Video Surveillance for Road Traffic Monitoring

Sergio Montoya Adam Szummer Laia Albors Ibrar Malik
Universitat Autònoma de Barcelona, Spain
{ sergio.montoyadepaco, adam.szummer, laia.albors, ibrar.malik }@e-campus.uab.cat

Abstract

In recent years, Road Traffic Monitoring has been gaining more attention as it has become a very demanded video-surveillance task. In this report, we analyze classic and state-of-the-art techniques for multi-target single camera (MTSC) tracking and multi-target multi-camera (MTMC) tracking problems. Specifically, we focus on obtaining the best results in the AI City challenge 2022. We observe that by performing appropriate filtering of tracks and fine-tuning all the parameters in the tracking pipeline, significant improvements can be achieved over a basic tracker baseline. We analyze the performance of the proposed techniques using standard metrics for video analysis such as IDF1. The code for this project is made available in the following link: https://github.com/mcv-m6-video/mcv-m6-2022-team3.

1. Introduction

Road traffic monitoring and Advanced Driver Assistance Systems (ADAS) are aimed to improve safety, efficiency, and comfort at road transportation by means of information technologies. In order to assess the state-of-the-art of tracking techniques for Road Traffic Monitoring, the AI City challenge [1] provides single camera and multiple camera evaluation tracks. Both single-camera and multi-camera tasks are evaluated using the IDF1 score [2].

The goal of Multi-Target Single-Camera Tracking (**MTSC**) is assigning a unique ID to the same object along the whole video sequence. This task involves assigning detections and predictions from consecutive frames and possibly re-linking fragmented tracks by using identification templates.

The goal of Multi-Target Multi-Camera Tracking (**MTMC**) is to track vehicles that pass through the field of views of multiple sensors. This task is by nature more challenging that MTSC, it involves synchronizing camera views, re-identifying vehicle tracks across cameras appearing in multiple cameras at the same time or at different time instants.

2. Related work

2.1. Object detection

This is well defined and known challenge in Computer Vision industry that was tried to be addressed with classical computer vision, before even deep learning methods has come across. We can think of such approaches such HOG [3], SIFT [4], ORB or SURF combined with classifiers such as KNN, SVM [5] or family of RandomTrees algorithms [6]. Nowadays, we can see deep-learning approaches in architecture families such as: RCNNs [7, 8, 9], YOLOs [10] or different Transformers that provide state-of-the-art detection. However, not all can be used for the problem of multi-target camera tracking.

2.2. Object tracking

There are filters that can be applied to try to describe motion model such as Kalman filters [11]. Since those are often limited to certain assumptions of linearity other solutions can come into picture such as particle filters [12] or optical flow analysis to better track given objects. Finally deep learning for object tracking can be considered that unfortunately is computationally heavy, hence have to be carefully selected for the situation.

2.3. Object identification - instance detection

Finally, if we wish to track the same object in multiple cameras, frames, streams there is a need for object representation that can be re-identified. Easy solutions such as color histograms are easy to compute but do not prove to be sufficient for decent results. There could be considered mixes of solutions such as feature descriptors, SIFT, ORB mixed with histograms, gradients maybe even pyramids for detection of different sizes. However, again at this point deep-learning approaches such as embedding encoders proves to show best results such as Re-ID network architecture which uses ImageNet pretrained networks as a baseline. [13]

3. Methodology

3.1. Multi-Target Single-Camera Tracking (MTSC)

To approach MTSC, we perform a tracking by detection approach. The tracking by detection approach consists of first detecting cars at the current frame, updating those tracks that are still alive, and performing the association of detections at the current frame with the predictions of the trackers. In the following subsections, we will perform an ablation study of the importance of each component and how they can be correctly tuned to achieve the best results. We will use as reference for this discussion the video coming from the 10th camera of sequence 3 of the AI City challenge 2022.

3.1.1 Detection

Due to the large amount of work dedicated to object detection, it is easy to find well-performing object detection networks. One key factor that we have seen is that annotations in the AI City challenge 2022 are only provided for non-parked cars that are near to the camera. Using a pre-trained car detection network results in many false positives leading to bad IDF1 score for tracking, due to the large number of false tracks. If we finetune the car detection network to detect only those cars that are annotated, using the training sequence, we can remove a lot of false tracks. Specifically, we experiment with SSD, FasterRCNN, and RetinaNet. We obtain the best results with FasterRCNN, for which we obtain an IDF1 of 0.357 without finetuning and 0.664 with finetuning. When applying the finetuned network for tracking, we have to consider a threshold to decide which bounding boxes to keep.

3.1.2 Trackers

When considering different trackers, we have experimented with KCF, static tracker, and the kalman tracker. For the kalman tracker we have decided to use a constant velocity model and the state is composed by the velocity, the center, and area of the bounding box.

3.1.3 Association

The tracker and bounding boxes association is performed with the Hungarian algorithm [14]. For solving this association problem, we need to define a similarity between detections and predictions. For establishing such similarity, we use the intersection over union (IoU) and a similarity of identification templates. The final similarity is controlled with a parameter $beta$ that determines which term is given more weight. For computing identification templates for each car, we use a pretrained Re-Identification network [13]. To compare the identification template of a new detection to all the detections in the whole track, we decide to save a single template for each track. We arbitrarily decide to use a running average of the identification template to summarize the identity of the whole track. The running average is parameterized by a parameter $alpha$ that determines how quick the new templates are incorporated. Lastly, those detections that have no association are used to create a new track. While for those tracks that have no detection correspondence, they are kept alive for a maximum amount of frames $skip\_frames$. If after $skip\_frames$ frames, no new detection has been matched, the track is killed.

3.1.4 Post-processing

We have seen already the training, detection and tracker association that yields certain results given the data. When analyzing deeper the results of the initial solution pipeline, it has been observed that cars far from the camera and some of the parked cars are still being detected because of good performance of the fine-tuned network. However, the ground truth used for this challenge does not consider those detections and tracks as valid hence corrupting the final result. To better represent performance of our solution, we incorporated two separate solutions to deal with those detections.

3.1.5 Online vs offline post-processing

The camera feed of traffic monitoring is working in both online and offline mode. Which essentially mean that it allows to process and interpret the video frames as they progress in real-time as well as provide results of the saved feeds. Online removal of parked cars consist of analysis of track history of the alive tracks. We calculate the center of the tracker's bounding box and calculate standard deviation of the centers within data analysed and discard the trackers where standard deviation is below threshold. To improve the computational and time cost we have introduced the approach that analyses first 10 frames of the track and the last 10 frames of track from current processing frame. This allowed to remove cars and trackers that do no move within the limited history of the track. The quantitative results can be seen in Table 1, where we run both process on cameras from sequence 03 of the AI City Challenge 2022.

3.2. Multi-Target Multi-Camera Tracking (MTMC)

3.2.1 Online tracking

We have taken two approaches to solve the MTMC tracking problem, the first one being the *online* approach. In this setting, we take advantage of the timestamps of the multiple camera videos to read the multiple frames ordered in time, and perform both multiple-camera tracking using only past

camera	Offline idf1	Online idf1
c010	0.88	0.86
c011	0.67	0.54
c012	0.88	0.82
c013	0.76	0.76
c014	0.73	0.76
c015	0.42	0.68
avg.	0.78	0.74

Table 1: Result of cameras from sequence 3. Average excluding c015.

data. The pipeline we follow to solve this problem is simple: we perform tracking as in the single camera setting, and assign a global id for every new track. If this new track is similar to an already existing track in another camera video, we assign its global id.

How do we compute this similarity? We use the feature vectors from the ReID network [13] from the latest bounding box of every track, and compute the similarity matrix of all the existing tracks and the new tracks we want to re-identify. As this is a linear assignment problem, where we want assignments that maximize the total similarity, we can use the Hungarian algorithm [14]. The online approach has other difficulties such as the removal of the static/parked cars, which we do by computing a standard deviation of the center of the detections, and after k frames we remove if its smaller than some threshold. The issues with this approach is that we can only use past information to solve this problem, losing the context of the whole track and all its detections/features. On the experimental results we will see how the offline approach performs better, by being able to implement more complex re-identification algorithms due to having all the tracks and features of the whole video before performing the global assignments, and thus performing track-to-track comparisons instead of frame-to-frame.

3.2.2 Offline tracking

In the offline approach we already have all the cars tracked in each camera, so we use all their frames to decide which two cars are the same between the video frames from two different cameras. Therefore, the pipeline that we follow for the offline approach is: we take all tracked cars identified in each camera, pass their frames through the pre-trained Re-Identification network previously explained [13], which gives us one feature for each frame, so we then aggregate all the features from each tracked car computing the mean. Now, we want to find which of these features are more similar and can be considered the same car. We can think of this as a graph: each node is the tracking of a car from one camera and we have an edge between two nodes if the features of these two tracked cars are similar.

To compute this similarity, we just do the dot product between any pair of tracked cars between different cameras. Then, we use the similarity matrix obtained and the Hungarian algorithm [14] to find matches between tracked cars from different cameras. From the matches obtained, we only put an edge if the similarity value is above a given threshold (Similarity Threshold, an hyperparameter that we will have to tune). Once we have constructed this graph, we want to assign the same ID to all tracked cars that belong to the same car. That is, we want to find the **connected component** from this graph. This can be done applying the classic algorithm, stated in Algorithm 1.

The problem with this idea is that we can assign the same ID to tracked cars from the same camera (because they end up in the same connected component), which is not realistic assuming that the videos with which we are dealing have a short duration (less than a day), so the same car can not be seen twice with the same camera. To fix this, we have adapted this algorithm a bit so we never join two connected components if this implies that we are joining tracked cars from the same camera. The new version of the algorithm, that we named **connected-ish components**, can be found in Algorithm 2. In both cases, after finding the connected components from the graph, we just assign the same ID to all the tracked cars that belong to the same connected component.

4. Experimental results

4.1. MTSC

Once the trackers, association and post-processing approach has been defined and implemented we have performed extensive grid search of parameters for the whole pipeline that included. Best results can be seen in Table 2.

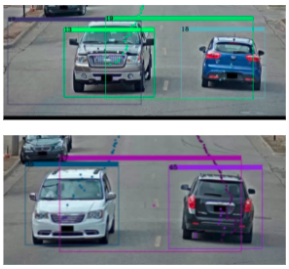
- alpha [0.2, 0.3, 0.4, 0.5, 0.6, 0.7] : the moving average parameters between consecutive feature vectors
- beta [0.2, 0.3, 0.4, 0.5, 0.6, 0.7] : describes the importance of feature vectors and IoU for association
- skip frames [5, 10, 15, 20] : maximum age of a tracker
- conf thresh [0.4, 0.5, 0.6, 0.7] : the detection threshold
- IoU thresh [0.2, 0.3, 0.4, 0.5]: the IoU for association
- tracker [IoU, kalman, kcf]: the different trackers used
- deep sort [True, False] : describes whether the feature vectors are used

When further analysing the results of grid search, we could see that for each sequence different set of best parameters have been chosen. That proves that it is indeed a challenge to find best possible setup for set of different cameras and locations. Qualitative analysis shown cases such as

Sequence	IDF1	alpha	beta	skip_frames	conf_threshold	Iou_thresh	Tracker	Deep_sort
S03	0.74	0.7	0.7	20	0.4	0.2	kalman	False
S04	0.60	0.5	0.0	15	0.6	0.4	kcf	False
S01	0.71	0.2	0.5	10	0.6	0.4	kalman	True

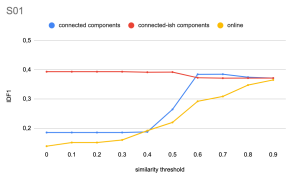
Table 2: Best result for each sequence where trained on other two. Grayed out numbers do not impact result.

(a) Car changing the track ID due to box size

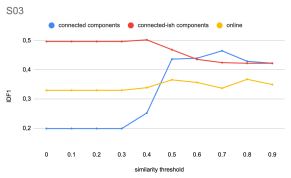


(b) Two cars being detected as one.

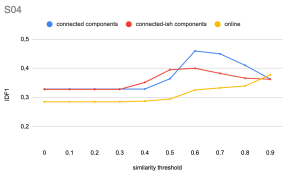
Figure 1: Qualitative results after grid search.



(a) Sequence 1 (S01)



(b) Sequence 3 (S03)



(c) Sequence 4 (S04)

Figure 2: Quantitative results regarding the offline approach in the MTMC tracking. Effect of the similarity threshold on the IDF1 in all three sequences.

car losing tracker because of quick change of size of bounding box due to location of the camera seen on Figure 1a or furthermore some of cars passing by being detected as one seen on Figure 1b.

4.2. MTMC

In the three methods explained, we only have one hyperparameter to tune (since we start from the the best tracking found in subsection 3.1), the similarity threshold.

In Figure 2 we can see that the offline algorithms always outperform the online one (as expected, since in the offline case we have more information when taking a decision). And between the two offline algorithms, except in sequence 4, Figure 2c, in general our modification of the connected components algorithm achieves better results than the classic one.

We can also see the effect of the Similarity Threshold. In the online case and the offline that uses the classic connected components algorithm, they get better results with higher thresholds. However, with the connected-ish components algorithm, we see that in general it performs better with lower thresholds, or at least is more stable and it does not depend as much as the other two on this hyperparameter.

Therefore, we can deduce that the best method to do MTMC tracking if the offline approach using the connected-ish components algorithm and a Similarity Threshold of 0.4. The quantitative results can be found in Table 3.

Metric	**IDF1**	**IDP**	**IDR**	**Precision**	**Recall**
S01	0.392	0.324	0.505	0.580	0.915
S03	0.502	0.441	0.590	0.703	0.948
S04	0.352	0.342	0.426	0.647	0.918
Average	0.415	0.369	0.507	0.643	0.927

Table 3: Final results for the MTMC tracking using the offline approach, the connected-ish components algorithm and a similarity threshold of 0.4.

5. Conclusions

In this work we have reviewed common methods to detect and track objects in a multi-camera setting. Building from single camera tracking, we have tested in practice different tracking methods such as KCF or Kalman. With all of these we see limitations when facing occlusions or track intersections for similar objects. Another important insight is on the importance of having good labelled data when evaluating. We must understand how our data is defined, such as knowing which objects are being tracked or which are not, e.g. parked cars. We must also know the limitations of the labelling: if small detections are not labelled, we must perform some post-processing to our predictions to correctly evaluate them.

Not only we depend a lot on the post-processing to get good evaluation results, we also need to perform hyperparameter tuning and fine-tuning of the learnt models to get good scores. This results on a tracking system that cannot generalize well to new scenes, and we must take this into account when comparing with other methods that might not need to tune any hyper-parameters. Finally, in our work we have compared online and offline algorithms to perform the tracking: the main difference being that we can effectively post-process the predictions on the offline methods, and aggregate features from tracks to have much more robust trackings. Depending on the requirements of our use case both offline and online can be good choices.

References

[1] Milind Naphade, Shuo Wang, David C. Anastasiu, Zheng Tang, Ming-Ching Chang, Yue Yao, Liang Zheng, Mohammed Shaiqur Rahman, Archana Venkatachalapathy, Anuj Sharma, Qi Feng, Vitaly Ablavsky, Stan Sclaroff, Pranamesh Chakraborty, Alice Li, Shangru Li, and Rama Chellappa. The 6th ai city challenge, 2022. 1

[2] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European conference on computer vision*, pages 17–35. Springer, 2016. 1

[3] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005. 1

[4] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 1

[5] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995. 1

[6] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. 1

[7] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 1

[8] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 1

[9] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015. 1

[10] 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

[11] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960. 1

[12] Jun S Liu and Rong Chen. Sequential monte carlo methods for dynamic systems. *Journal of the American statistical association*, 93(443):1032–1044, 1998. 1

[13] Zhedong Zheng, Minyue Jiang, Zhigang Wang, Jian Wang, Zechen Bai, Xuanmeng Zhang, Xin Yu, Xiao Tan, Yi Yang, Shilei Wen, et al. Going beyond real data: A robust visual representation for vehicle re-identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 598–599, 2020. 1, 2, 3

[14] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955. 2, 3

6. Appendix

6.1. Multi-Target Multi-Camera Tracking (MTMC)

Classic algorithm to find the connected components of a given graph, $G := \{V, E\}$, where V are the node and E the edges of the graph, G.

Algorithm 1 Connected Components

$G := \{V, E\}$
$CC := []$ ▷ Start with no connected components
for all edges e in E **do**
  $n_1 =$ First node from e
  $n_2 =$ Second node from e
  if n_1 and n_2 not in CC **then**
    $c \leftarrow \{n_1, n_2\}$
    Add c to CC
  else if Only n_1 in CC **then**
    $c_1 \leftarrow$ Connected component of n_1 in CC
    Update c_1 in CC adding n_2
  else if Only n_2 in CC **then**
    $c_2 \leftarrow$ Connected component of n_2 in CC
    Update c_2 in CC adding n_1
  else if Both n_1 and n_2 in CC **then**
    $c_1 \leftarrow$ Connected component of n_1 in CC
    $c_2 \leftarrow$ Connected component of n_2 in CC
    Remove c_1 and c_2 from CC
    $c \leftarrow$ Join $c1$ and $c2$
    Add c to CC
  end if
end for

Modification of the classic algorithm to find the connected components of a given graph, $G := \{V, E\}$, that avoids joining two connected components if it implies that they will be joining nodes with the same characteristic. In our case, this characteristic is the camera ID.

Algorithm 2 Connected-ish Components

$G := \{V, E\}$
$CC := []$ ▷ Start with no connected components
Sort E in descending order according to their similarity score
for all edges e in E **do**
  $n_1 =$ First node from e
  $n_2 =$ Second node from e
  if n_1 and n_2 not in CC **then**
    $c \leftarrow \{n_1, n_2\}$
    Add c to CC
  else if Only n_1 in CC **then**
    $c_1 \leftarrow$ Connected component of n_1 in CC
    if n_2 camera not in c_1 cameras **then**
      Update c_1 in CC adding n_2
    end if
  else if Only n_2 in CC **then**
    $c_2 \leftarrow$ Connected component of n_2 in CC
    if n_1 camera not in c_2 cameras **then**
      Update c_2 in CC adding n_1
    end if
  else if Both n_1 and n_2 in CC **then**
    $c_1 \leftarrow$ Connected component of n_1 in CC
    $c_2 \leftarrow$ Connected component of n_2 in CC
    if $n_{1,2}$ cameras not in $c_{1,2}$ cameras **then**
      Remove c_1 and c_2 from CC
      $c \leftarrow$ Join c_1 and c_2
      Add c to CC
    end if
  end if
end for