Single-Camera and Multi-Camera Vehicle Tracking for Road Traffic Monitoring

Eloi Bové, Jordi Burgués, Albert Jiménez and Arnau Roche
Universitat Politècnica de Catalunya
Barcelona, Spain
{eloi.bove, jordi.burgues.miro, albert.jimenez.tauste, arnau.roche}@estudiantat.upc.edu

Abstract

This paper presents a solution for Multi-Target Single-Camera (MTSC) and Multi-Target Multiple-Camera (MTMC) tracking of cars in the context of road traffic monitoring. Our tracking-by-detection algorithm uses detections from Mask-RCNN and a Kalman Filter approach to form the tracks. In the MTMC case, tracks are re-identified in different cameras using learned embeddings, which are pre-trained on the VeRi dataset and trained from scratch with our own training dataset. We evaluate our tracking results with the IDF1 metric and obtain a score of 0.56 in the MTSC case and 0.44 in the MTMC scenario. Code available here: https://github.com/mcv-m6-video/mcv-m6-2021-team5.

1. Motivation

Intelligent Transportation Systems (ITS) are becoming smarter as more data is available and as both researchers and industries develop high-quality models that unlock its potential. One specific challenge of ITS is the tracking of vehicles across multiple cameras. MTMC vehicle tracking consists in identifying vehicles, assigning them a unique ID and tracking them frame by frame, maintaining the same ID as they appear in the video captured by one or multiple cameras. The CVPR 2019 AI City Challenge [12][9], which aims to accelerate the research and development of ITS techniques, tackled such problem in one of its three tracks. Hence, we used a subset of the 2019 AI City Challenge dataset for Track 1 in order to develop a method for both single-camera and multi-camera tracking of cars in traffic surveillance videos.

2. Related work

2.1. Object Detection and Single-Camera Tracking

The *tracking-by-detection* approach consists in detecting objects at each frame to later arrange them and form tracks. This state-of-the-art solution needs to be based on reliable detections coming from models such as SSD512 [7] or YOLOv3 [11]. For single-camera tracking, some existing solutions use the Kalman Filter and the Hungarian Algorithm [1] [15] to match the detections, while other methods use custom losses that control smoothness, velocity or appearance based on a histogram-based representation [13].

2.2. Re-Identification and Multi-Camera Tracking

To scale a single-camera tracking algorithm to work across multiple non-overlapping cameras, it is vital to include a method that re-identifies vehicles when they appear in a different camera. Most re-identification methods are based on metric learning with various loss functions, such as the contrastive loss [2], triplet loss [4] or angular loss [14].

The winners of the MTMC tracking category in the CVPR 2019 AI City Challenge used deep features learned with a triplet loss after background removal and trajectory-based camera link models to solve the MTMC task [5].

3. Method

3.1. Multi-target single camera tracking

Several approaches for the single camera tracking have been tested, including the simplest maximum overlap tracking and the Kalman Filter tracking. These techniques will be detailed in this section. These two *tracking-by-detection* methods rely on having an object detector that serves as an input to the tracker. The quality of this object detection step is key for the tracking, hence some state of the art models have been tested for this purpose.

3.1.1 Object detection

Both single camera tracking methods make use of the intersection over union (IoU) to perform the data association step, matching the incoming frame instances to the tracks. This causes the tracking to be sensitive to the input detections that are given. Three different object detection models were tested: SSD-512, YOLOv3 and Mask-RCNN [3].

These models are all trained on the COCO dataset [6] and offer similar performance. However, Mask-RCNN outputs segmentation masks for the targeted objects and may have an advantage since bounding box accuracy tends to improve in such cases.

3.1.2 Maximum overlap tracking

This tracking method consists of matching detections between consecutive frames based on the IoU. More specifically, assign each detection in frame N to the tracked object of frame $N-1$ that registers the highest IoU, while also tracking new detections. This method is simple and easy to implement, while giving decent results. Nonetheless, it comes with some limitations such as the lack of memory, that is, the ability to recover a track after a missed detection, the inability to track fast moving objects and the issues with occlusions.

3.1.3 Kalman Filter tracking

The Kalman filter tracking takes as a baseline the maximum overlap, since it uses the IoU to match detections to tracks. The tracking implementation corresponds to [1]. In this method, each track is modeled using Equation 1.

$$\mathbf{x} = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]^T \tag{1}$$

That is, using the object position in pixels (u, v), the object area s and the aspect ratio r. The derivative of the position, $(\dot{u}, \dot{v})$ is the velocity of the target, while $\dot{s}$ is the rate of change of the area. These state variables are used to predict the actual velocity of the target using the Kalman filter framework, helping the tracks to "stay alive" if no detections are present for a few frames, and also helping when occlusions happen. Other relevant parameters for this method are:

- **Minimum hits**. Minimum number of appearances for an object to be considered a track. This parameter allows filtering spurious detections. This comes with a trade-off because if it's too high, objects that appear only for a few frames will be missed.

- **Maximum age**. Maximum number of frames to keep alive a track without associated detections. That is, the lifetime of a track.

- **IoU threshold**. Minimum amount of overlap for a detection to be associated with a track. Should be high to filter outliers in the data association, but if it's too high this will penalize bad estimations from the filter, that can occur if the velocity of the target is not constant.

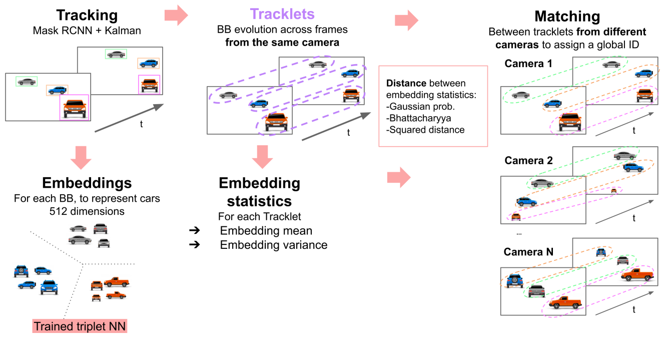

Figure 1. **Multi-target multi-camera tracking system**: MTSC tracking is used to extract tracklets (car BB evolution along time) of all the cars, as well as to compute representative embeddings for all the views of the different cars (see Section 3.2.1). This embeddings are used to add information at tracklet level: each tracklet has an embedding mean and variance (statistics) that represent each of the cars with all their possible points of view. Therefore, the problem becomes a *tracklet matching problem*, where different tracklets from different cameras are compared thanks to its embedding statistics (see Section 3.2.2)

3.2. Multi-target multi-camera tracking

This section details the techniques and implemented algorithms to tackle the Multi-target multi-camera problem comprising the tracking and identification of cars throughout different camera sequences. As shown in Figure 1, the system takes as a basis the MTSC tracking (see Section 3.1) to extract embeddings for all the BBs as well as to compute tracklets for each car that appears on the cameras. The embeddings are computed using a Triplet Network and are used to characterize each tracklet with a mean and a variance. Then, this representative statistics are exploited to compare all the tracklets between cameras and decide which ones are related to the same car.

3.2.1 Embedding computation

In order to address this problem we have used two models to obtain car representations: a trained model with the VeRi dataset [8], and a model trained from scratch on our own dataset:

- **VeRi dataset**. contains over 50,000 images belonging to 776 vehicles captured from 20 cameras. The recordings comprise 24 hour shots in different scenes.
- **Our own dataset**. formed by car crops from cars obtained from the AI City data. We have gathered over 44,000 images belonging to 184 cars. This set contains views from 36 different cameras and recordings of 3-4 minutes in nearby scenes.

Both models use a triplet network to learn feature vectors per every car (embeddings) that are compared afterwards

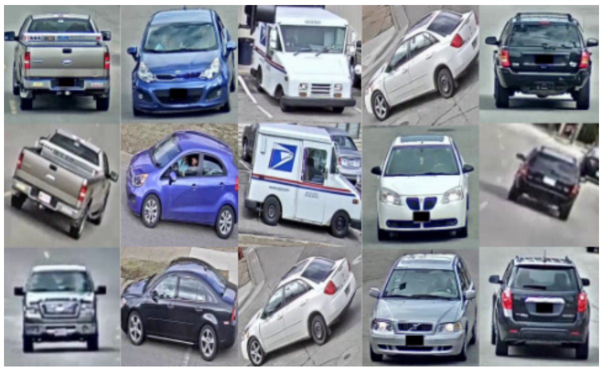

Figure 2. **Triplets of images** used to train a Triplet Network: anchor (top), positive (middle) and negative (bottom)

for ID matching and re identification. Regarding VeRi we have used a model that is already trained [10] and performed inference to obtain the desired embeddings. With respect to our own dataset, we have trained from scratch with sequences 1 and 4 from AI City data and validated with sequence 3 using a triplet network with a Resnet-18 architecture and the following configuration: a training (sequences 01 and 04) - validation (sequence 03) split of 85%-15% , a learning rate of 10^{-3} with decay scheduler, 46 epochs, a batch size of 32 and a SGD optimizer.

The Triplet Network is trained using triplets of images composed by an anchor, a positive and a negative sample (Figure 2). With these data the model learns the embeddings such that the distance between the anchor and the positive sample *d(a,p)* is small, and the distance between the anchor and the negative sample *d(a,n)* is big. This is known as a minimization of the triplet loss (see Equation 2).

$$\mathcal{L} = max(d(a,p) - d(a,n) + margin, 0) \quad (2)$$

With the pre-trained model with VeRi and our trained model, we compute the car embeddings in a 512-dimensional space. For visualization purposes, we perform t-Distributed Stochastic Neighbor Embedding (t-SNE) for dimensionality reduction. In Figure 3 it can be seen the different resulting clusters from the embedding computation using the aforementioned models.

3.2.2 Matching algorithm

Thanks to the embedding computation, each tracklet has a representative descriptor for all the car views. Before starting to match the tracklets, we detect and filter out the ones that are related to parked cars to avoid wrong re-identifications. This is done computing the median of the squared differences of consecutive object bounding box center points: if the median is low, we consider the car as parked.

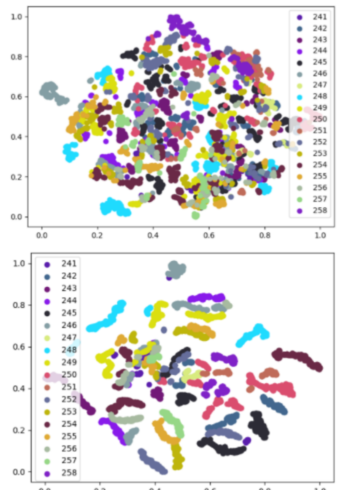

Figure 3. **Car embeddings**: VeRi model (top) and our model (bottom)

In order to ensure an online system, only tracklets up to the same (or earlier) timestamp can be considered for the comparison. In order to do so, we consider two lists of tracklets: **untracked tracklets**, with independent tracklets without ID, and **tracked tracklets**, with two or more tracklets with a common ID.

Untracked and tracked lists are updated as soon as new tracklets arrive. The statistics are used to compute a distance between the tracklet to classify and all the tracklets in the lists: if there is enough similarity (different thresholds are used according to the selected distance metric) the two tracklets are considered to be associated to the same vehicle. It is important to point out that a new tracklet can be matched to another from the untracked or tracked list indistinctly if the distance is small enough:

- If the distance is smaller with an untracked tracklet, both of them are considered a new car and they are moved to the tracked list with a new ID.
- If the distance is smaller with a tracked tracklet, the new one is considered as an extra view of the car and it is added to the tracked list with the existent ID.
- If the distance is not small enough, the tracklet is added to the untracked list

At the end, untracked tracklets are discarded, as they are considered cars that only appear in one camera.

4. Evaluation

4.1. Multi-target single camera tracking

The methods described in Section 3.1 are evaluated using the sequence 03 of the AICity Challenge dataset. This

Table 1. **MTSC quantitative results**: results for all the detection and tracking methods. The best results are provided by the Kalman tracking with Mask R-CNN detections

		Camera						IDF1	
Detector	**Tracking**	**c010**	**c011**	**c012**	**c013**	**c014**	**c015**	**Avg IDF1**	**Weighted avg IDF1**
SSD-512	Overlap	0.3422	0.3737	0.0863	0.4489	0.3889	**1.0**	0.4400	0.3292
YOLO3		0.4361	0.2284	0.0442	0.2416	0.5091	0.5925	0.3420	0.2923
Mask RCNN		0.4920	0.1810	0.0701	**0.6643**	0.5709	**1.0**	0.4964	0.3998
SSD-512	Kalman	0.4326	0.3003	**0.1422**	0.3175	0.3260	**1.0**	0.4197	0.3023
YOLO3		0.6397	0.3101	0.0257	0.2170	**0.6477**	0.6666	0.4178	0.3660
Mask RCNN		**0.6461**	0.3470	0.1097	0.6607	0.6029	**1.0**	**0.5611**	**0.4738**
	Num. frames	1923	2031	2156	2185	2275	16		

Table 2. **MTMC quantitative results**: our model outperforms the pretrained with VeRi dataset (all results are with squared distance)

Model	**IDF1**	**IDP**	**IDR**	**Precision (detection)**	**Recall (detection)**
Our model	**0.4363**	0.4937	**0.3909**	0.7878	**0.6237**
VeRi model	0.3853	**0.4973**	0.3190	**0.8182**	0.5367

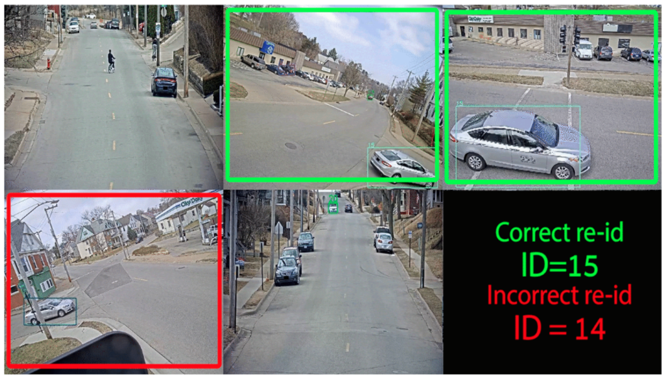

Figure 4. **MTMC qualitative results**: a car is correctly re-identified in two cameras (green), but a new ID is wrongly assigned in a third one (red), probably because of a bad BB crop (affected by an electricity street pole)

sequence contains data from 6 different cameras, and offers ground truth annotations for vehicles that appear in at least two cameras. Parked cars and other vehicles that are not visible from other cameras will appear as false positives, even if they are properly tracked. However, this evaluation will be used to choose the baseline method for the MTMC case, that will compute the initial tracklets. According to the results of Table 1, the single camera tracker that offers the best performance is the Kalman filter with Mask-RCNN detections, since it provides best weighted average IDF1 score across all the cameras, using the number of frames.

4.2. Multi-target multi-camera tracking

The pipeline presented in Section 3.2 shows the complexities of developing an algorithm to track cars from different views. In terms of qualitative results, it can be appreciated both matches and mismatches between the same cars occur, as depicted in Figure 4. This is highly dependent on the conditions in which the different cameras are offering information: more distorted or ambiguous views will lead to a wrong re identification, whereas clear and distinguishable objects will produce better results.
All in all, as it is shown in Table 2, our trained model produces better results than the pretrained one with the VeRi dataset, probably because the data is more specific to our scenario. While VeRi contains a wide variety of scenes, our own dataset is bounded to a very similar context. Since sequence 03 from AI City data is being used, the characteristics of test data resembles in a high degree those of the training data. Besides, the first model trained with VeRi has a better precision whereas our model offers a better recall.

5. Conclusions

In this paper we propose a solution for single-camera and multi-camera tracking of cars in urban scenes. In the single-camera scenario we achieve an IDF1 of 0.56 and we identify the tracking of transversal car movements to be the most frequent failure case. On the other hand, when evaluating our solution in the multi-camera setting we achieve an IDF1 score of 0.44. We are able to improve the quality of the car embeddings pre-trained on the VeRi dataset by fine-tuning with our own dataset, providing some success cases of re-identification. However, re-identifying vehicles in a non-overlapping camera setup is a challenging task and our algorithm is not always able to do it successfully.

As future work, both MTSC and MTMC results could be improved by making our tracking algorithm more robust to missed deductions by, for example, interpolating the missing bounding boxes. Also, more spatial and temporal constraints could be applied to exploit information such as car trajectories, time of appearance or camera disposition, even though some of these are very specific to a tracking scenario and may not generalize well.

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. 1, 2

[2] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006. 1

[3] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 1

[4] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International workshop on similarity-based pattern recognition*, pages 84–92. Springer, 2015. 1

[5] Hung-Min Hsu, Tsung-Wei Huang, Gaoang Wang, Jiarui Cai, Zhichao Lei, and Jenq-Neng Hwang. Multi-camera tracking of vehicles based on deep features re-id and trajectory-based camera link models. In *CVPR Workshops*, pages 416–424, 2019. 1

[6] 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. 2

[7] 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. 1

[8] Xinchen Liu, Wu Liu, Huadong Ma, and Huiyuan Fu. Large-scale vehicle re-identification in urban surveillance videos. In *2016 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2016. 2

[9] Milind Naphade, Zheng Tang, Ming-Ching Chang, David C Anastasiu, Anuj Sharma, Rama Chellappa, Shuo Wang, Pranamesh Chakraborty, Tingting Huang, Jenq-Neng Hwang, et al. The 2019 ai city challenge. In *CVPR Workshops*, pages 452–460, 2019. 1

[10] R. P. Pranoy Radhakrishnan. Vehicle re identification using triplet networks. `https://github.com/pranoyr/vehicle-reid`, 2021. 3

[11] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 1

[12] 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. 1

[13] 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. 1

[14] Jian Wang, Feng Zhou, Shilei Wen, Xiao Liu, and Yuanqing Lin. Deep metric learning with angular loss. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2593–2601, 2017. 1

[15] 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. 1