TrackNet: A Triplet metric-based method for Multi-Target Multi-Camera Vehicle Tracking

Igor Ugarte Molinet*, Juan Antonio Rodríguez García*,
Francesc Net Barnés* and David Serrano Lozano*

Universitat Politecnica de Catalunya
Barcelona, Spain
igorugarte.cvm@gmail.com , juanantonio.rodriguez@upf.edu,
francescnet@gmail.com, 99d.serrano@gmail.com

Abstract

We present TrackNet, a method for Multi-Target Multi-Camera (MTMC) vehicle tracking from traffic video sequences. Cross-camera vehicle tracking has proved to be a challenging task due to perspective, scale and speed variance, as well occlusions and noise conditions. Our method is based on a modular approach that first detects vehicles frame-by-frame using Faster R-CNN, then tracks detections through single camera using Kalman filter, and finally matches tracks by a triplet metric learning strategy. We conduct experiments on TrackNet within the AI City Challenge framework, and present competitive IDF1 results of 0.4733. Our implementation is available at: https://github.com/mcv-m6-video/mcv-m6-2022-team2.

1. Introduction

The continuous growth of urban mobility in cities arises big challenges in the management and optimization of traffic flow. Thanks to the high availability of surveillance cameras and the annotated datasets of road traffic, the task of Multi-Target Multi-Camera vehicle tracking (MTMC) has gained momentum in the Computer Vision community. This task aims to track and identify vehicles along large traffic areas and different road intersections through different video cameras. Good solutions for this problem would allow traffic engineers to improve road design and traffic flow.

The task of MTMC has been always tackled in a multi-stage approach. A first stage is devoted to detect vehicles using object detection algorithms. The second stage performs a tracking of the detected objects in a single video sequence. Finally, the re-identification stage aims to identify the cars along all cameras.

In this paper we propose *TrackNet*, a modular method for multi-camera vehicle tracking based on Faster R-CNN for vehicle detection, Kalman filter for single-camera tracking, and triplet metric learning for cross-camera identification through learned feature similarity.

2. Related work

In this section we review methods in the recent literature for the different stages of our method.

2.1. Object Detection

Object detection is one of the classic and fundamental computer vision tasks, which given an input image the methods try to produce bounding boxes and classify each object. Two-stage methods handle region proposal and classification separately. For example, Mask R-CNN [2] and Faster R-CNN [6] employ Region Proposal Network (RPN) and an additional regressor to suggest bounding boxes and classify the objects inside the proposals. On the other hand, one-stage detectors handle the two tasks simultaneously making them much simpler and faster such as YOLO [5] or RetinaNet [3].

2.2. Single camera tracking

The goal of MTSC (Multi-target single-camera tracking) is to estimate the trajectory of the car through a sequence of frames. The majority of the methods follow the same procedure, solving the problem in two steps: first the compute the detections and then perform the tracking. For example, K. Shim *et al.* is an example where a detector (Mask R-CNN) and feature extractor to extract the characteristics of the objects (ResNeXt-50) is used [9]. Besides, the tracking is computed using Kalman filter and Hungarian Algorithm.

*Equal contribution

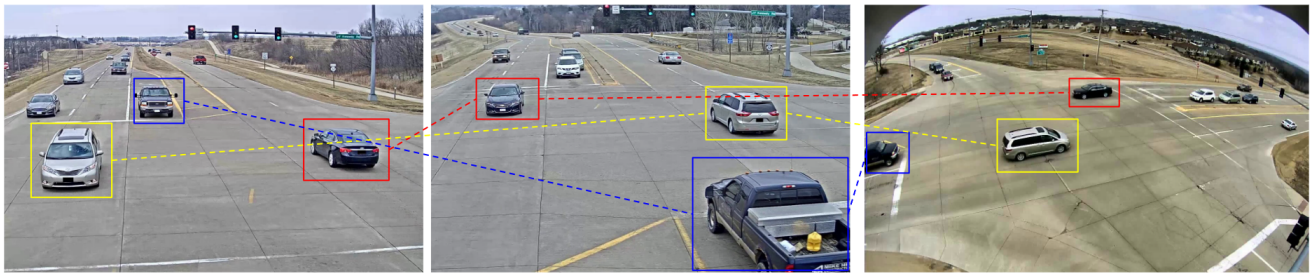

Figure 1. Overview of the MTMC task, where different vehicle trajectories are to be captured and matched along many cameras.

2.3. Multi camera tracking

Multi-camera tracking is a fundamental technique for traffic optimization. Most methods attempt to solve this challenge by first generating trajectories of the detected objects for each camera and then relating them between devices. For example, C. Liu *et al.*, the winners of track 3 of the 5th AI City Challenge [4], use a Direction Based Temporal Mask (DBTM) to match the similar vehicles once the tracking in every single camera has been produced. On the other hand, Ristani and Tomasi [8] proposed a method that did not require any manual annotation beyond the bounding boxes and ids using an adaptative weighted triplet loss to train a feature extraction network. For multi-camera tracking on vehicles the CityFlow [10] dataset has provided an important benchmark from city-scale traffic cameras.

3. Method

3.1. Vehicle detection

Our method follows the tracking-by-detection paradigm, similar to other state-of-the-art MTSC methods. That is, using an object detector to obtain bounding box locations of vehicles, and later computing the individual trajectories of the objects. Vehicle detection is the basis of both single and multi-camera tracking. Therefore, its effectiveness and reliability directly affects the performance of the entire system.

In this work, we evaluate three state-of-the-art detection algorithms: Mask R-CNN, RetinaNet and Faster R-CNN. In both of the three approaches we introduce raw input image frames that are processed by a Resnet50 backbone with Feature Pyramid Networks (FPN). The output is a set of bounding boxes and classes that correspond to the objects detected. Further filtering of vehicles is done in order to ignore classes of no interest.

3.2. Single-camera Tracking

Single camera tracking of objects follows three steps. First we ignore regions of the image that are not considered in the dataset. Second, we define an tracking algorithm to assign individual trajectories. Finally, we post-process the tracking by removing low-variant samples, such as parked cars.

Region of interest (RoI) filtering: In some cameras some cars appear outside the Region of Interest (ROI). To remove those detections, the Bounding Boxes are filtered by a camera mask provided in the Dataset (concretely the filtering is used under a distance threshold from the ROI border).

Vehicle tracking: Following the tracking-by-detection paradigm, the bounding boxes at each frame are assigned with a track id corresponding to a single trajectory. We explore two strategies, Kalman Filter and Maximum Overlap. The Maximum Overlap strategy assigns an existing track ID to a detection it overlaps by a $IoU_{0.5}$ threshold with a detection from the track in the past frame. If a detection does not overlap with any previous bounding boxes, a new track ID is assigned. Kalman filter works by propagating an object state into future frames using a linear constant velocity model, independent of other objects and camera motion.

We make use of SORT [1] and DeepSORT [11] implementations of the Kalman filter and the Hungarian Algorithm. In SORT, if a detection is associated to a target, the detected bounding box is used to update the target state where the velocity components are solved optimally by a Kalman filter. If no detection is associated to the target, linear velocity model is used. SORT handles correctly short-term occlusions, but is more unstable for long-term. In turn, DeepSORT integrates appearance information (spatial features) to SORT. The result is the combination, as a weighted sum, of the locations based on motion, which is useful for short term predictions, and appearance, which handles long term occlusions.

Variance filtering: In some cases parked cars are detected and assigned a track, although our task only considers cars that are in movement. To remove this samples, we come up with a way to detect tracks that are not moving. To that end, the centroid variance of each track is computed and if it is below a variance threshold, in the next frames it is removed.

3.3. Cross-camera tracking

We pose the task of identifying vehicles in different video sequences as a Metric Leaning problem, where we assume that images of the same car will generate similar spatial features. Learned appearance features are projected to a low dimensional space where simiralities can be imposed through a Triplet loss and identifiable clusters can be generated. The triplet loss is defined as,

$$L_{tri} = \sum_{i}^{N} \left[||f(x_i^a) - f(x_i^p)||_2^2 - ||f(x_i^a) - f(x_i^n)|| + \alpha \right], \quad (1)$$

where $f(x)$ is the inference function over the learned model, a, p, and n superscripts stand for anchor, positive and negative samples, and α is the margin. The margin allows to model the minimum distance allowed for negative pairs.

The target of the triplet is to generate representative features for each car, being distinctive between different vehicles and as similar as possible between the same one, even if the vehicle is taken from different cameras. However, the main drawback is that some detections are small because the car is far away from the capture point and the network has troubles in encapsulating the information. Therefore, in order to retain only the richest information from each single-camera track, each vehicle is sampled isochronously obtaining only five detections from the 30% part of the biggest bounding boxes.

The process of multi-camera tracklets re-ID is done sequentially by pairs, with the set of already re-ided tracks as first input and the corresponding camera as second. The re-id block matches two tracks if they are the most similar ones between them and they fulfill the cross-match condition. This condition ensures that a successful match i.e. the i-th track of camera A has the j-th in set B as the best match and vice-versa. By doing so, we make the system more robust against false matches.

4. Experimental results

4.1. Dataset and evaluation

The dataset used for both training and testing the proposed method is *CityFlowV2*, the one proposed for the AI City Challenge. This dataset is composed of video sequences acquired by 46 cameras in 16 different city intersections. There is a total of 313931 annotated bounding boxes for 880 distinct vehicles. The original dataset contains 6 different scenarios, 3 for training, 2 for validation and 1 for test. In our setting, we use only three sequences (1, 3 and 4), that are split in a two-fold cross-validation fashion. It shows challenging conditions such as parked cars, occlusions, or lack of bounding boxes when vehicles are at high distance.

Method	$AP_{0.5}$
Off-the-shelf Mask R-CNN	0.4952
Fine-tuned Mask R-CNN	-
Off-the-shelf RetinaNet	0.7269
Fine-tuned RetinaNet	0.9710
Off-the-shelf Faster R-CNN	0.4813
Fine-tuned Faster R-CNN	0.9852

Table 1. Vehicle Detection Results for $AP_{0.5}$, using off-the-shelf ImageNet-pretrained parameters and fine-tuning with CityFlowV2. We do not report Fine-tuned results for Mask R-CNN because CityFlowV2 does not offer masks.

To evaluate the method, the official challenge metrics are considered, namely IDF1, IDP, IDP, Detection Precision and Detection Recall, as described in [7]. The principal evaluation metric for MTMC is IDF1. We also make use of Average Precision (AP) to asses vehicle detectors.

4.2. Experiment setting

The proposed method is implemented in Pytorch 1.8.1, with the use of available libraries such as Detectron2 c for Object Detection, SORT c and DeepSORT c for object tracking. The Triplet Metric Learning procedure was adapted from x Github code. We trained the models using a single 3060 GPU during 5000 epochs. We use AdaGrad optimizer with LR of 0.001 and batch size of 16.

Experiments on vehicle detection: Table 1 presents $AP_{0.5}$ results on different architectures for vehicle detection. We explored RetinaNet, Mask R-CNN and Faster R-CNN with ResNeXt101 backbones, using the pretrained weights on the COCO dataset and fine-tuning with CityFlowV2. The fine-tuning is highly effective for all architectures , and Faster R-CNN outperforms the other methods. Even though time was not reported, we empirically observed that RetinaNet outperforms the other methods in inference speed, due to its one-stage design. This makes it a better choice in an online or production settings.

Experiments on Single-Camera Tracking: Table 2 presents IDF1 results and ablation study of the different tracking methods in the single-camera scenario. In all approaches we observe that pre and post processing steps of RoI Filtering (RF) and Variance Filtering (VF) improve results. We conclude that DeepSORT outperforms the others in mostly all cameras.

Experiments on Multi-camera Tracking: Table 3 presents the IDF1 results for the proposed *TrackNet* method. The results shown in each row correspond to using the sequence i as test and the other two as train. In the last row, the 3-fold cross-validated result is presented, and an IDF1 of 0.4733 is attained, which is competitive with the results

	Seq. 3							Seq. 1	Seq. 4
Camera id	c010	c011	c012	c013	c014	c015	**Avg**	**Avg**	**Avg**
Max. Overlap	0.4210	0.3039	0.0323	0.8781	0.3907	0.0042	0.3384	0.7653	0.5161
+ RF	0.9239	0.3039	0.0323	0.8781	0.6576	0.0861	0.4803	0.7912	0.6280
+ VF	0.9446	0.8214	0.2105	0.9092	0.8677	0.1349	**0.6480**	0.8334	0.7439
SORT	0.4334	0.3003	0.0327	0.6833	0.4010	0.0042	0.3229	0.7864	0.5338
+ RF	0.9358	0.3003	0.0327	0.7657	0.7160	0.0867	0.4729	0.8057	0.6363
+ VF	0.9401	0.6676	0.1426	0.7657	0.8672	0.1140	0.5692	0.8057	0.6642
Deep SORT	0.4180	0.3040	0.0324	0.9057	0.3920	0.0042	0.3427	0.8345	0.5716
+RF	0.9292	0.3040	0.0325	0.9057	0.6601	0.0890	0.4867	0.8594	0.6881
+VF	0.9327	0.8531	0.0915	0.8914	0.8710	0.1230	0.6271	**0.837**8	**0.7693**

Table 2. Results on IDF1 metric for the MTSC scenario. Three different settings are presented: First, using Sequence 3 as test and Sequence 1 and 4 for train. Second, using Sequence 1 for test and the other two for train. Finally, using Sequence 4 as test and the other two for train. For each method we show an ablation study, where the baseline is improved using ROI Filtering (RF) and Variance Filtering (VF). We show average IDF1 results for the three settings, and additionally the results for different cameras that recorded Sequence 3.

Seq.	IDF1	IDP	IDR	Prec.	Recall
Seq. 1	0.5268	0.5272	0.5262	0.8903	0.8887
Seq. 3	0.4259	0.3546	0.5330	0.6225	0.9358
Seq.4	0.4672	0.5220	0.4227	0.9298	0.7530
Avg	**0.4733**	**0.4679**	**0.4940**	**0.8142**	**0.8592**

Table 3. Results of the proposed method TrackNet based on DeepSort and Triplet Metric Learning.

of other works. Figure 2 shows qualitative results of our method, showing that the re-identification works good, but has some lacks in the bounding box detection due to the Kalman filter refinement.

Figure 2. Qualitative results of TrackNet. We observe the tracking and identification of a pickup truck at different cameras and timesteps.

5. Conclusion

In this work we present *TrackNet*, a method for the task of Multi-Target Multi-Camera Vehicle tracking, and evaluate it on the AI City Challenge framework using CityFlowV2 dataset. Our method is based on Faster R-CNN with Resnet50+FPN for vehicle detection and DeepSORT for tracking. We explore Triplet Metric Learning to learn feature projections that allow for clustering the vehicles across cameras. Our method proves to be effective in the task, showing a IDF1 score of 0.4733, and also giving good qualitative results. The overall performance is sensible to errors in the intermediate layers, hence improvements can be made by trying to enhance the individual stages. The main challenges that we encountered were with occlusions, illumination and noise conditions, high speed vehicles and very similar vehicles.

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

[2] 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. 1

[3] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer*

vision, pages 2980–2988, 2017. 1

[4] Milind Naphade, Shuo Wang, David C Anastasiu, Zheng Tang, Ming-Ching Chang, Xiaodong Yang, Yue Yao, Liang Zheng, Pranamesh Chakraborty, Christian E Lopez, et al. The 5th ai city challenge. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4263–4273, 2021. 2

[5] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 1

[6] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 1

[7] Ergys Ristani, Francesco Solera, Roger S. Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. 3

[8] Ergys Ristani and Carlo Tomasi. Features for multi-target multi-camera tracking and re-identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6036–6046, 2018. 2

[9] Kyujin Shim, Sungjoon Yoon, Kangwook Ko, and Changick Kim. Multi-target multi-camera vehicle tracking for city-scale traffic management. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4193–4200, 2021. 1

[10] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8797–8806, 2019. 2

[11] 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. 2