_e^1$ represent the same identity at neighboring cameras, the following relationships should be valid: $d(l_s^1, l_s^2) \geq d(l_e^1, l_s^2)$, and $d(l_e^2, l_e^1) \geq d(l_s^2, l_e^1)$. These criteria ensure that the direction of travel across cameras stay consistent going forward and backward in time.

1- Tracks captured through a same camera share similarities in orientation, shape and background that can negatively impact the re-identification results by severely reducing the inter-class distance which may lead to failure cases. Therefore, single camera tracks from the same camera cannot be matched together. Given the recent improvements in the area of Multi-object single camera tracking the chance of occurring ID switches has significantly reduced. Note that there is a chance that a passing vehicle might get back to the scene after a while. This can be fixed by setting a minimum time limit to consider single camera tracks from the same camera. However, as this adds additional hyper-parameters to the system, for the sake of simplicity we assume that the chance of reappearance is zero in this work. This is a reasonable assumption as the multi-camera tracks that are inactive for a while are flushed out of the system.

2- In the event that the two tracks are captured by two non-overlapping cameras, their occurrence time should not overlap. Otherwise, their similarity should be set to zero.

3- Vehicle's travel velocity should be within a reasonable range that can be determined for each particular scenario. In our implementation, similar to [8] we considered track similarity to be a quadratic function of speed, *i.e.* $sim_v = \max(0, 4\bar{v}(v_{max} - \bar{v})/v_{max})$ where v_{max} and $\bar{v} = d(l_s^2, l_e^1)/(t_s^2 - t_e^1)$ are the maximum possible and the average travel speeds between the two locations where tracks tr_1 and tr_2 are observed. Start and end time-location of tr_1 are (t_s^1, l_s^1) and (t_e^1, l_e^1). Similarly, (t_s^2, l_s^2) and (t_e^2, l_e^2) represent the start and end time-location of tr_2. Here we assume that $t_s^2 > t_e^1$. Also $d(.,.)$ represents the physical distance between the two points using their latitude and longitude. To obtain latitude and longitude of points in the image domain, extrinsic camera calibration can be performed using the Perspective-n-Point (PnP) technique and providing a set of corresponding points GPS and 2D pixel locations. This enables us to compute the homography matrix and its inverse to project points

from real-world to 2D pixel coordinates and vice versa.

4- Given the topology of traffic cameras, we can further reduce the search to only tracks that are observed in adjacent cameras. This constraint significantly reduces the number of comparisons needed for association of tracks while improving the accuracy. Figure 7 shows that the topology of cameras can be a valuable source of information which is readily available as latitude and longitude of traffic cameras are typically provided.

5- Since we reduce the search to only neighboring cameras in traffic rule 4, if two different tracks, tr_1 and tr_2 as shown in Figure 8, represent the same vehicle identity, their travel direction should stay consistent in adjacent cameras both going forward and backward in time. Assuming $t_s^2 > t_e^1$ and they represent the same vehicle identity, the following should hold true:

$$d(l_s^1, l_s^2) \geq d(l_e^1, l_s^2), \quad d(l_e^2, l_e^1) \geq d(l_s^2, l_e^1) \tag{4}$$

Based on the above-mentioned traffic rules, the pairwise similarity of tracks tr_i and tr_j can be computed as follows:

$$sim(i,j) = \begin{cases} (1 - \frac{||f_i - f_j||_2}{2}) sim_v & \text{Traffic rules are satisfied} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Note that the deep visual embeddings f_i and f_j are normalized *i.e.* $||f_i||_2 = 1$ and $||f_j||_2 = 1$. This formulation allows us to compute the similarity matrix for all the pairs of tracks that are being considered for association. Note that, all the tracks are considered as both query and gallery tracks. Therefore, the resulting similarity matrix is symmetrical and all diagonal elements are set to zero.

3) Hierarchical Clustering: Since the number of true identities is not known beforehand, we adopt a hierarchical clustering algorithm to perform this task. Once the similarity matrix is computed we need to identify those tracks representing same vehicle identities. First, we enforce that a minimum similarity value to be met in order for tracks to represent identities. Therefore, depending on the dataset, we apply a minimum threshold τ_{min} to zero out similarity values smaller than τ_{min}. Afterwards, we use hierarchical clustering algorithm which initiates each track as its own cluster and in an iterative manner merges the clusters with highest similarity. Note that the first traffic rule asserts that no more than one single camera track from the same camera can represent a vehicle identity. Therefore, once a track, *e.g.* tr_i from camera i, is merged with another track, *e.g.* tr_j from camera j, then we need to make sure that tr_i cannot be merged with any other track from camera j and track tr_j cannot be merged with any other track from camera i. This operation helps us to maintain the transitivity property among all merged tracks.

IV. Experimental Results and Implementation Details

In this section we evaluate our proposed real-time multi-camera tracking system on the traffic data streaming on the RITIS platform as well as the 2021 AI City challenge multi-camera tracking dataset. Details of each type of data, the implementation details, justification for design choices, and discussions are provided accordingly.

A. RITIS Platform Streaming Data

1) Dataset: The RITIS system gathers traffic data streams from various local municipalities throughout the United States on its platform. The streaming data provides typical metadata such as resolution, frame rate, and GPS coordinate of the camera. Sample video stills are shown in Figure 9. This data has hallmarks of true operational data including low resolution (usually 320x240), low frame rate (usually 10 fps or lower), and motion and compression artifacts. A visual inspection shows that the domain of this data significantly differs from that of other curated traffic datasets. For optimal performance on detection, *i.e.* the most fundamental task in our proposed multi-camera tracking pipeline, it is important to leverage all the available labeled data while accounting for the domain gap. Therefore, we employ labeled traffic datasets discussed in IV-A1 to perform domain adaptation and train our vehicle detection model.

Labeled Data: Two external labeled datasets are used for training the vehicle detector which will be referred to as our labeled source data.

- UA-DETRAC [20] dataset contains over ten hours of video from twenty-four different traffic cameras in Beijing and Tianjian, China. The videos were recorded at twenty-five frames per second, at the resolution of 960x540, and under various lighting conditions. Of the collected data, 140k frames were annotated which resulted in 1.21 million bounding boxes for 8250 vehicles.
- CityCam [56] dataset contains recordings from 212 traffic cameras in the United States, with a resolution of 352x240 and a frame rate of 1 frame/per second. A total of 60k frames and 900k objects were annotated.

Unlabeled Data: Instead of operating on continuously streaming data, we collected a dataset which we refer to as our unlabeled target data. This dataset contains one-minute video recordings captured by 62 different cameras in the Washington D.C - Maryland - Virginia area. The recordings were captured four times during a day: 9 am, 1 pm, 5 pm, and 8 pm. Since there are no annotations available for this dataset, to evaluate our algorithms on target data, we curated a small detection validation set containing 1408 images from 62 cameras, with an approximately equal number of images per camera. From these images, 11,147 vehicles were labeled as ground truth. The details of labelling is provided in Table I. Additionally for the tracking task, nine 1-minutes videos were labeled in their entirety as our single camera tracking validation set; the summary of labeled boxes and tracks is presented in Table II.

TABLE I: Detection validation set statistics

	Car	Bus	Truck	Van	SUV	**Total**
Boxes	9659	74	283	655	476	11147

2) Multi-Vehicle Detection: For this data, we adopt RetinaNet [14] and Faster R-CNN [15] trained on the COCO dataset. As pointed out before, the characteristics of RITIS streaming data are quite different compared to other publicly

(a) (b) (c) (d)

Fig. 9: Sample frames of traffic data obtained from RITIS platform. The low resolution, motion and compression artifacts are hallmarks of the streaming data on this platform.

TABLE II: Single camera tracking validation set statistics

	Car	Bus	Truck	Van	SUV	**Total**
Boxes	22296	555	1487	2375	518	27231
Tracks	157	3	11	11	5	187

TABLE III: The results of training the detector. The data is fine-tuned on the specified dataset and tested on the operational traffic camera data. "All" refers to training on the combination of the best performing domain-adapted UA-DETRAC dataset and the CityCam Dataset.

Model	Dataset				
	Baseline	CityCam	UA-DETRAC	DA_UA-DETRAC	All
	mAP(%)	mAP(%)	mAP(%)	mAP(%)	mAP(%)
FasterRCNN-101	64.8	73.92	37.86	61.49	**77.83**
RetinaNet-101	62.24	69.95	36.21	60.62	75.84

available benchmark datasets. Therefore, there is a domain shift which can hinder the performance. Moreover, there are no annotations available for this dataset. However, annotations such as bounding box information, transfer across domains. Hence unsupervised domain adaptation from our labeled data to unlabeled data, can help us to compensate for this gap and prepare training data for the vehicle detection module as discussed in the following section.

Domain Adaptation: A typical procedure to leverage labeled source data and a pre-trained detector is to fine-tune on the labeled source data directly and test on the target data. However, domain shift can deteriorate performance. For instance, as shown in Table III, when off-the-shelf RetinaNet and Faster R-CNN models are applied to our validation set as baseline models, the detection mAP is 62.24% and 64.8% respectively. However, once these detectors are fine-tuned on UA-DETRAC dataset, the performance drops to 36.21% and 37.86% correspondingly, showing the gap between the two domains. To leverage all available data, we use CycleGAN [57] to perform unpaired image-to-image translation to transfer domain information from the target data into the UA-DETRAC labeled source data while maintaining the inherent structure and labels of the source data. Note that no domain adaptation is needed for the CityCam dataset as fine-tuning directly improved detection results on the held out validation set over the baseline as can be seen in Table III. After we learn the mapping, we transfer the UA-DETRAC dataset to the domain of target data and reference it as DA_UA-DETRAC. Figure 10 demonstrates a sample image of UA-DETRAC dataset and its progression during the course of domain adaptation process. To measure the success of domain adaptation and control how much style was transferred, we use the detection accuracy as a proxy task. To do so, after every $10K^{th}$ iteration we record the checkpoint for the mapping function, train the object detector on that particular mapped UA-DETRAC dataset, and calculate the accuracy of detection. The mAP scores after each $10k^{th}$ iteration are shown in Figure 11. We then choose to use the domain-adapted datasets corresponding to the highest performance. Table III shows that domain adaptation leads to a nearly 24 point increase in detection accuracy once switched from UA-DETRAC to DA_UA-DETRAC.

After we obtain DA_UA-DETRAC, we combine this with the Citycam dataset to increase the size of the training set and get the best performance. Next we fine-tune both RetinaNet101 and Faster R-CNN101 models on this combined data. Note that to train object detectors, both the stem and the first residual stage of the networks are frozen while the rest of the networks are trained. Stochastic Gradient Descent optimizer with learning rate of 0.0025 and momentum of 0.9 is used for 20K iterations. Table III reports that domain adaptation of DA_UA-DETRAC along with combination with the CityCam dataset help to increase the detection accuracy approximately 13% from the baseline models. RetinaNet101 and Faster R-CNN101, with a single NVIDIA RTX 2080 Ti GPU card, need 54 and 69 milliseconds to process frames from RITIS data respectively. As Faster R-CNN101 achieved higher mAP on our validation set, we choose it to be our detection model in the multi-camera vehicle tracking. Despite being 15 milliseconds slower than RetinaNet, we can still afford to meet the real-time requirements of 100 milliseconds (10 fps).

3) Single Camera Tracking: Here we use standard evaluation metrics, *i.e.* Multiple Object Tracking (MOTA) and IDF_1 scores, as described in [58] to evaluate the single camera tracking performance. MOTA is a function of the number of false negatives, false positives, fragmentations, and true detections. MOTA accounts for the frequency of mismatches; however, it is important to consider how long these errors occur, which is enumerated by the IDF_1 score [58]. As discussed in section III-A1, we use a modified version of DeepSort to associate detected objects in a single camera. We use our tracking validation set with statistics presented in Table II to determine the best set of hyper-parameters required by this tracker. We use the Euclidean distance measure for similarity and set our IoU threshold to 0.3, above which tracks

(a) Original image (b) DA after 10k iterations (c) DA after 50k iterations (d) DA after 100k iterations

Fig. 10: The domain transfer applied to the UA-DETRAC. (a) shows the original image, while figures (b)-(d) shows the extent of domain adaptation during the course of training.

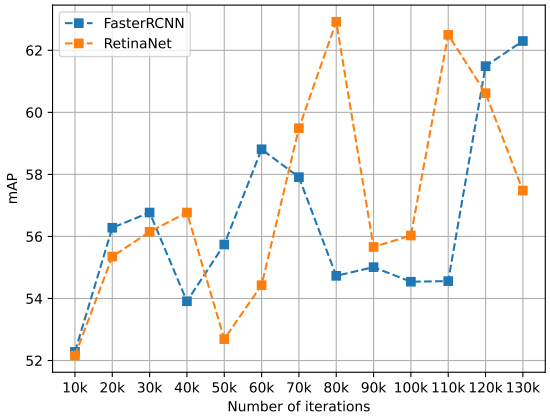


Fig. 11: Performance vs number of iterations of domain adaptation

are candidates for matching. The results, as well as the desired direction of scores for each metric, are shown in table IV.

TABLE IV: Single Camera Tracking results on our held out validation set

	IDP(↑)	IDR(↑)	IDF1(↑)	MOTA (↑)
Overall	85.5	74.5	79.7	68.7

4) Deep Feature Extraction: For the purpose of extracting frame-level deep features for the RITIS data, we use ResNet50_IBN-a as the backbone architecture of EVER model. In contrast to the vehicle detection task which mainly requires large-scale information to localize vehicles, vehicle re-identification has to deal with small-scale and subtle details of vehicles. Therefore, employing domain adaptation for publicly available datasets to be transferred to the RITIS data might degrade microscopic features and negatively impact the re-identification process. Based on this, we decided to create a vehicle re-identification dataset directly from RITIS data. In total we collected 75,685 images (858 tracks) of 475 vehicle identities across 13 cameras. 400 of these vehicle identities which form 721 vehicle tracks have been assigned for training and the rest and reserved for validation. Afterwards, we train the video-based version of EVER model as discussed in section III-B2 and by the following hyper-parameters. We set the track temporal length to $L = 5$. Also, the number of unique identities within each mini batch is set to $K = 16$. Since the resolution of videos in this domain is relatively low, *e.g.* 320×240, vehicles incorporate fewer pixels. Therefore,

TABLE V: Evaluation results of the trained EVER model on our RITIS validation set for the task of track-based vehicle re-identification.

Model Settings	mAP(%)(↑)	CMC(%)(↑)	
		@1	@5
EVER-trained ResNet50_IBN-a	35.0	32.7	44.0
+ Horizontal Flip Augmentation	37.4	33.5	44.2
+ Re-ranking	38.1	34.0	44.1
+ Camera Bias Mitigation	38.5	34.6	44.3

we resized vehicle crops to height and width $H = 64$, and $W = 64$ as opposed to 224×224 or 256×256 which are typically used. This in turn contributes to reducing the inference time. In fact, this model can compute frame-level deep features of 128 vehicle images in only 6.51 milliseconds on a NVIDIA RTX 2080 Ti GPU card. Consequently, we perform horizontal-flip test-time augmentation technique, in which we also compute the representation of the horizontally flipped tensor of a track and average it with its original representation to obtain a more robust embedding. Table V presents the track-based vehicle re-identification results on the validation set. Based on the observed performance, it can be seen that conducting re-identification in low resolution images with motion and compression artifacts is challenging. Table V shows the impactful role of test-time horizontal flip augmentation and camera bias removal.

5) Multi-Camera Tracking: Once all the required modules in our proposed multi-camera vehicle tracking system are prepared, we start tracking vehicle identities in the RITIS platform over the set of cameras that are of interest. Since RITIS gathers real-time traffic data streams, all the multi-camera processing is needed to be done in real-time to be able to catch up to the continuous stream of video frames. Our proposed method has been implemented on this platform as a prototype for multi-camera tracking and it has shown to be able to track vehicle identities in real-time. Figure 12 shows a screen shot of the interface for this prototype. The highlighted track has been re-identified in the three cameras.

B. AI City Challenge

1) Dataset: This multi-camera vehicle tracking dataset, namely CityFlow, contains 3.5 hours of traffic videos collected from 46 cameras spanning 16 intersections in a mid-sized city in Iowa, United States. The videos resolution are at least 960p with 10 frames per second. In addition, an online evaluation server is provided to rank participating teams based on the

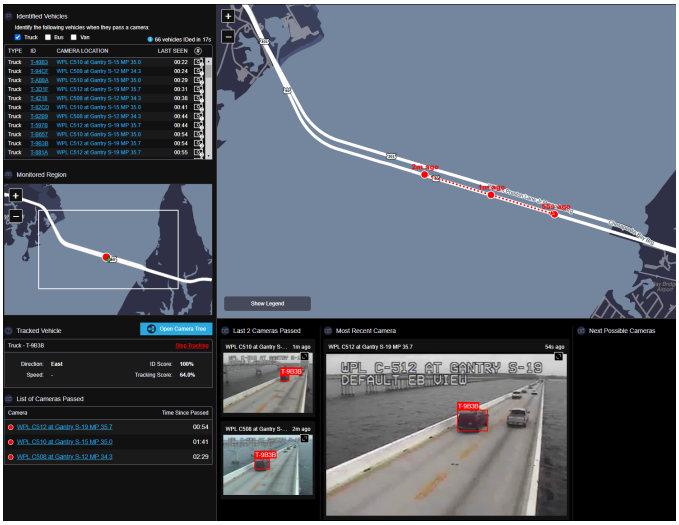


Fig. 12: Screen shot of real-time multi-camera vehicle tracking capability on three sampled neighboring cameras on RITIS platform.

TABLE VI: Inference time-Detection Accuracy comparison of popular object detection models. mAP is measured on the COCO 2017 benchmark. Inference time, represents the average time it takes for a single NVIDIA RTX 2080 Ti GPU card to process high resolution video frames of 2021 AI City Challenge with batch size of 1.

model	Backbone	mAP(%)(↑)	Inference time(ms)(↓)
RetinaNet	ResNet101	40.4	64
Faster R-CNN	ResNet101	42.0	60
	ResNext101	43.0	110
Mask R-CNN	ResNet101	42.9	63
	ResNext101	44.3	113
EfficientDet	D3	45.6	**49**
	D4	48.8	81
	D7	**50.2**	198

IDF_1 tracking metric which is the ratio of correctly identified tracks over the average number of ground-truth and computed tracks. In another words, IDF_1 balances identification precision and recall via computing their harmonic mean.

2) Multi-Vehicle Detection: For this dataset, we choose to use off-the-shelf object detector trained on COCO dataset as this data does not contain properties such as video tear or compression artifacts. RetinaNet, Faster R-CNN, Mask R-CNN and EfficientDet object detectors are considered. Table VI presents the average inference time (On a single NVIDIA RTX 2080 GPU card using batch size of 1) of these detectors on the high resolution AI City Challenge dataset along with their respective mean Average Precision (mAP) on the COCO 2017 benchmark. Among these detectors, EfficientDet D3 is a viable choice as it has the least inference time, *i.e.* suitable for real-time multi-camera tracking, while beating the detection accuracy of RetinaNet, Fatser R-CNN and Mask R-CNN. In addition, we set the object detector to only retain detections of only "car", "bus", and "truck" types with a minimum of 0.35 detection confidence. NMS with IOU threshold of 0.85 is applied to the detection results.

TABLE VII: Evaluation result of the trained EVER model on our validation set for the task of track-based vehicle re-identification.

Model Settings	mAP(%)(↑)	CMC(%)(↑)	
		@1	@5
EVER-trained ResNet101_IBN-a	54.2	61.8	80.5
+ Horizontal Flip Augmentation	54.6	62.2	80.7
+ Re-ranking	56.6	62.9	81.7
+ Camera Bias Mitigation	56.8	64.0	81.3
+ Model Ensemble	63.5	69.4	85.0

3) Deep Feature Extraction: To extract the deep visual features for this dataset, we use the ResNet101_IBN-a as the backbone architecture of the EVER model. To train this model, we use CityFlow-ReID and VehicleX [59] that are provided for the multi-camera vehicle re-identification task. CityFlow-ReID contains 85,058 images from 880 vehicle identities. Out of these images, 52,717 (2173 tracks) from 440 vehicle identities form the training set and the rest are reserved for testing. We randomly sample 100 identities from the training set as the validation set which gives rise to 11349 images (469 tracks). VehicleX is a synthetic vehicle re-identification dataset which has over 190,000 images (14663 tracks) of 1300 vehicle identities which is only provided for training purposes. However as discussed in [54], [60], there is a domain gap between this synthetic data and the real data used for evaluation. Therefore, similar to our approach for vehicle detection on the RITIS data, we used CycleGAN to reduce this domain gap and learn a generator function to translate synthetic images to real data domain. After this step, we gather all the real and domain adapted synthetic data in our training set to train EVER model and extract discriminative deep representations of a vehicle track as described in III-B2 section. Note that for this dataset, we set the number of distinct identities in each batch $K = 16$, the track length $L = 5$, and height and width of images to $H = 256, W = 256$. After training, this model computes frame-level deep visual features to be used by single camera tracker as well as the track level representations to be used by the multi-camera tracker. Note that during inference, the forward pass of the ResNet101_IBN-a for the batch size of 128 takes 13.09 milliseconds on average. This allows us to meet the real-time requirement of 100 milliseconds (10 fps) traffic videos while performing horizontal flip test-time augmentation technique. Table VII reports the track-based vehicle re-identification performance on our validation set. It is shown that a model ensemble can significantly contribute to the performance [54]. However, having an ensemble is a significant overhead that is computationally prohibitive for the real-time applications. Here we merely want to show the performance boost by considering 3 different models namely, ResNet101_IBN-a trained on real and domain adapted synthetic data, ResNet101_IBN-a only trained on real data, and a ResNet101 train on real and domain adapted synthetic data. While this improves the performance,it requires 3 times of the regular inference time and computational resources that is not possible in a real-time multi-camera tracking system unless expensive computational units are available.

TABLE VIII: Our proposed real-time multi-camera tracking results on the test set of 2021 AI City Challenge.

Settings	$IDP(\uparrow)$	$IDR(\uparrow)$	$IDF_1(\uparrow)$
Baseline	0.5207	0.4469	0.4810
+ Traffic Rule #4	0.5572	0.6196	0.5867
+ Traffic Rule #5	0.6608	0.6890	0.6746
+ Hyper-parameter Turning	**0.7088**	**0.7292**	**0.7189**

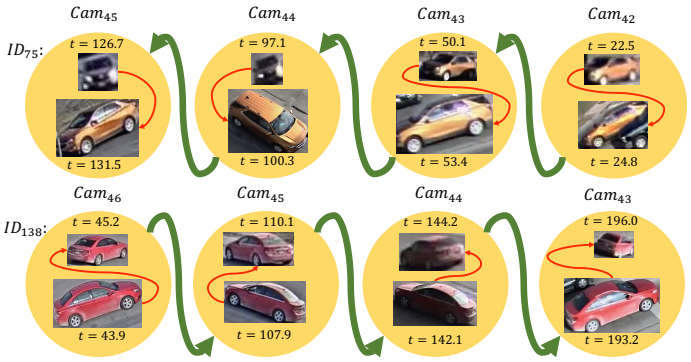


Fig. 13: Sample of two Multi Camera tracks generated by our method for the AI City challenge test data.

4) Multi-Camera Tracking: After putting all the modules in our proposed real-time multi-camera tracking system, we evaluate our approach on the test set of 2021 AI City Challenge and compare against submissions to this challenge. Note that submissions are ranked for this challenge only based on IDF_1 tracking score without any consideration for processing time. In addition, the evaluation server provides IDP and IDR evaluation metrics. The test set is composed of 6 simultaneously recorded videos of resolution 1280x960 from 6 different cameras as shown in Figure 7a. The length of each video is 3 minutes and 20 seconds. Table VIII summarizes our performance on this dataset and shows the impact of the traffic rules we considered during similarity matrix computation. The baseline model refers to our proposed system without the consideration of traffic rules #4 & #5. From Table VIII we can appreciate the impact of effective incorporation of spatio-temporal information on the refinement of multi-camera tracking results. Most notably, by reducing the search only to neighboring cameras, *i.e.* Traffic Rule #4, there is a significant boost in the IDR metric showing that capability of recalling the multi-camera tracks is very well improved. Finally, in the public leaderboard of the challenge our method is ranked 5^{th} out of 23 participating teams without any considerations for the computational efficiency, inference time, and entering additional spatio-temporal information beyond what is provided by the challenge organizers. For instance, [5] shows that introducing crossroad zones, *i.e.* the connectivity of the different road zones in one camera to the neighboring cameras, search for multi-camera tracks can be extremely refined.

V. Conclusion

In this work, we highlighted the importance of designing a scalable and real-time multi-camera vehicle tracking system that can provide precise and timely information for transportation applications. Moreover, we shed light on the practical issues involved in achieving this goal and subsequently propose our multi-camera tracking system to address these issues and can process camera feeds in parallel and identify vehicle identities over a network of traffic cameras in real-time. Thanks to its effectiveness, it has been adopted in the Regional Integrated Transportation Information platform to provide real-time understanding of the status of traffic and identify sources of traffic congestion. In addition, we have participated in the 2021 NVIDIA AI City city-scale multi-camera tracking challenge and our model is ranked among the top five contestants without any consideration of processing time and computational complexity of submissions.

References

[1] W. H. Organization, "Road safety facts," https://www.who.int/news-room/facts-in-pictures/detail/road-safety, December 2018.

[2] W. S. D. of Transportation, "Traffic signals," https://wsdot.wa.gov/travel/operations-services/traffic-signals, December 2018.

[3] X. Zhou, V. Koltun, and P. Krähenbühl, "Tracking objects as points," 2020.

[4] P. Bergmann, T. Meinhardt, and L. Leal-Taixé, "Tracking without bells and whistles," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[5] C. Liu, Y. Zhang, H. Luo, J. Tang, W. Chen, X. Xu, F. Wang, H. Li, and Y.-D. Shen, "City-scale multi-camera vehicle tracking guided by crossroad zones," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2021, pp. 4129–4137.

[6] M. Wu, Y. Qian, C. Wang, and M. Yang, "A multi-camera vehicle tracking system based on city-scale vehicle re-id and spatial-temporal information," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4077–4086.

[7] J. Ye, X. Yang, S. Kang, Y. He, W. Zhang, L. Huang, M. Jiang, W. Zhang, Y. Shi, M. Xia *et al.*, "A robust mtmc tracking system for ai-city challenge 2021," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4044–4053.

[8] Z. He, Y. Lei, S. Bai, and W. Wu, "Multi-camera vehicle tracking with powerful visual features and spatial-temporal cue," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 203–212.

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[10] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[11] ——, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

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

[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[14] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," 2018.

[15] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016.

[16] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[17] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[18] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[19] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 781–10 790.

[20] L. Wen, D. Du, Z. Cai, Z. Lei, M. Chang, H. Qi, J. Lim, M. Yang, and S. Lyu, "UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking," *Computer Vision and Image Understanding*, 2020.

[21] H. Perreault, G.-A. Bilodeau, N. Saunier, and M. Héritier, "Spotnet: Self-attention multi-task network for object detection," 2020.

[22] Z. Fu, Y. Chen, H. Yong, R. Jiang, L. Zhang, and X. S. Hua, "Foreground gating and background refining network for surveillance object detection," *IEEE Transactions on Image Processing*, vol. 28, no. 12, pp. 6077–6090, 2019.

[23] S. Wu, M. Kan, S. Shan, and X. Chen, "Hierarchical attention for part-aware face detection," *International Journal of Computer Vision*, vol. 127, no. 6, pp. 560–578, Jun 2019. [Online]. Available: https://doi.org/10.1007/s11263-019-01157-5

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, Nov. 1997. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[25] W. Liu, S. Liao, W. Ren, W. Hu, and Y. Yu, "High-level semantic feature detection: A new perspective for pedestrian detection," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 5182–5191.

[26] K. Shafique and M. Shah, "A noniterative greedy algorithm for multiframe point correspondence," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 1, pp. 51–65, 2005.

[27] D. Reid, "An algorithm for tracking multiple targets," *IEEE Transactions on Automatic Control*, vol. 24, no. 6, pp. 843–854, 1979.

[28] G. Shu, A. Dehghan, O. Oreifej, E. Hand, and M. Shah, "Part-based multiple-person tracking with partial occlusion handling," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 1815–1821.

[29] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes, "Globally-optimal greedy algorithms for tracking a variable number of objects," in *CVPR 2011*, 2011, pp. 1201–1208.

[30] A. A. Butt and R. T. Collins, "Multi-target tracking by lagrangian relaxation to min-cost network flow," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1846–1853.

[31] H. B. Shitrit, J. Berclaz, F. Fleuret, and P. Fua, "Multi-commodity network flow for tracking multiple people," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 8, pp. 1614–1627, 2014.

[32] S. Sun, N. Akhtar, H. Song, A. Mian, and M. Shah, "Deep affinity network for multiple object tracking," 2019.

[33] 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.

[34] 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.

[35] M. Li, X. Huang, and Z. Zhang, "Self-supervised geometric features discovery via interpretable attention for vehicle re-identification and beyond," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 194–204.

[36] P. Khorramshahi, N. Peri, J.-c. Chen, and R. Chellappa, "The devil is in the details: Self-supervised attention for vehicle re-identification," in *European Conference on Computer Vision*. Springer, 2020, pp. 369–386.

[37] N. Peri, P. Khorramshahi, S. S. Rambhatla, V. Shenoy, S. Rawat, J.-C. Chen, and R. Chellappa, "Towards real-time systems for vehicle re-identification, multi-camera tracking, and anomaly detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 622–623.

[38] T.-S. Chen, C.-T. Liu, C.-W. Wu, and S.-Y. Chien, "Orientation-aware vehicle re-identification with semantics-guided part attention network," in *European Conference on Computer Vision*. Springer, 2020, pp. 330–346.

[39] D. Meng, L. Li, X. Liu, Y. Li, S. Yang, Z.-J. Zha, X. Gao, S. Wang, and Q. Huang, "Parsing-based view-aware embedding network for vehicle re-identification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7103–7112.

[40] Y. Bai, Y. Lou, Y. Dai, J. Liu, Z. Chen, L.-Y. Duan, and I. Pillar, "Disentangled feature learning network for vehicle re-identification." in *IJCAI*, 2020, pp. 474–480.

[41] P. Khorramshahi, A. Kumar, N. Peri, S. S. Rambhatla, J.-C. Chen, and R. Chellappa, "A dual-path model with adaptive attention for vehicle re-identification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6132–6141.

[42] P. Khorramshahi, N. Peri, A. Kumar, A. Shah, and R. Chellappa, "Attention driven vehicle re-identification and unsupervised anomaly detection for traffic understanding," in *CVPR Workshops*, 2019.

[43] J. Zhao, Y. Zhao, J. Li, K. Yan, and Y. Tian, "Heterogeneous relational complement for vehicle re-identification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 205–214.

[44] J. Zhao, F. Qi, G. Ren, and L. Xu, "Phd learning: Learning with pompeiu-hausdorff distances for video-based vehicle re-identification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2225–2235.

[45] M. Naphade, S. Wang, D. C. Anastasiu, Z. Tang, M.-C. Chang, X. Yang, L. Zheng, A. Sharma, R. Chellappa, and P. Chakraborty, "The 4th ai city challenge," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020, p. 2665–2674.

[46] M. Naphade, S. Wang, D. C. Anastasiu, Z. Tang, M.-C. Chang, X. Yang, Y. Yao, L. Zheng, P. Chakraborty, C. E. Lopez, A. Sharma, Q. Feng, V. Ablavsky, and S. Sclaroff, "The 5th ai city challenge," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2021.

[47] X. Pan, P. Luo, J. Shi, and X. Tang, "Two at once: Enhancing learning and generalization capacities via ibn-net," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 464–479.

[48] L. He, X. Liao, W. Liu, X. Liu, P. Cheng, and T. Mei, "Fastreid: A pytorch toolbox for general instance re-identification," *arXiv preprint arXiv:2006.02631*, 2020.

[49] J. Gao and R. Nevatia, "Revisiting temporal modeling for video-based person reid," *arXiv preprint arXiv:1805.02104*, 2018.

[50] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *arXiv preprint arXiv:1703.07737*, 2017.

[51] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[52] H. Luo, Y. Gu, X. Liao, S. Lai, and W. Jiang, "Bag of tricks and a strong baseline for deep person re-identification," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[53] X. Zhu, Z. Luo, P. Fu, and X. Ji, "Voc-reid: Vehicle re-identification based on vehicle-orientation-camera," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[54] H. Luo, W. Chen, X. Xu, J. Gu, Y. Zhang, C. Liu, Y. Jiang, S. He, F. Wang, and H. Li, "An empirical study of vehicle re-identification on the ai city challenge," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2021, pp. 4095–4102.

[55] Z. Zhong, L. Zheng, D. Cao, and S. Li, "Re-ranking person re-identification with k-reciprocal encoding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1318–1327.

[56] S. Zhang, G. Wu, J. P. Costeira, and J. M. F. Moura, "Understanding traffic density from large-scale web camera data," *CoRR*, vol. abs/1703.05868, 2017. [Online]. Available: http://arxiv.org/abs/1703.05868

[57] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," 2020.

[58] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *European conference on computer vision*. Springer, 2016, pp. 17–35.

[59] Y. Yao, L. Zheng, X. Yang, M. Naphade, and T. Gedeon, "Simulating content consistent vehicle datasets with attribute descent," in *The European Conference on Computer Vision (ECCV)*, August 2020, p. 775–791.

[60] P. Khorramshahi, S. S. Rambhatla, and R. Chellappa, "Towards accurate visual and natural language-based vehicle retrieval systems," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2021, pp. 4183–4192.