Multi-camera vehicle tracking and re-identification based on visual and spatial-temporal features

Xiao Tan, Zhigang Wang, Minyue Jiang, Xipeng Yang, Jian Wang, Yuan Gao,
Xiangbo Su, Xiaoqing Ye, Yuchen Yuan, Dongliang He, Shilei Wen and Errui Ding

Baidu Research, Baidu Inc.

{tanxiao01,wangzhigang05,jiangminyue,yangxipeng01,wangjian33,gaoyuan18}@baidu.com
{suxiangbo,v_yexiaoqing,yuanyuchen02,hedongliang01, wenshilei,dingerrui}@baidu.com

Abstract

Due to the heavy occlusions, large variations in different viewing perspectives and low video resolutions, the tracking and re-identification of vehicles under multi-camera become challenging tasks for the intelligent transportation system (ITS). In this work, we propose a novel framework for multi-camera tracking, which integrates visual features, orientation prediction and temporal-spatial information of the trajectories for optimization. In addition, based on the tracking information generated by our framework, we propose a united method for multi-camera re-identification that takes both visual features and tracking information into account. In order to make the visual feature robust for occlusion and perspective variation, our method adopts various features that are extracted from global image, regions and areas around key points, and the tracking information are also used to refine the retrieval results generated by the visual features. Our algorithm achieves the first place in vehicle re-identification at the NVIDIA AI City Challenge 2019.

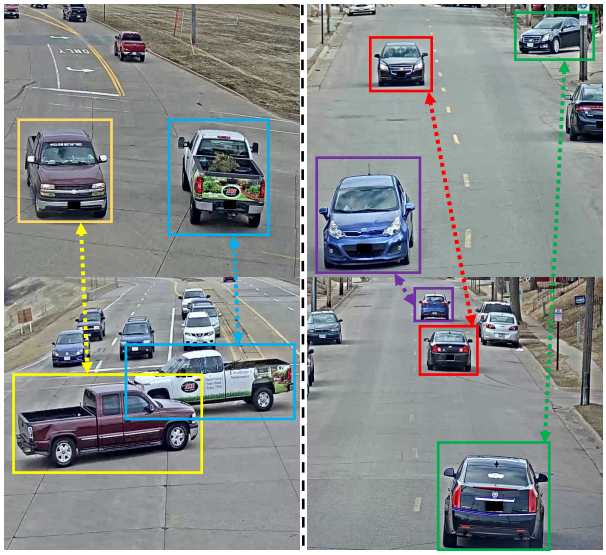

Figure 1. Overview of multi-camera tacking and re-identification, large variations in different viewing perspectives make the task very challenging.

1. Introduction

The Intelligent Transport System (ITS) has drawn increased interest in both academia and industry [28, 30] since it is a critical component in building an AI-city. For instance, it facilitates transportation design and optimization by the estimation of traffic flow characteristics, and provides more comprehensive information about the roads and their surrounding environments for automated driving technology. However, there is no off-the-shelf solution to this problem at present. Conventional multiple object tracking (MOT) [4, 26] typically focus on human tracking by using a single camera in a relatively limited region. In the scenario of ITS, however, it is required to identify vehicles under separated cameras in different locations that are usually faraway to each other. In addition, several inherent properties of vehicle make the task even more challenging. For example, due to the variance in appearance of different facets of a specific vehicle, it is difficult to identify the vehicles from different view directions. Second, the vehicles may in swift motion, which results in blurred footage. Finally, occlusions are inevitable in the case of busy traffic, especially at traffic pivots.

Fig. 1 shows different scenarios of an ITS. In the left part, the same vehicle may appear in overlapping cameras; while in the right part, the cameras are located in different places, and the same vehicle may appear in these cameras with particular order. An ITS should be robust to both the scenarios above. In this paper, we design a new ITS which

consists of a single camera tracking (SCT) module and an inter-camera association (ICA) module. The SCT module takes a video sequence of a single camera and performs detection and tracking of the vehicles shown up within. It also computes the feature representation of all the tracklets. The ICA module matches the identities of vehicle tracklets in different cameras by using features in terms of appearance similarity, as well as motion feature under the geometry and motion constraints.

1.1. Single Camera tracking (SCT)

The SCT is typically carried out in 2D image domain (in terms of pixels). However, if the camera is calibrated and the 6 degrees of freedom (6DoF) of a vehicle is available, we can project the 2D location of a vehicle back to the real 3D coordinate system. Nevertheless, occlusion and tracklet intercrossing need to be properly handled.

1.2. Inter-camera Association (ICA)

In the ICA module, the vehicle re-identification task is formulated into a constrained clustering problem, where the objective is to maximize the inter-association of tracklets generated by SCT under the constraints that each cluster contains a single instance identity. To measure the association between tracklets of different cameras, feature representations are computed. These features are used to distinguish different instances, which include appearance features and motion features. The appearance features include low-level semantic features such as color histogram and local binary pattern and high-level appearance features extracted by Deep Convolution Neural Networks (DCNN). The appearance features and the motion features in terms of location and velocity are integrated for association computation.

As for the vehicle re-identification part, we propose a united method that takes both visual features and tracking information into account where visual features are designed to be robust for occlusions and perspective variations.

The rest of this paper is organized as follows. A brief review of the related work is summarized in Section 2. A comprehensive description of our approach is given in Section 3. In Section 4, we show the evaluation results of our approach. And finally, the conclusion is drawn in Section 5.

2. Related Works

2.1. Objection Detection

Recently, object detection becomes popular in both two-stage and single-stage detectors [21][15]. In the single-stage pipelines, the locations of the target objects are generated by a single CNN, and in two-stage pipelines, *e.g*. Fast R-CNN[8] and Faster-RCNN[24], the final predictions are generated from features which are generated in a specific region of interests. In spite that single-stage detectors are efficient, state-of-the-art object detectors in the term of accuracy usually adopt two-stage approaches. Recently, detection frameworks with multiple stages emerge as an increasingly popular paradigm, *e.g*. cascade R-CNN [3]. In the cascade RCNN, the output of each stage is fed into the next stage for more accurate bounding box refinements.

2.2. Single-Camera Tracking

Multiple object tracking (MOT) within a single camera has been a challenging task in computer vision, especially when in difficult circumstances, such as object occlusion or objects with similar appearance feature exist. Most existing MOT methods tend to establish the object tracklets by solving a template-matching problem based on various object features. For example, [1] applies a rudimentary combination of the Kalman filter and Hungarian algorithm and achieves decent accuracy and efficiency. [33] extends the work of [1] and employs deep convolutional neural network (CNN) to extract features and use these features for tracking, which contributes to greatly improved performance. However, when the occlusion parts become irregular or unpredictable, such as the fast changing of viewpoints of vehicles under traffic scenes, better strategies need to be considered. [16] utilizes 3D deformable vehicle models to define multiple kernels in 3D space. [29] combines kernel-based MOT with camera self-calibration for automatic 2D-to-3D back-projection. And more recently, [25] proposes 3D bounding box based model for fine-grained vehicle recognition. The 3D modeling of vehicles provides satisfactory performance when the contours of each car can be clearly defined, *e.g*. in light traffic condition. However, when the traffic is heavy, its performance is witnessed to drastically degrade due to collision of the contours of vehicles. Therefore, more advanced MOT methods for traffic scenes that can effectively solve the occlusion issues above are highly desirable.

2.3. Vehicle Re-identification

Vehicle re-identification is also improved significantly in recent years with the help of CNN, such as [9], [20]. [34] use diverse vehicle attributes to mine the correlations between different vehicle images. [14] summarizes existing loss functions, sampling methods and other training tricks, based on which an efficient re-identification framework is built. However, for vehicle re-identification, the variations of vehicle orientation still make the re-identification task difficult. To alleviate such issue, methods like synthesis multi-view features by transforming single-view features [36] and orientation invariant feature embedding [32] are hence proposed.

2.4. Multi-camera tracking

Multi-camera tracking is a frontier research area in recent years. Due to challenges such as occlusion, large variation in different viewing perspectives and low video resolution, it is usually hard to achieve satisfying results by visual feature based methods alone. On the other hand, [30] proposes a fusion framework for multi-camera tracking, which achieves the top performance in the NVIDIA AI City Challenge 2018.

3. Methodology

For the multi-camera tracking task, we mainly follow the tracking-by-detection paradigm for single camera as well as across multiple cameras. Fig. 3 demonstrates the pipeline of our method. Firstly, vehicle detection is conducted on each frame of all the cameras; after which we perform single camera tracking to get the tracklets; then visual features are extracted for each tracklet by our vehicle ReID models; finally, we use multi-camera tracking to match tracklets from multiple cameras. For the vehicle re-identification task, we first use discriminative visual features that are relatively robust to orientation and occlusion to get the initial ranking result, which is then refined by taking the tracking information from multi-camera tracking into account. Each algorithmic component will be elaborated in detail as below.

3.1. Objection Detection

We use a multi-stage cascade R-CNN [3] to build our detection framework, which adopts SENet [11] as the backbone feature extractor. To increase the global context information in the extracted features, we add FPN [17] to the backbone. The RoIAlign [10] is replaced by deformable RoI pooling [5] to encounter the camera distortion problem. Multi-scale and data flipping are exploited as our data augmentation for training, and only multi-scale is employed for testing. As for the post-processing, softNMS [2] is used to further boost the detection recall performance. The detection training dataset is comprised of COCO [18], KITTI [7] and AICity [28].

3.2. Single-Camera Tracking

After obtaining the bounding boxes through detection of each frame, we project the bounding box locations into the 3D space by the intrinsic parameters of the cameras. We follow the pipeline of DeepSORT [33] on 3D space to obtain short tracklets, and then perform tracklets association to formulate long tracklets. Different from [33], we define the state space in Kalman filtering framework as $(u,v,r,h,\dot{x},\dot{y},\dot{r},\dot{h})$, which contains the center position of bounding box in 3D space $(u,v,1)$, aspect ratio r, height h, and their respective velocities in 3D coordinates. We use CNN feature of objects described in 3.3 to obtain similarity

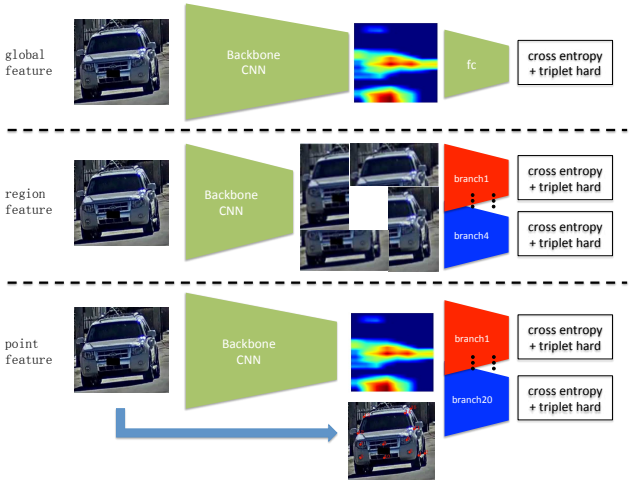

Figure 2. Illustration of different re-identification methods.

matrix between detections and tracklets, then perform Hungarian algorithm to match the detections and tracklets. Similarly, the Mahalanobis distance between the detected object and the predicted position by Kalman filter is calculated as gate, in order to disregarded infeasible assignments. Since there are occluded, missing and noisy detections, which might lead one tracklet to be confused by several similar tracklets, we need to associate and integrate the tracklets in the same camera by defining the appearance similarity and the motion similarity. The confused tracklets mainly appear in two situations: temporally overlapping and disconnection. For temporally disconnected tracklets, we directly adopt the appearance feature of the object on the last tracklet frame, where the interruptions commonly occur. Thus, we first cluster the appearance features of objects in a tracklet, and then use its cluster center, which is free of outliers, as the tracklet's representation. The appearance similarities are then calculated to perform bipartite graph matching by the Hungarian algorithm. For temporally overlapping tracklets, we calculate the intersection of union (IoU) between the bounding boxes of two tracklets in the overlapped interval. If the IoU is greater than a pre-defined threshold, they will be merged into one tracklet.

3.3. Vehicle Re-identification Appearance Model

Our vehicle re-identification method consists of two components, *i.e.* image based feature embedding module and video based constraint module. As shown in Fig. 2, our image based feature embedding module has three parts, global feature, multiple granularities region feature, keypoint based local feature. Global feature is used to describe the overall appearance of a vehicle, which is the linear transformation of the pooling feature of last convolution module in CNN. In order to make the global feature more salient, we add self-attention constrain [12], denoted

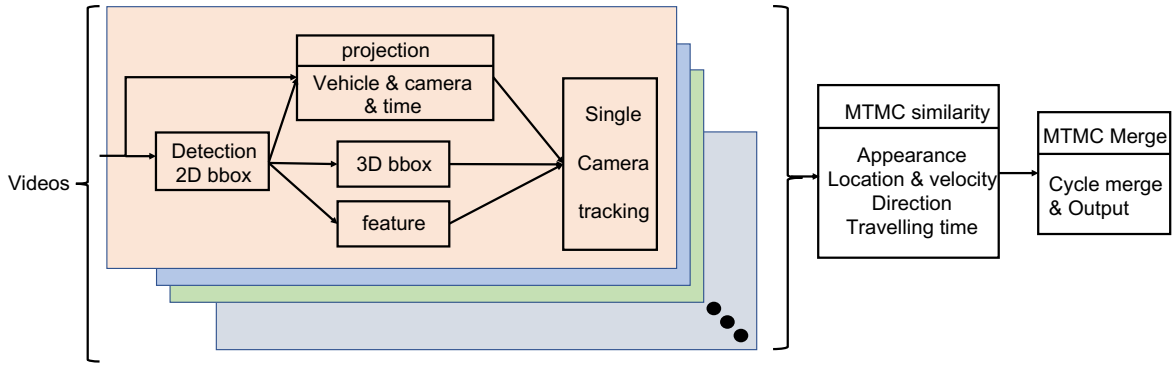

Figure 3. The framework of our multi-target multi-camera tracking method, where each color represents one camera.

as SAC, to the backbone to make network pay more attention to the spatial regions which have larger activation on feature maps. However, global feature alone is insufficient to distinguish the vehicle precisely, as it is in lack of features from discriminative vehicle parts. Fortunately, many existing works in person re-identification and vehicle re-identification have proven that extracting feature from the multiple image parts could improve the performance of feature significantly. Here, we adopt the framework of multiple granularities network [31], denoted as MGN, for learning on semantic parts. Instead of only using several horizontal stripes in person re-identification, we use two horizontal stripes and two vertical stripes, which is more suitable for representing vehicle semantic parts. Nevertheless, vehicles may still have detailed information that is hard to be found by global features and semantic part features, such as logo, light, etc. Inspired by [32], we use another CNN network to predict several key points which sited on the key parts of vehicle, and then extract feature around these key parts by the assistance of heatmap generated from key point network for further classification and metric learning. Finally, we assemble several features from aforementioned models as the appearance representation. The final feature is not only used in vehicle re-identification, but also serviced in single camera tracking and multi-camera tracking.

3.4. Multi-Camera Tracking

Multi-target cross-camera tracking (MTMC) combines a variety of information, including DCNN and spatial-temporal features. Our target is to fuse these features to calculate the losses. The objective function of vehicle similarity can be expressed as:

$$S = W_a * S_a + W_l * (S_l + S_v) + W_d * S_d + W_t * S_t, \quad (1)$$

where W_a, W_l, W_d and W_t is the weight to balance each term. S_a is appearance similarity for each tracklet. S_l and S_v is similarity of location and velocity if the cameras have overlaps, otherwise $S_l = S_v = 0$. S_d is direction item, which computes the similarity of vehicles motion direction between leaving camera and reaching camera. All these terms will be described below in detail. S_t is traveling time item, which metrics the similarity of predicted traveling time and truth traveling time between two not overlapped cameras, if cameras have overlap, $S_t = 0$.

Appearance similarity

We extract appearance feature from by using metric learning, and for each tracklet we select 5 features from all tracklets to represent it. To compute the appearance distance between two tracklets, we first compute the cosine distance between two features sets containing 5 features for each tracklet, and a 5×5 matrix is obtained. Then we select the minimum value as the two trajectories distance. And compute similarity S_a by following function:

$$S_a = exp(-\frac{d_a}{\delta_a}), \quad (2)$$

where d_a is two trajectories Euclidean distances, δ_a is appearance distance variance parameter.

Overlap location similarity

For each tracklet, we select the first and last 5 frames to estimate incoming and outgoing features in the terms of velocity and location on ground. Based on the single camera tracking result, we can estimate the start time and end time of each tracklet. Under two overlapped cameras, if two tracklets are considered covering the same identity, their timestamp time will overlap, and the similarity between the incoming feature and outgoing feature is small. Location distance d_l and velocity distance d_v is computed by l_2 distance. If two cameras have not overlap, the distance is a largest value. We compute the similarity by following function:

$$\begin{aligned} S_l &= exp(-\frac{d_l}{\delta_l}), \\ S_v &= exp(-\frac{d_v}{\delta_v}), \end{aligned} \quad (3)$$

where d_l and d_v is respective to the all location distance and velocity distance. δ_l and δ_v is the location distance and velocity distance variance parameter.

Direction similarity

From above we can get the outgoing direction and incoming direction in terms of outgoing velocity and incoming velocity. Using the point of bottom center of bounding box to represent a vehicle. To compute the ground location of a camera, we select the closest points of vehicle tracklet to the center of the bottom edge of the image, then project this point on ground to represent the ground location of this camera. Finally, we compute the direction of outgoing camera and incoming camera defined by the ground location of two cameras. According to the outgoing direction, incoming direction and camera direction, we compute outgoing angle and incoming angle respective to the camera direction, if the two tracklets with same identity, the two angles should be similar. The direction distance d_d is the computed by the two angles. Direction similarity is computed as follows:

$$S_d = exp(-\frac{d_d}{\delta_d}), \tag{4}$$

where d_d is direction distance. δ_d the direction distance variance parameter.

Traveling time similarity

The traveling time estimated by matched tracklets. So we first carry out tracklet matching by using three terms as mention above get initial matching results. For cameras not sharing overlapped region, we estimate the traveling time by kernel regressors use arrival time and outgoing time. Then compute the traveling time difference in estimated traveling time and real traveling time. The real traveling time is computed by outgoing time and arrival time of potential matching tracklet pairs. Traveling time similarity is computed by following function:

$$S_t = exp(-\frac{d_t}{\delta_t}), \tag{5}$$

where d_t is travelling distance. δ_t is the traveling time distance variance parameter. If tracklet pairs are in cameras with overlapping regions, then the traveling time similarity is set to 0.

Clustering Tracklets

The clustering is carried out in two steps. Firstly, an initial clustering is achieved by using Hungarian algorithm where we try to assign all tracklets of a camera to a cluster from current clustering results. The distance between a tracklet and a cluster determines the cost to the Hungarian algorithm, which is defined as the minimum distance between a tracklet and all tracklets in a cluster. More concretely, if a tracklet is assigned to a cluster by Hungarian and the assignment cost is smaller than a threshold T_h, this assignment will be confirmed. While, a new cluster will be assigned to the tracklet if this tracklet cannot be assigned to any clusters or the assignment cost by Hungarian algorithm is larger than a threshold T_h. We select a camera and assign a cluster for each tracklet, which is treated as current clustering results, and then we randomly picking a camera not processed and try to merge its all tracklet with current clustering results. This process is carried out until all cameras are processed and an initial clustering is generated. However, the tracklets of a same vehicle identity may not be assigned to a same cluster owing that this initial clustering is vulnerable to the process sequence of cameras. We hence propose a graph based clustering approach to further merge the initial clustering results. This method is inspired by the graph based image segmentation [6]. Assume we have two clusters C_m and C_n containing m and n tracklets respectively, the distance between these two clusters is defined by the minimum distance between tracklets in two clusters: $dis\left(C_n, C_m\right) = \min dis\left(i, j\right)$ where $i \in C_m$ and $j \in C_n$. Given an initial clustering results containing N clusters, we compute all $N * \left(N-1\right)/2$ distances among them and sort the distance in asend order. Then we check all distance values and merge clusters with the following criteria:

(1) If two clusters are already merged into a same cluster, then do nothing.

(2) If there are two tracklets from the same camera but different clusters, then the two clusters will not be merged.

(3) Otherwise, merge the tracklets in the two clusters and generate a larger one.

3.5. Video-based Multi-Constraint Vehicle ReID

Due to the large intra-class variations and similar inter-class appearances, image-based vehicle ReID remains a challenging task even with the help of deep learning. Therefore, in this paper, we propose a video-based multi-constraint vehicle ReID method that significantly improves the performance. We denote the query set and gallery set of CityFlow-ReID as $Q = \{q_i\}_{i=1}^{M}$ and $G = \{g_j\}_{j=1}^{N}$ where M and N are query numbers and gallery numbers, respectively. For each vehicle, the corresponding tracklet sequence is first obtained by our multi-camera tracking method. Subsequently, we densely compare the euclidean feature distance between q_i (or g_j) and every image in each tracklet sequence to determine which tracklet q_i belongs to. Meanwhile, the camera information and tracklet duration of q_i can also be obtained. After that, a simple but very effective constraint can be used, *i.e.* if q_i and g_j are from the same camera but with different track Ids, we push them far from each other in the feature space. This constraint can filter lots of false positives that are similar with a certain query in appearance and direction, which is formulated as follows.

$$\underset{cam(i,j)=1, tid(i,j)=0}{d(i,j)} = \begin{cases} d(i,j) + \sigma 1 & d(i,j) <= \tau 1 \\ d(i,j) + \sigma 2 & otherwise \end{cases}, \tag{6}$$

where $d(i,j)$ denotes the euclidean feature distance between q_i and g_j. $cam(i,j)$ is a indicator function in which the value is 1 indicates q_i and g_j are from the same camera. Similarly, $tid(i,j) = 0$ indicates q_i and g_j belong to different tracklets. $\tau 1$ is a threshold, $\sigma 1$ and $\sigma 2$ are two punish-items. When q_i and g_j correspond to the same vehicle while the tracking sequence is separated into several tracklets due to some errors, their $d(i,j)$ should be less than a threshold, in this case, we set a small value to punish-item $\sigma 1$. Otherwise, q_i and g_j are supposed to be different vehicles, and we set a larger value to $\sigma 2$.

As indicated in the scenario 2 of [27], the directions of cameras 7 and 8 are opposite, so are cameras 6 and 9. Based on this observation, we can infer that when the same vehicle pass through the above camera pairs, the captured images should have different vehicle orientations. A direction classification model, as shown in Fig. 4, is trained on the direction-extended training set of CityFlow-ReID (see section 4.1). Both the query and gallery set of CityFlow-ReID are predicted using the direction classification model to get the direction probability vectors normalized by softmax function. When q_i and g_j are from the same camera of scenario 2, the direction constraint can be conducted using the following function.

$$d(i,j) = \begin{cases} d(i,j) + \sigma 3 & \langle v_i, v_j \rangle > \tau 2 \\ d(i,j) & othersize \end{cases}, \tag{7}$$

where v_i and v_j are direction probability vectors of q_i and g_j, $\langle \cdot \rangle$ stands for dot product operation, $\tau 2$ is a similarity threshold and $\sigma 3$ is a punish-item. In short, according to the aforementioned analysis, the same car cannot present similar directions when captured by direction-opposite camera pairs. Thus, when two direction probability vectors are similar, the corresponding query and gallery are likely to be different vehicles and their feature distance should be pushed far away.

Due to the weird locations and angles of several cameras, *e.g.* camera 35, obvious ReID errors may occur in our experiments. For instance, two vehicles with different car types are matched as the same one with a high confidence. To alleviate such problem, a multi-task model with car type classification branch, as shown in Fig. 4, is trained on the attribute-extended training set of CityFlow-ReID (see section 4.1). Similar to direction constraint, all test images are predicted by the car type model to get the type probability vectors normalized by softmax function. The car type constraint can be formulated as follows:

$$d(i,j) = \begin{cases} d(i,j) + \sigma 4 & \langle t_i, t_j \rangle < \tau 3 \; and \; d(i,j) > \tau 4 \\ d(i,j) & othersize \end{cases}, \tag{8}$$

where t_i and t_j are type probability vectors of q_i and g_j, $\tau 3$ and $\tau 4$ are two thresholds in this constraint, and $\sigma 4$ is a

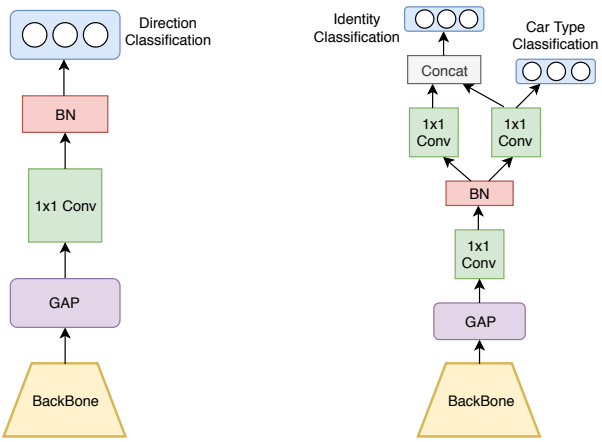

Figure 4. left: direction classification model. right: multi-task model with car type classification branch.

car type punish-item. We do not punish a query-gallery pair with a hard manner when their car types are different. This is because car type classification itself remains a difficult task in some particular camera angles.

Vehicles usually obey a certain temporal distribution when they move in a scenario. Temporal information can also be utilized as a strong constraint. Specifically, we use s_i and e_i to denote the start time and end time of the tracklet of q_i, and use s_j and e_j to denote the start time and end time of the tracklet of g_j. c_i and c_j indicate the camera Ids of q_i and g_j, respectively. Then, the temporal constraint can be formulated as follows.

$$d(i,j) = \begin{cases} d(i,j) + \sigma 5 \\ \quad min(e_i, e_j) + t(c_i, c_j) - max(si, sj) < 0 \\ d(i,j) \;\; othersize \end{cases}, \tag{9}$$

where $t(c_i, c_j)$ is a estimated time duration when a vehicle move between two cameras, $\sigma 5$ is also a punish-item. The intuition behind this temporal constraint is that the time difference must be less than an estimated time window when a vehicle moves from one camera to another.

Due to the large vehicle pose variations, the original appearance feature distance $d(i,j)$ may be inaccurate. In this paper, we propose to use extra group distance between q_i and g_j as well as the original $d(i,j)$ to represent the similarity between them. Combining the provided the tracklet information of CityFlow-ReID and the results of our multi-camera tracking, we can get the group split of all gallery images. Assume that g_j belongs to group D_k. The group distance $gd(i,j)$ between q_i and g_j is calculated as follows.

$$gd(i,j) = min\left\{d(i,j) \; g_j \in D_k\right\}. \tag{10}$$

The final feature distance $fd(i,j)$ between q_i and g_j is set as $fd(i,j) = d(i,j) + \alpha \cdot gd(i,j)$, where α is a weight parameter.

The reranking technique [35] has been proved effective in pedestrian ReID task. In this paper, we also use it to improve the performance of vehicle ReID. In our experiment, tracking, direction, car type and temporal constraint is first conducted on the original feature distances. After that, reranking operation is carried out. We find that several false positives reoccur after reranking. Therefore, constraints are applied again to exclude bad cases as much as possible. Eventually, the ranking list is obtained based on the final feature distance between each query and gallery image.

4. Experiments

4.1. Dataset for Vehicle ReID

Dataset for Training and Evaluation

In our initial experiments, the training set of CityFlow-ReID [27] is split into two parts for model selection and parameters learning, where the last 50 identities are used for evaluation and others are for training. We note that the final models are trained on a combined dataset containing the whole CityFlow-ReID with 333 identities and a fine-grained vehicle classification dataset Cars [13].

Dataset for Training Base Models

To improve the performance and robustness of our vehicle ReID models, we collect a large number of car images from the Internet and combine them with the VehicleID dataset [19] to establish a large vehicle ReID dataset with 30000+ identities. We first train our vehicle ReID models on this large dataset, after which finetuning is conducted on the aforementioned CityFlow-ReID+Cars dataset.

Refined Dataset by Our Detector

Since the images of CityFlow-ReID are padded with extra pixels, more background information is captured. To decrease the interference of background, we refine the images with tight bounding boxes generated by our own vehicle detector. Most models are trained on both the original images and refined images. For a certain model, the average features of its corresponding two versions are used for the final evaluation.

Extra Annotation on CityFlow-ReID

To exploit more information for the vehicle ReID task, we annotate the training set of CityFlow-ReID with extra directions and attributes. There are total eight directions including front, rear, left, left front, left rear, right, right front, right rear and five attributes including color, type, roof rack, sky window, logo. It should be noted that we find most attributes are not robust in the experiments and only the type attribute is used in the final solution.

4.2. Multi-Camera Tracking

The proposed method is submitted to NVIDIA AI City Challenge 2019 [28] for evaluation, and we achieve the rank #6 in multi-camera tracking task. In the final experiment, all δ parameters are set heuristically: $0.8, 2, 1, 0.6, 15$ and the weight parameters are tuned sequentially on training dataset in the order of appearance, location, velocity, direction, time until the best performance is achieved. We firstly tune appearance weight: W_a by setting all other weights to 0, when the best appearance weight is found, we it is fixed and then we search for the best location weight: W_l. During weight tuning all searched weights are fixed and all other weights are set to 0 except the one under tuning.

Table 1. Evaluation results of vehicle Reid on the CityFlow-ReID dataset [27].

Rank	Team	score
1	**Ours**	0.8554
2	team21	0.7917
3	team97	0.7589
4	team4	0.7560
5	team12	0.7302
6	team53	0.6793
7	team131	0.6091
8	team5	0.6078
9	team78	0.5862
10	team127	0.5827

4.3. Vehicle Re-identification

The proposed method is submitted to *NVIDIA AI City Challenge 2019* for evaluation, and we achieves the rank #1 in vehicle ReID task. In the final experiment, we concatenate the normalized features of all our models for feature distance calculation, which significantly outperforms each single model. The features are global feature of seresnext101, global feature of resnet101, global feature of resnet152, global feature of hrnet32, global feature of resnet101 with SAC, feature of resnet101 with MGN and keypoints guided feature of seresnext101, respectively. We also conduct k-recipcial reranking with the parameters set as $k1 = 50$, $k2 = 15$ and $\lambda = 0.5$. The reranking technique can make the ranking results even better. Finally, video-based multiple constraints are used to refine the reranking results, and each proposed constraint is proved effective by the evaluation system, and helps to improve the final performance to a much higher level.

There are totally 84 teams participating this task, the performances of the top10 teams are summarized in Tab. 1. It is seen that our method outperforms the other teams by a large margin, which demonstrates the superiority and effectiveness of our proposed method.

Some important parameters of the proposed method is listed in Tab. 2.

Several qualitative results of our baseline ReID models are shown in Fig. 5, from which we can see that many false positives similar to the query image are ranked at the top of the list. It is hard to filer out these samples merely by

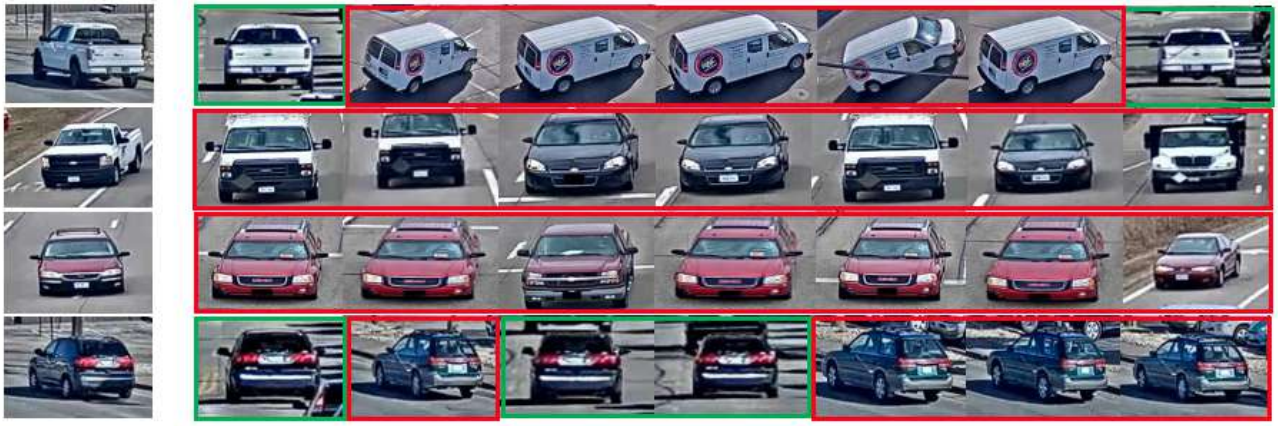

Figure 5. Qualitative results of our baseline ReID model (ResNet 50 trained by softmax cross entropy loss and triplet loss). The first column shows several different query images, each row demonstrates the top 7 gallery images founded by the model. The images in green box are true positives, and the remaining in red box are false positives.

Figure 6. Qualitative results of the proposed method. The query images are the same with Fig. 5.

Table 2. Parameters setting of the proposed method.

Parameter	Value
$\sigma 1$	0.5
$\sigma 2$	1.5
$\tau 1$	2.5
$\sigma 3$	5.0
$\tau 2$	0.03
$\sigma 4$	5.0
$\tau 3$	0.1
$\tau 4$	3.0
$\sigma 4$	2.5
$alpha$	1.0

image-based vehicle ReID models.

The comparison results of our method are demonstrated in Fig. 6. Benefiting from more powerful deep models and the proposed multiple constraints, most hard false positives can be excluded, which significantly improves the performance.

5. Conclusion

In this paper, we propose a novel multi-camera tracking and re-identification system. The tracking part integrates both appearance visual feature and temporal-spatial information. The vehicle orientation prediction is also utilized for the optimization of tracking loss function. In the re-identification part, several powerful ReID models are trained to extract visual features. A direction classification model and car type classification model are also explored to obtain the auxiliary information. To filter out hard false positives, multiple constraints are excavated including tracking, direction, car type and temporal limitations. To get a better representation of distance between two vehicles' features, the group distance is also proposed in this paper. Our method achieves the first place in vehicle re-identification at the *NVIDIA AI City Challenge 2019*, and outperforms other teams by a large margin.

References

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

[2] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S Davis. Soft-nms–improving object detection with one line of code. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5561–5569, 2017.

[3] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6154–6162, 2018.

[4] Chun-Te Chu, Jenq-Neng Hwang, Hung-I Pai, and Kung-Ming Lan. Tracking human under occlusion based on adaptive multiple kernels with projected gradients. *IEEE Transactions on Multimedia*, 15(7):1602–1615, 2013.

[5] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017.

[6] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004.

[7] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

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

[9] Haiyun Guo, Chaoyang Zhao, Zhiwei Liu, Jinqiao Wang, and Hanqing Lu. Learning coarse-to-fine structured feature embedding for vehicle re-identification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

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

[11] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[12] M. Jiang, Y. Yuan, and Q. Wang. Self-attention learning for person re-identification. In *Proc. BMVC*, 2018.

[13] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[14] Ratnesh Kumar, Edwin Weill, Farzin Aghdasi, and Parthsarathy Sriram. Vehicle re-identification: an efficient baseline using triplet embedding. *arXiv preprint arXiv:1901.01015*, 2019.

[15] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 734–750, 2018.

[16] Kuan-Hui Lee, Jenq-Neng Hwang, and Shih-I Chen. Model-based vehicle localization based on 3-d constrained multiple-kernel tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(1):38–50, 2015.

[17] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125.

[18] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[19] Hongye Liu, Yonghong Tian, Yaowei Wang, Lu Pang, and Tiejun Huang. Deep relative distance learning: Tell the difference between similar vehicles. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2167–2175, 2016.

[20] Hongye Liu, Yonghong Tian, Yaowei Yang, Lu Pang, and Tiejun Huang. Deep relative distance learning: Tell the difference between similar vehicles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2167–2175, 2016.

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

[22] Anton Milan, Stefan Roth, and Konrad Schindler. Continuous energy minimization for multitarget tracking. *IEEE transactions on pattern analysis and machine intelligence*, 36(1):58–72, 2014.

[23] Anton Milan, Konrad Schindler, and Stefan Roth. Multi-target tracking by discrete-continuous energy minimization. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):2054–2068, 2016.

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

[25] Jakub Sochor, Jakub Špaňhel, and Adam Herout. Boxcars: Improving fine-grained recognition of vehicles using 3-d bounding boxes in traffic surveillance. *IEEE Transactions on Intelligent Transportation Systems*, (99):1–12, 2018.

[26] Zheng Tang, Jenq-Neng Hwang, Yen-Shuo Lin, and Jen-Hui Chuang. Multiple-kernel adaptive segmentation and tracking (mast) for robust object tracking. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1115–1119. IEEE, 2016.

[27] Zheng Tang, Milind Naphade, Ming-Yu Liu, Xiaodong Yang, Stan Birchfield, Shuo Wang, Ratnesh Kumar, David C. Anastasiu, and Jenq-Neng Hwang. Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification. *CoRR*, abs/1903.09254, 2019.

[28] Zheng Tang, Milind Naphade, Ming-Yu Liu, Xiaodong Yang, Stan Birchfield, Shuo Wang, Ratnesh Kumar, David Anastasiu, and Jenq-Neng Hwang. Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking

and re-identification. *arXiv preprint arXiv:1903.09254*, 2019.

[29] Zheng Tang, Gaoang Wang, Tao Liu, Young-Gun Lee, Adwin Jahn, Xu Liu, Xiaodong He, and Jenq-Neng Hwang. Multiple-kernel based vehicle tracking using 3d deformable model and camera self-calibration. *arXiv preprint arXiv:1708.06831*, 2017.

[30] Zheng Tang, Gaoang Wang, Hao Xiao, Aotian Zheng, and Jenq-Neng Hwang. Single-camera and inter-camera vehicle tracking and 3d speed estimation based on fusion of visual and semantic features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 108–115, 2018.

[31] G. Wang, Y. Yuan, X. Chen, J. Li, and X. Zhou. Learning discriminative features with multiple granularities for person re-identification. In *Proc. ACM. MM*, pages 274–282, 2018.

[32] Zhongdao Wang, Luming Tang, Xihui Liu, Zhuliang Yao, Shuai Yi, Jing Shao, Junjie Yan, Shengjin Wang, Hongsheng Li, and Xiaogang Wang. Orientation invariant feature embedding and spatial temporal regularization for vehicle re-identification. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[33] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[34] Ke Yan, Yonghong Tian, Yaowei Wang, Wei Zeng, and Tiejun Huang. Exploiting multi-grain ranking constraints for precisely searching visually-similar vehicles. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 562–570, 2017.

[35] Zhun Zhong, Liang Zheng, Donglin Cao, and Shaozi Li. Re-ranking person re-identification with k-reciprocal encoding. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 3652–3661, 2017.

[36] Y Zhou, L Shao, and A Dhabi. Viewpoint-aware attentive multi-view inference for vehicle re-identification. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn*, volume 2, 2018.