Video Surveillance for Road Traffic Monitoring

Marcelo Sánchez[1], Sergi Vidal[2],Eudald Ballescà[3], Kevin Martín[4]
Universitat Autònoma de Barcelona
Plaça Cívica, 08193 Bellaterra, Barcelona
marcelosanchezortega@gmail.com[1], servidal95@gmail.com[2]
eudald.ballesca@gmail.com[3], kevin.martin@gmail.com

Abstract

This paper analyses and sumarized the knowledge learned during the M6 module of Video Analysis in the Master in Computer Vision aiming to solve the third track of the AI-City Challenge [1]. This competition focuses in the problem of detection and tracking multiple vehicles across multiple cameras in an urban enviroment.We explain the process of solving multi-tracking in single camara to extending it to multi-tracking in multiples cameras. This paper discuss and analyses our proposed solution and the performance on the test set. Code is publicly available at: https://github.com/mcv-m6-video/mcv-m6-2022-team6

1. Introduction

This project was presented to us in the first lecture of the module with the idea of participating and, hopefully, solving the CVPR 2022 AI City Challenge [1]. This challenge is organized by the CVPR conference, one of the best conferences in the field of Computer Vision. This challenge has an educational background and aims to push the research and knowledge in Multi-Camera vehicle tracking. This challenge provides a dataset of more than 3 hours of video from 40 fixed cameras spawned across 10 intersections. The dataset is splitted in 5 scenarios: 3 for training, 2 for validation and 1 for test. In all this sequences we can find more than 220K bounding boxes from 666 annotated vehicles.

2. Related work

In this section we will give a brief overview of existing methods in the field of Multi-Target Single-Camera (MTSC) and Multi-target Multi-Camera (MTMC) scheme. Even though at first glance these two methods might look different, both of them share many similarities as they must detect object in each individual frame and provide a consistent tracking across multiple frames.

Both problems can be splitted in a first step of detection and a latter of tracking. For the detection the challenge organizers already give us the detection from several SoTA (state-of-the-art) methods for object detection such as SSD512 [2] YOLOv3 [3] and MASK R-CNN [4] detector. All of these predictions have been used as a base for the tracking algorithm.

In terms of MTSC, we can find multiple state-of-the-art methods such as:

- DeepSORT [6], an online method that combines deep learning features with Kalman-filter based tracking and the Hungarian algorithm.
- MOANA [7], which employs spatio-temporal data association using an adaptive appearance model to deal with identity switch caused by occlusion and similar appearance among nearby targets.
- TC [8], which uses a fusion of visual and semantic features to cluster and associate data in each single camera view. Additionally, a histogram-based adaptive appearance model is introduced to learn long-term history of visual features for each vehicle target.

Finally, re-ID matches the objects across cameras on MTMC. Originally, most of the works focused on person re-ID, such as Hermans et al. [9] with an implementation that used triplet loss with hard mining. In recent years, with an increasing interest on intelligent traffic management, vehicle re-ID has gained more attention and several proposals appeared. PROgressive Vehicle re-Identification (PROVID) [10] presents a framework based on deep neural networks. It considers vehicle re-ID in two progressive procedures: coarse-to-fine search in the feature domain, and near-to-distant search in the physical space. Hongye Liu et al. [11] propose a Deep Relative Distance Learning (DRDL) method which exploits a two-branch deep convolutional network to project raw vehicle images into an Euclidean space where distance can be directly used to measure the similarity of two vehicles.

3. Methodology

This section is divided in two main parts: MTSC tracking (Sec. 3.1) and MTMC tracking (Sec. 3.2).

3.1. Multi-target single camera tracking

The main steps we carry out in order to implement MTSC tracking are:

- Compute the detections of the cars for each frame separately.
- Use a tracking method to assign the same IDs to the same objects through the video.
- Apply a post-processing technique to remove small detections and parked cars to match the ground truth annotations.

3.1.1 Car detections

State of the art in this matter has already solve this task, detect cars can be done with several detectors, we will try it with some that are already pre-trained in Detectron2 [20] like RetinaNet [18] and Faster R-CNN [19] but we want to fine-tuned to optimize the results. Models will be tested using 75% as training set and 25% as test set.

3.1.2 Tracking method

Traditional methods of labeling follow coordinates rules but we want keep the same label independently of position we will test these three methods: maximum overlapping , maximum overlapping with direction and Kalman filter [6].

The maximum overlapping tracking method is based on intersection over union (IoU), where an auxiliary ID is assigned to every bounding box of the frame under evaluation and then assigned with the corresponding bounding box of the previous frame with highest overlapping. In other words, once bounding boxes are stored from current and previous frame the IoU metric is applied and IDs are matched and reassigned to highest scores. A threshold prevent mismatching.

The main problem of this method is, as is checking with the previous frame, if for one frame car is hidden or not detected, then the tracking is completely lose. To give extra information and solve this problem **maximum overlapping with direction** was implemented. This method give trajectory to each bounding box, using the previous 2 frames, so if 2 cars pass each other, the ID wont swap.

Kalman filter, the third method solve the same problem, this method use previous frames to predict where is going to be mixing the prediction with confidence obtained in previous frames.

3.1.3 Post-Processing

As challenge was focused on detecting cars that pass from one camera to another, that means we must have two consideration:

- All that cars that are parked are not tracked in the ground truth.
- Cars in far away positions, detector has problem to extract features because are tiny and most of the features can't be appreciate it, are also discarted.

For this reason, in order to have a properly evaluation, we must do the same.

- A threshold that checks if centroid of the bounding box move enough in 5 frames.
- A threshold that discard bounding box according to heigth and width.

3.2. Multi-target multi camera tracking

In order to solve the problem of keeping the same ID across all cameras, we use Pytorch Metric Learning, an open library [5] that allows us to include metric learning in our pipeline with the following steps:

- Create a dataset containing patches of all cars from different cameras with their label ID.
- Create an embedding space to cluster the instances using a siamese network.
- Create a re-identification algorithm using the trained network to find matches of a given car instance across different cameras.

3.2.1 Embedding Space using a Siamese Network

Once we have already identified the multiple cars in each of the single cameras, we face the problem of how to properly match the same car that passes through multiple cams. A naive approach we first followed was histogram 3D technique inspired by [12]. However, this approach really fails in this problem, as different cameras observe the cars from different directions and angles. 3D Histograms are not robust enough to encode all the information. To solve this problem we followed the approach of [13]. What we did was to create a representation space to later on properly match the same car from different views.This space is learning through a metric learning methodology using a Triplet Loss [14]. In our approach we first extracted features from a Resnet50 [15] network and then forward this feature trough a fully Conected network that creates a 512 size embedding space.

A clear representation of this space using U-MAP algorithm [16] can be seen in Fig. 1.

This space is latter on used in inference to match the same car from different views. We use k-means clustering in the embedding space to find most similar car.

One aspect that we found that really helps to better learn a more robuts embedding space is the use of transformations. The improvements can be see in 3.

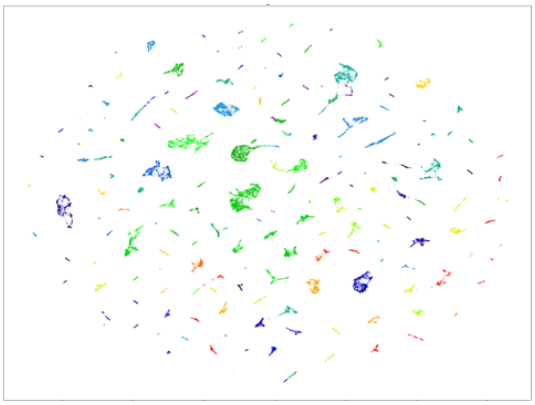

Figure 1. Resulting embedding space using U-MAP.

3.2.2 Re-ID method

To explain our algorithm we use two cameras: cam1 and cam2. For each camera, we have tracked each car separately, and we crop them to obtain car patches with their corresponding IDs. One camera is defined as the reference camera, therefore the IDs of its cars will be maintained: in this case cam1. Then, we take P patches from a car of cam2, and compare them (independently) with N patches of each car of cam1, in an all vs. all approach. In order to compare patches we use the trained Siamese Network, which provides a metric distance between patches. And if the distance between patches is below a threshold, they are considered as a match. To decide which car ID to assign to the patch of cam2 we count the number of matches that each car of cam1 has, and the ID of the car with more matches is assigned to the car of cam2. To ensure a more robust comparison we have used more than one patch for each car, as it might be that some patches of a car are not representative enough (e.g. due to occlusions).

To extrapolate this system to the six cameras of sequence 3, we propose a cascade scheme. First, two cameras are compared taking one as a reference. Once the re-id is done, these two cameras are combined and taken as a reference, to be compared with another one. This is done until all cameras have been re-identified. This scheme has been implemented in a random order of cameras, but we could have performed a spatio-temporal analysis to know which cameras to look for at each moment. However, the cameras are not well synchronized, as some frames are lost in the middle of the sequence. Moreover, by doing this analysis we would have over fitted the system to this specific sequence, as this would need a different configuration if the cameras were different.

4. Evaluation

4.1. MTSC tracking

To evaluate the MTSC methods we compute the IDF1, as is defined in the challenge, for each camera of the sequence 3. The results are summarized in table 2. That table summarize results of a fine-tuned process, with different tracking methods and post-processing applied. Overall Faster RCNN overpass RetinaNet experiments but to decide which tracker chose the decision is not obviously, 50% of the sequences has better results with Kalman and the other 50% is outperformed with IoU.

Camera C015 has a really bad results, the main problem in this sequence is some cars that appears here are not included in the ground truth.

4.2. MTMC Tracking

The siamese network is able to cluster in a 512-d embedding space the car instances of the dataset created with sequences 1 and 4, as observed in Fig. 1. After fine-tuning the network, the highest test accuracy obtained when matching car instances in sequence 3 is 0.35.

In Table 1 we present the MTMC results in sequence 3, using our final configuration: threshold to consider a match of 0.5, N = 4 and P = 6. As expected, MTMC performs worse in terms of IDF1 score than MTSC. This makes sense if we consider the augmented difficulty of the task, and the low accuracy obtained with the siamese network. Nonetheless, in the qualitative results presented in Fig. 2, we are able to track correctly a vehicle with the same id across 4 out of 5 cameras.

Sequence	**Metric**				
	Precis.	**Recall**	**IDR**	**IDP**	**IDF1**
3	74.83	86.55	46.01	42.72	43.69

Table 1. Results obtained with MTMC tracking and object detection on sequence 3.

5. Conclusions

Through this Module we have implemented from scratch a pipeline to solve different task, starting from single object single camera tracking to multi object multi camera tracking. This problem is based heavily on detections, a bad detection significantly impairs the performance of tracking methods. The best results were obtained with Faster R-CNN [19] in

Figure 2. Vehicle correctly tracked across four cameras (red) and erroneously in one (blue).

the detection side. In the tracking part of the problem, we saw that Kalman outperforms other methods.

In the problem of MTMC we found trough experimenting that siameses network outperform any other method and image augmentation also squeezes a little bit of improvement compared to no-augmentation pipeline.

In the problem of re-identify vehicles and track them across multiple cameras, we have obtained worse results than in the single camera case. In the MTMC scenario, we rely on the performance of the siamese network, and its low accuracy impairs the overall re-ID system.

There is a lot of improvement in this work, we think that a end-to-end system to properly track and Re-Id method such as [17] could transform the problem into fully differentiable and properly learn more insights about the data.

References

[1] Naphade, Milind, et al. "The 6th ai city challenge." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022. 1

[2] Liu, Wei, et al. "Ssd: Single shot multibox detector." European conference on computer vision. Springer, Cham, 2016. 1

[3] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. 1

[4] He, Kaiming, et al. "Mask r-cnn." Proceedings of the IEEE international conference on computer vision. 2017. 1

[5] K. Musgrave, S. Belongie, and S.-N. Lim, "Pytorch metric learning," 2020. 2

[6] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," 2017. 1, 2

[7] Z. Tang and J.-N. Hwang, "Moana: An online learned adaptive appearance model for robust multiple object tracking in 3d," IEEE Access, vol. 7, p. 31934–31945, 2019. 1

[8] Z. Tang, G. Wang, H. Xiao, A. Zheng, and J.-N. Hwang, "Single-camera and inter-camera vehicle tracking and 3d speed estimation based on fusion of visual and semantic features," in 2018 IEEE/CVF Conference on Computer Vision and Pattern RecognitionWorkshops (CVPRW), pp. 108– 1087, 2018. 1

[9] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," 2017. 1

[10] X. Liu, W. Liu, T. Mei, and H. Ma, "Provid: Progressive and multimodal vehicle reidentification for large-scale urban surveillance," IEEE Transactions on Multimedia, vol. 20, no. 3, pp. 645–658, 2018. 1

[11] H. Liu, Y. Tian, Y. Wang, L. Pang, and T. Huang, "Deep relative distance learning: Tell the difference between similar vehicles," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2167–2175, 2016. 1

[12] Kehl, Wadim, et al. "Real-time 3D model tracking in color and depth on a single CPU core." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017. 2

[13] Shuai, Bing, et al. "Siammot: Siamese multi-object tracking." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021. 2

[14] Dong, Xingping, and Jianbing Shen. "Triplet loss in siamese network for object tracking." Proceedings of the European conference on computer vision (ECCV). 2018. 2

[15] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. 2

[16] Becht, Etienne, et al. "Dimensionality reduction for visualizing single-cell data using UMAP." Nature biotechnology 37.1 (2019): 38-44. 3

[17] Meinhardt, Tim, et al. "Trackformer: Multi-object tracking with transformers." arXiv preprint arXiv:2101.02702 (2021). 4

[18] T. Lin, P. Goyal, R. Girshick, K. He, P. Dollár. "Focal Loss for Dense Object Detection" arXiv 1708.02002 (2018) 2

Detector IDF1	c010	c011	c012	c013	c014	c015	AVG
Retinanet Overlap	0.79	0.73	0.93	0.8	0.81	0.05	0.67
RetinaNet Kalman	0.8	0.63	0.86	0.69	0.82	0.05	0.64
Faster RCNN Overlap	0.88	0.73	0.91	0.84	0.87	0.06	0.72
Faster RCNN Kalman	0.89	0.69	0.93	0.75	0.85	0.05	0.69

Table 2. Post-processing MTSC results

Experiment	Metric				
	Precis.	**Recall**	**IDR**	**IDP**	**IDF1**
Resnet18	72.56	84.25	46.01	40.22	39.98
Resnet18 with transf.	72.94	84.45	46.11	41.34	40.75
Resnet50	72.71	84.37	45.10	40.52	41.85
Resnet50 with transf.	72.79	84.44	45.21	41.12	41.92
Resnext	47.7	86.25	46.89	41.51	53.44
Resnext with transf.	74.83	86.55	47.11	42.72	43.69
MobileNetV3	69.21	80.05	42.12	38.75	39.69
MobileNetV3 with transf.	70.01	80.15	42.27	41.52	39.81

Table 3. Embedding Result

[19] S. Ren, K. He, R. Girshick, and J. Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks". Proceeding from IEEE Transactions on Patter Analysis and Machine Intelligence 2, 3

[20] R. Stojnic, R. Taylor, M. Kardas et al. Detectron2 [Software] Available from https://github.com/facebookresearch/detectron2 2