Module 6. Video Analysis

Maria Vila Abad* Sara Lumbreras Navarro† Yael Tudela Barroso‡ Diego Velázquez§

Abstract

This report summarizes the work done in the final week of the Video Analysis module from Computer Vision Master. The main goal of the module is to perform detection and tracking of moving objects in video sequences. This week pretends to summarize all the learned techniques across the module to solve the third track of the AI-City Challenge. Concretely, in this report, we present the developed methods for multi-target single camera and multi-target multi-camera tracking.

Our source code is publicly available at: https://github.com/mcv-m6-video/mcv-m6-2020-team4.

1. Motivation

Object tracking consists of the identification of different objects assigning each object a unique ID and track them across frames in a video maintaining their IDs. In this paper, we explore two different types of tracking:

- Multi-target single camera tracking (MTSC): consists on tracking objects withing a single camera
- Multi-target multi camera tracking (MTMC): consists on tracking objects across several cameras

The different algorithms have been tested on video sequences from traffic video surveillance cameras available for the 2020 AI CITY CHALLENGE [8]. This challenge arises from the opportunity to make transportation systems smarter. Concretely, it consists on four different tracks from which this paper focuses on solving the third track: City-Scale Multi-Camera Vehicle Tracking. The dataset provided for this track contains 215.03 minutes of videos collected from 46 cameras spanning 16 intersections in a mid-sized U.S. city. This dataset is divided into 6 scenarios, from which three are used for training, two for validation, and one for testing.

*mariava.1213@gmail.com
†[2]Email: jfslumbreras@gmail.com
‡[3]Email: yael.tudela@cvc.uab.es
§[4]Email: diegovd0296@gmail.com

2. Related work

As the name suggests, multiple object tracking consists on keeping track of objects in a video as they move around. In other words, a good model has to accurately detect objects in each frame, and provide a consistent labelling of them. Challenges arise when objects are partially or completely occluded, or temporarily leave the field of view, since ideally the objects should keep their former IDs when reappearing. Furthermore, objects whose paths intersect might confuse the model and cause it to erroneously switch their IDs. Given how powerful object detectors have become, a very popular tracking method is based on object detection, namely *Tracking by Detection*. Some of the most prominent work along this branch is found in [7]. They use an algorithm based on Quad-CNN, which learns to associate object detections between frames using quadruplet loss. Recurrent Neural Networks (RNNs) have also been used for tracking. In [5] they use them to exploit the temporal axis that comes intrinsic with these object detections and their relationships between frames.

3. Multi-Target Single Camera Tracking

In order to solve the MTSC tracking problem we have followed a pipeline that consists on three main steps.

The first step of the pipeline consists on obtaining car detections for every frame of the videos in the AI CITY CHALLENGE dataset. Concretely, we have used the off-the-shelf predictions obtained with SSD [4], YOLO [6] and Mask R-CNN [2].

The second step of the pipeline consists on the tracking algorithm itself. In this paper, we detail two different tracking algorithms that are simple, fast and obtain good results. The first on is the SORT [1] algorithm which uses Kalman filtering to track objects across frames and to also refine bounding box predictions. This method performs state predictions by propagating the state probability distribution function (PDF) and updates the prediction PDF with the measurements. Concretely, the statistical model used is a linear velocity model. The second tracking algorithm used is the Maximum Overlap. Our implementation can be observed in algorithm 1. Note that each frame is compared with the previous five frames and the first five frames of the video. This is done to avoid flickering on the tracking and

to better detect parked cars as we assume that cars remain parked across the video.

Algorithm 1 Tracking in a video using maximum overlap

For each frame remove overlapping detections.
1st frame: assign a new ID for each detection
for frame F from second frame **to** last frame of the video **do**
  if overlap between detection in F and detections in (F - min(6, F), F-1) $>$ 0.4 **then**
    Assign the same ID value
  else

    if overlap between detection in F and detections in the first five frames of the video $>$ 0.4 **then**
      Assign the same ID
    else
      Assign new ID
    end if
  end if
end for

Finally, the third step consists of post processing the obtained tracking. This step is required as the ground truth does not have parked cars and our detections do. Therefore, this post processing step consists on removing the parked cars. In order to do so, for each object tracked along the video sequence, we compute its centroid. If the standard deviation of the centroid is smaller than a threshold we mark the car as parked and thus remove the object track.

3.1. Evaluation

To set which is the best threshold to remove parked cars, a range of different values is tested on videos from sequence S01 and S04. It can be observed on figure 2 the achieved IDF1 scores for the two mentioned methods and different threshold values to remove the parked cars. Between Maximum overlap and Kalman filter with constant velocity there is not a great difference but Kalman results are slightly better in most cases except when using Mask R-CNN detections. When the remove parked cars threshold is set too low we can observe an improvement comparing with the results with no post processing (although we might not be removing all the parked cars). For high values, we observe a drop on the IDF1 score as relevant detections are discarded.

A threshold of 25 is the one used as it yields the best IDF1 score. In table 1, we compare the different methods for sequence S03. See that in average the best performance is achieved with SSD detector, Kalman filter and post processing. Also notice that each individual camera has a different winning pair of detector and tracking method but they all work better with post processing. It is also interesting to highlight that some cameras perform worse than other be-

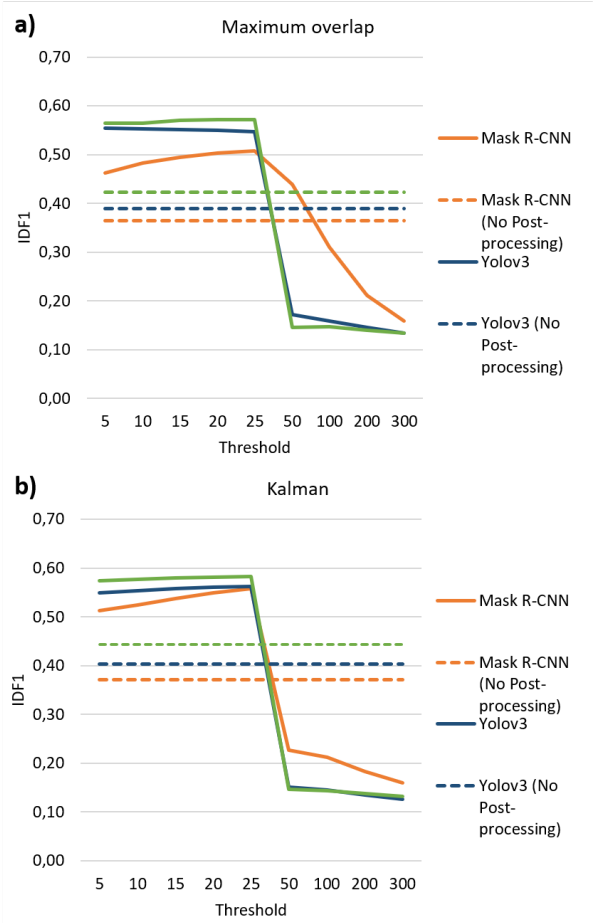

Figure 1. Average of IDF1 score for all videos in S01 and S02 for different thresholds to remove parked cars. a) Shows results when using the maximum overlap algorithm and b) Kalman filter with constant velocity.

cause there are few detections in the GT and a small mistake drops the score reached. This is the case, for instance, of camera c015.

Using SSD detections and Kalman with constant velocity for tracking we obtained the results for all the sequences. See the results of IDF1 score, precision and recall for the detections in table 2. We think that, regarding the simplicity of the algorithms, we have obtained good results.

A qualitative analysis of the results helped to detect the sources of some problems. For instance, on figure **??**, we can observe that the main issue of Kalman filter is that when there is a missed detection of an object at some frame in the video it is relabeled with a different ID afterwards. This makes tracking labels not consistent for the same object. Maximum overlap is able to deal with this issue. However, in Maximum Overlap, when an object occludes another one, it is labeled with the same ID. These kind of errors will later introduce some inconsistencies to our multi-target multi camera tracking algorithm.

Detector	Tracking	Post processing	c010	c011	c012	c013	c014	c015	Average
Mask R-CNN	Kalman cte vel.	No	0.2505	0.0408	0.0254	0.3099	0.5880	0.0028	0.2029
		Yes	0.7666	0.5241	**0.6263**	0.8184	**0.7467**	0.0699	0.5920
SSD	Kalman cte vel.	No	0.4361	0.2418	0.0394	0.6247	0.5829	0.0152	0.3234
		Yes	0.8353	0.6601	0.5156	0.6867	0.7361	**0.1611**	**0.5992**
YOLOv3	Kalman cte vel.	No	0.3580	0.0911	0.0286	0.6600	0.5481	0.0050	0.2818
		Yes	0.8415	**0.6839**	0.2291	0.7349	0.6908	0.0879	0.5446
Mask R-CNN	Maximum Overlap	No	0.2500	0.0419	0.0254	0.2964	0.4960	0.0029	0.1854
		Yes	0.7972	0.4481	0.5689	0.3130	0.5734	0.0515	0.4587
SDD	Maximum Overlap	No	0.4286	0.2125	0.0423	0.7775	0.5491	0.0143	0.3374
		Yes	**0.8756**	0.6082	0.4988	0.8161	0.6375	0.0982	0.5890
YOLOv3	Maximum Overlap	No	0.3358	0.0873	0.0243	0.7266	0.4387	0.0049	0.2696
		Yes	0.8431	0.6737	0.4901	**0.8274**	0.4652	0.1014	0.5668

Table 1. IDF1 score for each video in S03 for the different used methods.

Metric	IDF1	Precision (Detection)	Recall (Detection)
SEQ 1	59.1%	94.5%	51.7%
SEQ 3	59.9%	72.3%	70.4%
SEQ 4	62.9%	88.9%	59.1%
AVERAGE	60.6%	85.2%	60.4%

Table 2. Average for IDF1, precision and recall for detections for all the sequences in the test dataset.

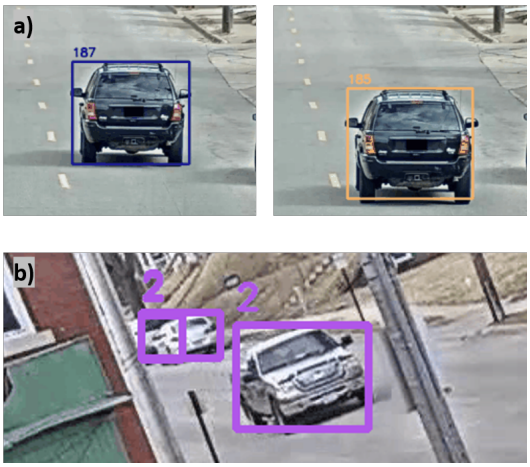

Figure 2. Qualitative examples of some errors for the different tracking algorithms a) Kalman filter and b) Maximum Overlap.

4. Multi-Target Multi Camera Tracking

In this section, we detail the method used to solve the MTMC tracking problem following the pipeline shown in figure 3. To solve the multi-camera problem we designed a method capable of relating same instance detections, regardless of the camera. A Siamese network is implemented to learn car representations in a concrete feature space. Resnet-50 is used as a feature extractor as it offers a good trade-off between quality of the embedding and processing time. A 256 feature space is used as last layer to project the learned features to a fixed size space. It is trained using fixed size cars crops from sequences 01 and 04 from the dataset. This set up works with two different learning rates; a reduced one for the backbone (Resnet-50 layers) and a higher one for the projection layer. The motivation behind this is to adapt the learned features slowly while we learn the re-projection to the 256 space. The network is optimized using SGD with momentum using Triplet-loss [3] as our cost function. We also perform online batch negative sampling in order to speed up the learning process. Once we have a system that is capable of relating two images, we designed a system capable of synchronizing cameras and match those embeddings. The synchronization is done by taking a reference camera, and fast-forwarding the rest of the cameras given their offset and their fps with respect to the reference camera. This is done in order to greatly reduce the number of embeddings needed to analyze, otherwise we would have to analyze the embeddings in every frame looking for matches in order to perform the re-identification. The embedding matching is done using euclidean distance and a grid-searched threshold.

4.1. Evaluation

We can observe in table 4 that reducing the embedding space reduces a lot the NMI. That translates to obtaining more sparse clusters, making it difficult to find correlations between multiple samples. We also observed a severe drop on both accuracy and NMI changing the backbone to a Resnet-18.

The results using this method were sub-par as we can see on table 4.1. We suspect that this is due to the tracking algorithm, since we can observe in the qualitative results a lot of flickering in the ids and constant changes in the id for the same car.

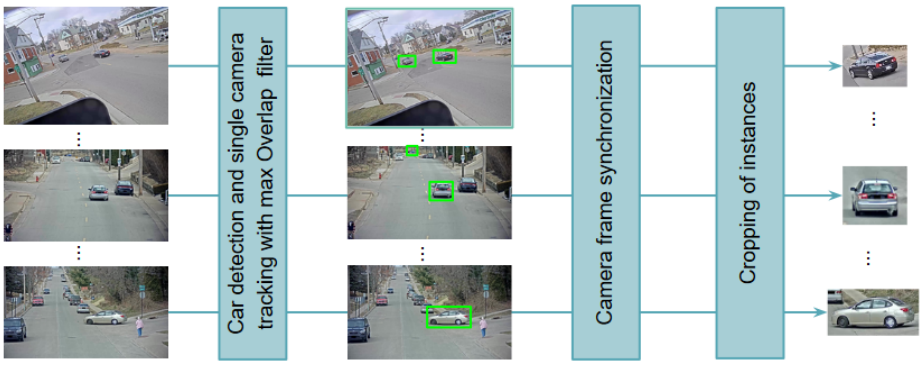

Figure 3. Pipeline of our method in MTMC.

			Camera						
Detector	Tracking	Metric	C010	C011	C012	C013	C014	C015	Average
Mask R-CNN	Max Overlap	IDF1	0.77	0.28	0.06	0.31	0.54	0.01	0.32
		Precision	0.63	0.17	0.03	0.23	0.58	0.01	0.27
		Recall	0.99	1	1	0.99	0.98	1	0.99
		IDP	0.63	0.16	0.03	0.19	0.43	0.01	0.24
		MOTA	0.42	-3.83	-28.24	-2.31	0.16	-158.5	-32.05
		MOTP	42.05	120.79	124.43	151.11	81.14	49.56	94.84

Table 3. Results obtained for our multi-camera method on secuence 3.

Backbone	Embedding size	Accuracy	NMI
Resnet-50	256	60.4	77.3
Resnet-50	128	57.3	69.3
Resnet-18	256	54.7	67.3

Table 4. Results obtained from the different backbones and embedding size

5. Conclusions

The proposed tracking methods for single and multi-camera are both valid although in the latter we have not obtained the results we were looking for. Single-camera tracking is a rather simple task which we have accomplished using rather simple algorithms. In this task, an IDF1 score of 60.6% is reached for an average of all the videos in the test set. Both methods used have some inherent problems which introduced misleading tracking IDs for the objects in the video. We think those errors might be messing with the embedding space studied for the multi-camera task of the challenge. The reached IDF1 score for this task is 32.0%. For multi-camera, the result is half as good as single-camera due to the wrong change of IDs. We believe the main sources of problems are the ones mentioned above for the tracking algorithms. These results could be improved with a more complex matching function or by improving the stability and consistency of the tracking method.

References

[1] Alex Bewley et al. Simple online and realtime tracking. In *IEEE International Conference on Image Processing (ICIP)*, 2016. 1

[2] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, Oct 2017. 1

[3] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In Defense of the Triplet Loss for Person Re-Identification. *arXiv*, Mar 2017. 3

[4] 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

[5] Anton Milan, S Hamid Rezatofighi, Anthony Dick, Ian Reid, and Konrad Schindler. Online multi-target tracking using recurrent neural networks. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017. 1

[6] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017. 1

[7] Jeany Son, Mooyeol Baek, Minsu Cho, and Bohyung Han. Multi-object tracking with quadruplet convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5620–5629, 2017. 1

[8] Ming-Yu Liu Xiaodong Yang Stan Birchfield Shuo Wang Ratnesh Kumar David Anastasiu Jenq-Neng Hwang Zheng Tang, Milind Naphade. Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification. 2019. 1