Video Surveillance for Road Traffic Monitoring

Daniel Azemar i Carnicero
Universitat Autònoma de Barcelona
daniel.azemar@e-campus.uab.cat

Richard Segovia Barreales
Universitat Autònoma de Barcelona
richard.segovia@e-campus.uab.cat

Sergi Solà i Casas
Universitat Autònoma de Barcelona
sergi.solac@e-campus.uab.cat

Sergio Casas Pastor
Universitat Autònoma de Barcelona
sergio.casas@e-campus.uab.cat

Abstract

Video surveillance for road traffic has a wide range of applications, from statistical traffic analysis to accident warning systems and many other security systems. The emergence of challenges like the AI City Challenge, increased further the interest in these systems. In this project we implemented a video surveillance system that aims to solve the Multi-Target Single-Camera (MTSC) and the Multi-Target Multi-Camera (MTMC) tasks from the AI City Challenge. For the MTSC task we use a Retina net and a Faster R-CNN to get the detections and we propose 3 different techniques to solve the tracking, thus including constant velocity filtering and optical flow averaging. In the MTMC, to get the features we use a ResNet-50 retrained using a triplet loss approach and we propose a voting system to relate the tracks.

1. Introduction

CityFlow is a city-scale traffic camera dataset consisting of more than 3 hours of HD videos from 40 cameras across 10 intersections. The dataset contains more than 200K annotated bounding boxes covering a wide range of scenes, viewing angles, vehicle models, and urban traffic flow conditions. Camera geometry and calibration information are provided to aid spatio-temporal analysis. The AI City Challenge consists on tracking cars on the CityFlow dataset. To achieve this goal, we have to address three related research problems: 1) Detection and tracking of targets within a single camera, known as Multi-Target Single-Camera (MTSC) tracking; 2) Re-identification of targets across multiple cameras, known as ReID; and 3) Detection and tracking of targets across a network of cameras, known as Multi-Target Multi-Camera (MTMC) tracking. MTMC tracking can be regarded as the combination of MTSC tracking within cameras and image-based ReID with spatio-temporal information to connect target trajectories between cameras.

2. Related Work

We briefly explain the previous state-of-the-art methods based on CityFlow benchmark [1].

ReID problem

For the vehicle ReID problem, the recent work explores the advances in batch-based sampling for triplet embedding. Another state-of-the-art vehicle ReID method is the winner of the vehicle ReID track in the AI City Challenge Workshop at CVPR 2018 [2], which is based on fusing visual and semantic features (FVS).

MTSC tracking problem

Most state-of-the-art MTSC tracking methods follow the tracking-by-detection paradigm. They first generate detected bounding boxes using well-known methods such as YOLOv3 [3], SSD512 [4] and Faster R-CNN [6]. For all detectors, we use default models pre-trained on the COCO benchmark, where the classes of interest include car, truck and bus.

The most important methods used for MTSC tracking are the following: DeepSORT [7] is an online method that combines deep learning features with Kalman-filter-based tracking and the Hungarian algorithm for data association. TC [8] is an offline method that won the traffic flow analysis task in the AI City Challenge Workshop at CVPR 2018 [2] by applying tracklet clustering through optimizing a weighted combination of cost functions. Finally, MOANA [9] is another online method that achieves state-of-the-art performance employing similar schemes for

spatio-temporal data association, but using an adaptive appearance model to resolve occlusion and grouping of objects.

MTMC tracking problem

In the case of MTMC, there are several approaches for spatio-temporal association. In Progressive and multimodal Vehicle ReID framework (PROVID), a spatio-temporal-based re-ranking scheme is employed. The spatio-temporal similarity is measured by computing the ratios of time difference and physical distance across cameras. Another method, called two-way Gaussian mixture model features (2WGMMF), is based on learning the transition time between camera views using Gaussian distributions. Finally, in FVS, the temporal distribution is pre-defined based on the estimated distance between cameras.

In Tables 1 and 2, we observe the performance of the state-of-the-art methods for MTSC tracking and MTMC tracking, respectively.

3. Multi-Target Single-Camera

The first part of the problem is to achieve detection and tracking of targets using each individual camera independently. The problem is known as Multi-Target Single-Camera (MTSC). For the detection of the cars on each camera, we use two different detection networks, RetinaNet [5] and Faster R-CNN [6]. We use this two networks to evaluate the impact on using one-stage (RetinaNet) versus two-stage (Faster R-CNN) detectors. We fine tune these two detection networks using Detectron2 framework and some of the sequences available for the 2020 AI City Challenge. Concretely, we use two trained models: one trained with sequences 1 and 4 (and sequence 3 for testing) and another one with sequences 3 and 4 for training (and sequence 1 for testing).

3.1. Detection pre-processing

The annotated bounding boxes in CityFlow dataset contains only the objects that are visible from several cameras at the same time. For this reason, a pre-processing of the detected bounding boxes, before applying tracking techniques, is necessary. In this step, we remove the following detections that are not present in the annotations:

- **Parked (static) cars.** We remove them by calculating the Intersection over Union (IoU) of each frame with respect to the first frame (where the parked cars are already present). If the IoU is greater than 0.5, these detected bounding boxes are removed.
- **Small and big cars.** Bounding boxes with a width or height higher than 40 % of the width or height of the image, or lower than 10 % of the width or height of the image, are removed from the detections.
- **Repeated bounding boxes are removed.** We compute the IoU between each bounding box of a certain frame with all the other ones from the same frame and remove it if for some bounding box we get an IoU value close to 1.

3.2. Tracking methods

In this project, we use three different tracking techniques and we compare their performance on MTSC. Finally, we select only the best one to work with it on the second problem.

3.2.1 Tracking by overlap

This method consists on comparing the current frame detections with the detected bounding boxes from the previous 5 frames. For each of the current detected bounding boxes, we compute the IoU with all the previous bounding boxes. Then we go from the detection that reaches a higher IoU to the last one. If the maximum IoU of this box is greater than 0.5, this bounding box is assigned the same track as the box that performs that IoU. If there is no maximum IoU that surpasses the value of 0.5, a new track ID is assigned to this bounding box. We use 5 previous frames, to cover the case when a bounding box is occluded in some frames but appears again. In this case, we want to keep the same track when the object appears at the next frames.

3.2.2 Kalman filter

This method consists on adding a Kalman filter to the overlap tracking method explained before. This was done using an implementation of SORT method on GitHub [10]. The addition of the Kalman filter helps to improve the tracking results, specially in the cases were the object get occluded and gets its tracking lost. However, the results can get worse for detection if it differs form the predicted shape or speed.

3.2.3 Optical flow

The optical flow of a pair of frames describes the apparent motion of objects from one of the images to the other. The estimation of the optical flow between two consecutive frames can be used to improve the tracking precision for the overlap tracking method that we explained before as this allow us to estimate an expected position for each bounding box. For this optical flow method, we use Farnebäck [11] optical flow implementation that is included in the OpenCV library.

The developed method is the following: first we compute the optical flow between the previous frame and the

current one. Once we have this, for each bounding box in the previous frame, we crop the equivalent rectangle from the optical flow image and using the motion vectors from it we calculate the expected displacement. These predicted bounding box are then used as input for the overlap method to compare with the bounding boxes predicted by the neural network.

As optical flow prediction can contain errors, we alternatively compute the mean between the previous bounding boxes and the ones predicted from the optical flow.

3.3. Results

In Tables 3 and 4 we can see the obtained IDF1 values for the different cameras of each sequence (3 and 1, respectively) and its average IDF1 value. From these tables we observe how results depend hugely on how the cameras are located and oriented in the scene. On cameras where objects are too close, they cross to fast and the cars are more difficult to track. On the other hand, if the camera is too far from the road, the detected cars will have too small bounding boxes and so they are removed in the pre-processing step.

Having into account that, we can see how the models that uses the RetinaNet detector seem to provide better results, although the best results for the sequence 3 are obtained using the Faster R-CNN with the overlap+optical flow tracking technique. Moreover, we observe how adding the Kalman filter to the overlap tracking technique seems to improve the results when we use the Faster R-CNN detector, while it seems to worsens the results when we use the RetinaNet detector. Nevertheless, the use of the optical flow estimations to improve the tracking, does not seem to provide any significant improvement.

If we compare our results with the results of the state-the-art in Table 1, we can see that our results are quite lower. One of the reasons why it differs is because the ground truth only contains the cars that are present in at least two cameras, while we detect and track all the cars present in the videos. In addition, sequences are very different between them, and we are only training the model with two of them, instead of three like in the actual challenge, and testing with different sequences. We think that using more sequences to train the models could improve the obtained results.

4. Multi-Target Multiple-Camera

Once we have achieved the problem of Multi-Target on Single Camera (MTSC), we want to achieve good performance on tracking the detected cars in a multi-camera set up. The main aim of this problem is re-identifying the detected and tracked cars on each camera to determine the tracking in the whole multi-camera set up.

4.1. Input data

In this project, the MTMC task is performed using the detection and tracking information of each camera obtained from the MTSC task. To do so, we crop and save the bounding boxes of the different detected cars using the ground truth information, which will be the input data of our ReID method, and we save the IDs of each tracklet obtained for each of the cameras, used for track matching and to determine the relation between cameras.

4.2. ReID method

As we just explained, we use the predictions from MTSC as starting point for our ReID method. The main idea behind the method is to extract some features from the bounding boxes that allows us to relate the different tracks. To do so, we train a Triplet Network based on three ResNet-50 with shared weights between them, pre-trained on ImageNet, to extract embeddings describing the cropped bounding boxes. The embeddings are 2048-dimensional and the network is trained using the triplet loss as loss function. The model was extracted from an already implemented ReID framework that can be found on GitHub [12].

The network is trained using an Adam optimizer with a learning rate of 0.0003, with a single step scheduler with an step size of 20. For the loss function, we used a weight of 0.7 and 1 for the triplet and softmax terms, respectively, also adding a margin of 0.3 to get more reliability on the results. The whole training is done in 10 epochs with a batch size of 32. Some data augmentation techniques such as random flip or random crop are also applied.

4.2.1 Triplet Loss

To train our ReID method we use the Triplet Loss, which is the most used loss function in triplet networks. The Triplet Loss is a function that compares an anchor input to a positive and a negative input. The main aim of this function is minimize the distance between the anchor and the positive inputs, while maximizing the distance between the anchor and the negative inputs. This leads the Triplet Loss to be one of the most suitable loss functions for metric learning and similarity learning.

4.2.2 Tracks matching

The idea behind track matching is to relate tracks from two cameras using some kind of distinctive characteristic or feature. Our implementation first extracts features from each bounding box in each track using the neural network explained in 4.2.1. Once we have the features, we then compute a distance matrix between all the bounding boxes from one camera to the other. This distance matrix contains

the euclidean distance between the features of the bounding boxes. Moreover this matrix is built grouping together bounding boxes that belong to the same track. This allows us to ease computations as we know where tracks start and end. Then we compute the minimum value for each row of the distance matrix. By doing this, we obtain for each bounding box in the first camera, the most similar bounding box in the second camera. Next we count all the minimum values that are located at each track intersection, giving us the amount of votes. For each row we keep the most voted intersection, and if this intersection has a minimum percentage of votes tracks are assumed to be related and the same ID will be assigned to both of them. If the number of votes is not over the threshold, that track is assumed to be unique and a new ID is assigned to it. For a visual explanation, please check Figure 1.

4.2.3 Relating multiple cameras

The methods explained before allow us to relate tracks between two cameras, but sequences have more than two cameras. To cope with this situation, we develop two different approaches:

- **Reference camera.** In this method we select one of the cameras as a reference camera and, using the previously explained methods, we relate the tracks between the reference camera and each one of the other cameras in the sequence. Tracks which do not have a strong relation with any track from the reference camera are given a new ID.

- **Merge camera features.** Similar to the previous method, using the extracted features for each bounding box, we relate the tracks between two cameras. Once we have the tracks related, we group together bounding boxes features that are from the same object and we use these groups as input to compute the relation between them and the set of tracks from the new camera. This process is repeated for all the remaining cameras.

4.3. Results

In Table 5, the obtained values for the evaluation metrics for the MTMC methods for the two sequences used to test the model are presented. From this table we can see that our results are comparable, or even better, to the ones obtained from the current state-of-the-art methods presented in section 2. However, as in the case of MTSC, the obtained results vary hugely depending on the sequence we use to test. Therefore, training the model with more sequences should help to improve the results or even to better select the voting threshold for the sequences, as selecting this value accurately helps on improving the results.

In figure 2, the obtained cars classified by the model to be the same car are presented for the first three tracks. From this figure, it can be seen how the model can correctly identify the similarity between cars even if the orientation differs significantly or if the object is partially occluded. However, some cars can be misclassified even differing in the color of the car, although in some cases the error is due to more subtle changes, such as the third car from the third track, whose differences with respect to the correct car are even difficult to see for the human eye.

5. Conclusions

In this project, we have proposed a solution to address the AI City Challenge consisting on the MTSC tracking, ReID and MTMC tracking. The overall conclusions we can extract from this work are the following:

- For the MTSC tracking problem, the main source of problems is the detection. The detection performance is highly affected by camera positioning and orientation, so pre-processing of the bounding boxes is key for a good performance. If the detections are correct, using kalman filter and optical flow provides an improvement in the tracking with respect to the single overlap method.

- The ReID method we propose works well even with very different viewing angles. However, the proposed method has some failures specially using the cameras with lower performance in the MTSC tracking problem.

- Some advantages of our method are that it does not need perfectly synchronized cameras. Moreover, we observe that triplet loss is a great tool to tweak an object classifier network onto a ReID tool, generating an easy-to-understand embedding.

References

[1] Zheng Tang, Milind Naphade, Ming-Yu Liu, Xiaodong Yang, Stan Birchfield, Shuo Wang, Ratnesh Kumar, David Anastasiu, Jenq-Neng Hwang. 2019. CityFlow: A City-Scale Benchmark for Multi-Target Multi-Camera Vehicle Tracking and Re-Identification. *arXiv:1903.09254v4*

[2] Milind Naphade, Ming-Ching Chang, Anuj Sharma, David C. Anastasiu, Vamsi Jagarlamudi, Pranamesh Chakraborty, Tingting Huang, Shuo Wang, Ming-Yu Liu, Rama Chellappa, Jenq-Neng Hwang, and Siwei Lyu. 2018. The 2018 NVIDIA AI City Challenge. *In-Proc. CVPR Work-shops*

[3] Joseph Redmon and Ali Farhadi. 2018. YOLOv3: An incremental improvement. *arXiv:1804.02767*

[4] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C.Berg. 2016. SSD: Single shot multibox detector. *InProc. ECCV*

[5] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollár. 2018. Focal Loss for Dense Object Detection. *arXiv:1708.02002v2*

[6] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, Serge Belongie. 2017. Feature Pyramid Networks for Object Detection. *arXiv:1612.03144v2*

[7] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. 2017. Simple online and real time tracking with a deep association metric. *InProc. ICIP*

[8] Zheng Tang, Gaoang Wang, Hao Xiao, Aotian Zheng, and Jenq-Neng Hwang. 2018. Single-camera and inter-camera vehicle tracking and 3D speed estimation based on fusion of visual and semantic features. *InProc. CVPR Workshops*

[9] Zheng Tang and Jenq-Neng Hwang. 2019. MOANA: An online learned adaptive appearance model for robust multiple object tracking in 3D. *IEEE Access, 7(1):31934–31945*

[10] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, Ben Upcroft. 2016. Simple Online and Realtime Tracking. *arXiv:1602.00763*

[11] Gunnar Farnebäck. 2003. Two-Frame Motion Estimation Based on Polynomial Expansion. *Computer Vision Laboratory, Linköping University*

[12] Zhou, Kaiyang and Xiang, Tao. 2019. Torchreid: A Library for Deep Learning Person Re-Identification in Pytorch. *arXiv preprint arXiv:1910.10093*

A. Tables and Figures

Method	IDF1	MOTA
DeepSORT+YOLO	78.9%	67.4%
DeepSORT+SSD	79.5%	68.9%
DeepSORT+FRCNN	78.9%	66.7%
TC+YOLO	79.1%	68.0%
TC+SSD	79.7%	70.3%
TC+FRCNN	78.7%	68.4%
MOANA+YOLO	77.8%	68.6%
MOANA+SSD	72.8%	67.0%
MOANA+FRCNN	75.6%	68.6%

Table 1: State-of-the-art methods for MTSC tracking and object detection on CityFlow

Method	Best IDF1
PROVID+DeepSORT	35.6%
PROVID+TC	40.6%
PROVID+MOANA	34.4%
2WGMMF+DeepSORT	40.3%
2WGMMF+TC	45.1%
2WGMMF+MOANA	38.1%
FVS+DeepSORT	41.4%
FVS+TC	46.3%
FVS+MOANA	39.5%

Table 2: State-of-the-art methods for MTMC tracking on CityFlow. Best IDF1 based on different image-based ReID

Model	c10	c11	c12	c13	c14	c15	Avg. IDF1
RetinaNet + Overlap	0.8863	0.5887	0.4067	0.5577	0.8256	0.0404	0.5509
RetinaNet + Kalman	0.8782	0.5515	0.4525	0.5918	0.8238	0.0382	0.556
RetinaNet + Optical Flow	0.8863	0.5887	0.4067	0.5577	0.8256	0.0404	0.5509
Faster R-CNN + Overlap	0.7776	0.5362	0.2701	0.5343	0.8007	0.0237	0.4904
Faster R-CNN + Kalman	0.7530	0.5420	0.3112	0.5701	0.7900	0.0219	0.4980
Faster R-CNN + Optical Flow	0.9349	0.5777	0.3759	0.5740	0.8324	0.0705	0.5609

Table 3: Obtained IDF1 values obtained for each camera of the sequence 03 of the dataset.

Model	c10	c11	c12	c13	c14	Avg. IDF1
RetinaNet + Overlap	0.7843	0.6770	0.6578	0.7381	0.5486	0.6812
RetinaNet + Kalman	0.8154	0.7063	0.7161	0.8257	0.5837	0.7294
RetinaNet + Optical Flow	0.7843	0.6770	0.6578	0.7381	0.5486	0.6812
Faster R-CNN + Overlap	0.7669	0.6906	0.6441	0.7953	0.5302	0.6854
Faster R-CNN + Kalman	0.7970	0.7160	0.6965	0.8105	0.5569	0.7154
Faster R-CNN + Optical Flow	0.7669	0.6944	0.6452	0.7963	0.5307	0.6863

Table 4: Obtained IDF1 values obtained for each camera of the sequence 01 of the dataset.

Sequence	Thres.	IDF1	IDP	Precision	Recall
Sequence 01	0.7	0.4530	0.5093	0.8943	0.7163
Sequence 03	0	0.4824	0.5030	0.8325	0.7519
Average		0.4652	0.5062	0.8634	0.7341

Table 5: Obtained values for the evaluation metrics for the MTMC method for the two sequences used to test the model: 01 and 03.

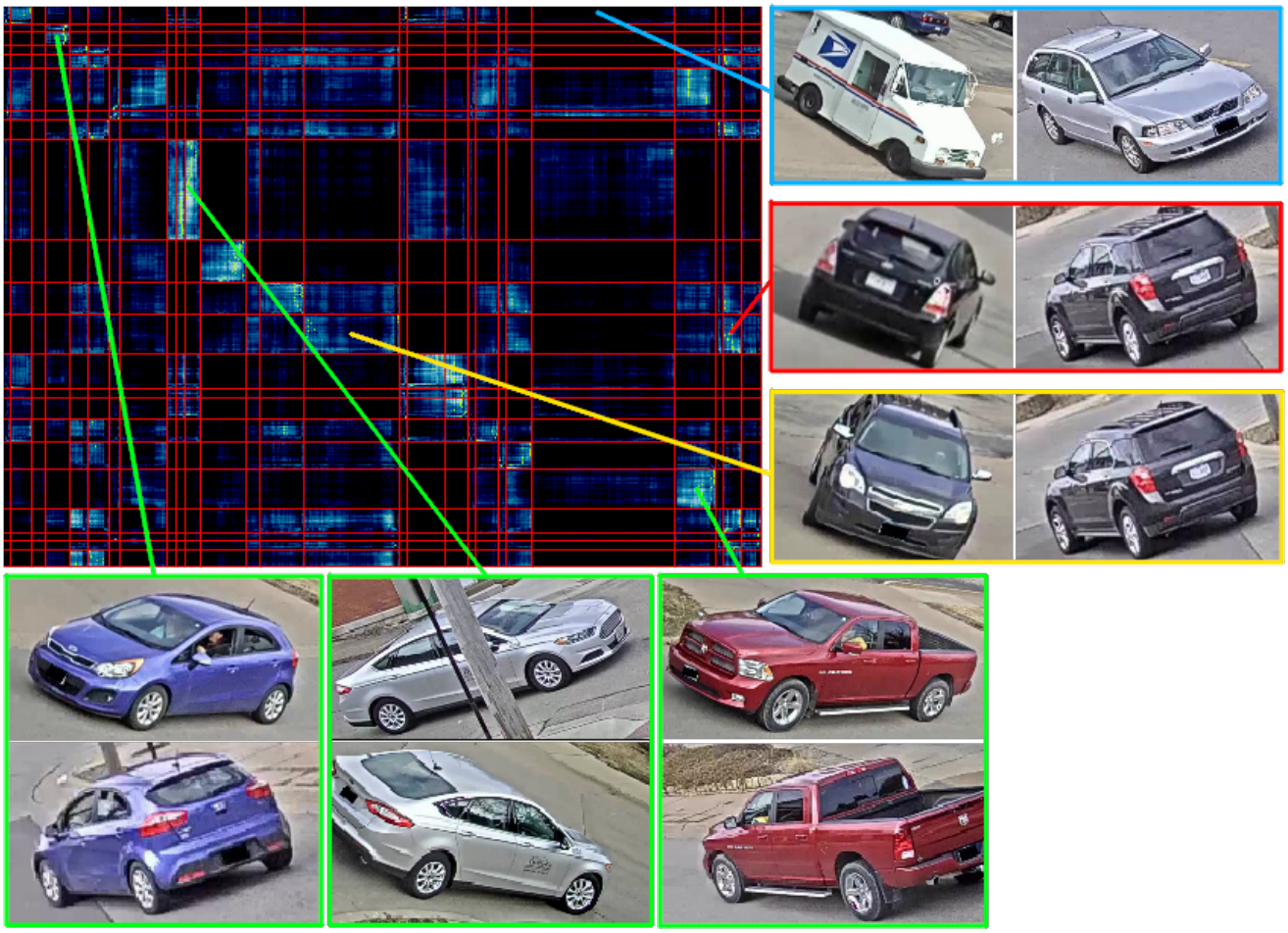

Figure 1: Distance Matrix between cars embeddings visualization. Different pairs are shown. Green: True Positive. Blue: True Negative. Red: False Positive. Yellow: False Negative.

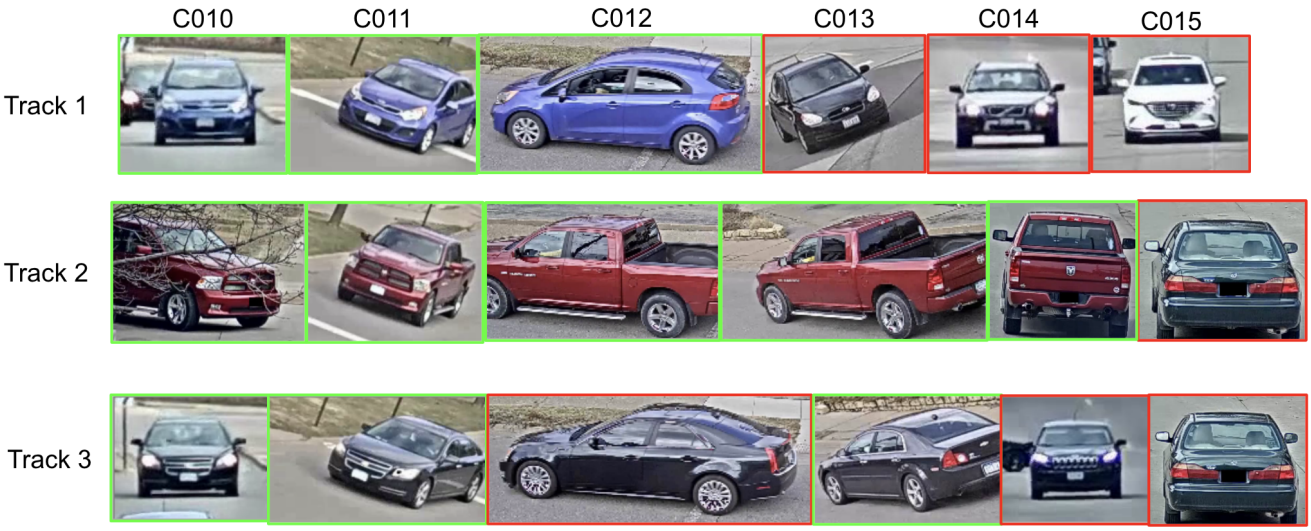

Figure 2: Example of the obtained cars classified for the MTMC model to be the same car for the first three tracks.