Video Surveillance for Road Traffic Monitoring

Ruben Bagan Joan Fontanals Vernon Stanley Pablo Domingo
Master in Computer Vision

Abstract

We present a comparison between different techniques to tackle the multi-object single and multi-camera tracking problem proposed by the AI City challenge 2021, which consists on tracking different vehicles across multiple cameras spread out in different areas of a city. The comparison is done by establishing a baseline in order to proper compare the proposed methods and then analyze the performance of the proposed techniques using standard metrics for video analysis such as mAP and IDF1. The code used for this analysis can be found at: `https://github.com/mcv-m6-video/mcv-m6-2021-team2`

1. Motivation

Nowadays, with the increasing number of self-driving cars, there's a need to make video surveillance systems smarter, to prevent accidents and improve self-driving cars. One project that focuses in solving this is the AI City [4] specifically the Multi-Object Single and Multi-Camera tracking challenges.The challenge consist on detecting and tracking vehicles captured across multi cameras in different areas and intersections of a city. This challenge provides different sequences where different cameras record some streets and intersections.

2. Related Work

Multi-Object Multi-Camera Object tracking has received a lot of attention in research [5]. Metric Learning based approaches have been widely used specially in the challenge of person re-identification [11]. This is useful because once the have the id for each instance identify in each camera there is a need to stablish the same id across the multiple cameras.This can be done by first assigning a principal camera and then (using pairs of cameras) re identifying the id instances. The mian problem is that, this is only applicable when the cameras are separated across time, meaning that one instance does not appear in the same time in another camera, but when the camera principal camera can't be assigned, then problems arises. In our approach we try to apply feature extraction combines with a graph-based approach that handles the re-identification by finding maximal cliques in a graph [2]. This helps us ignore the spatial-temporal, as the identifications are done separately not taking into account time or space, instead the graph decides the voting and therefore an instance can be identified in different cameras in different timestamps.

3. Methodology

The intention is to divide the work in two sections, one for each challenge. The first section devoted to explain the techniques used for single camera tracking approach and then scale the techniques for the multi camera approach.

3.1. Multi-Object Single Camera Tracking

AI City challenge apart from more than 3 hours of video recordings also provides the detections of cars of three detectors being Mask-RCNN, Yolo3 and SSD512. Apart from that it also provides computed tracking algorithms in order to compare with our methods, such as DeepSort [3], Moana [8], TC [7]. Another thing to notice is that the ground-truth provided only contains moving objects while the tracks provided also track parked cars. This would cause problems when computing the IDF1 [6] so in order to solve it we decided to filter the tracks using two techniques

- Impose a minimum number of tracks to determine that a track is valid
- Remove the trackings where it's detections centroids are not distance enough, which will mean that the objects are not moving.

The first point allows us to remove spurious tracks detected when bounding boxes are missing or when there are multiple detections on one object and multiple tracks are created to follow one unique object. We decided to use 5 detections at least for the track to be valid. Then to remove the parked cars we decided to take the centroids of the bounding boxes in a track and compute the distance between them. If the accumulated distance is bigger than a threshold is considered that the track is moving.
In order to find the best baseline to compare it against our

methods we decided to test different thresholds for the distance between centroids, for all three detectors and all three provided tracking techniques. The results can be seen on figures 6, 7 and 8. As the results show, we decided to use the algorithm TC as baseline with the detections from Yolo3 and the distance between centroids of 725 which had a result of 62.55% in IDF1. The IDF1 was computed using the motmetrics [9]. Once explained how we obtained our baseline to compare against our methods, now we will explain which methods we used for tracking in single camera scenario.

Maximum Overlap
The maximum overlap technique matches bounding boxes in consecutive frames according to the IoU. Each bounding boc on frame N+1 is exclusively assigned to the track of the highest IoU match of bounding boxes on frame N. When on frame N there is no bounding box that match the current tracks, that track is finished and with new detections new tracks are created.

Maximum Overlap has a two main problems, the first one is that spurious tracks can appear and the second that as it depends a lot of the bounding boxes detected, if a detections is missing, the track will finish creating two tracks for the same object. The first problem is solved using the filtering we have mentioned previously. The second problem can be solved thanks to Kalman Filtering.

Kalman Filter
Kalman filtering is a two-step procedure which allows to track objects across frames, first an estimation of the future position of the instance is done followed by an update after the new observation arrives.

Optical Flow
In order to improve the tracking of the instances by maximum overlap an option is to add the information obtained by the optcal flow, which indicates where the object is moving. So a enhancement of Maximum Overlap could be to diplace the detected bounding boxes form the previous frame according to the estimated motion from the optical flow

3.2. Multi-Object Multi-Camera Tracking

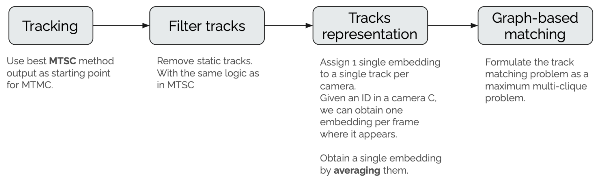
Multi-Object Multi-Camera tracking (MTMC) can be seen as the composition of two main problems that can be handled independently. The first one is to detect the objects independently in each camera. This problem is shared with Multi-Object Single Camera Tracking, and therefore the techniques already seen are used for MTMC. The second problem is to find correspondences between the tracks identified in each camera. This means, identifying which tracks of different cameras correspond to the same car.

In order to solve this challenge we developed a pipeline relying on different feature extraction techniques and the finding of maximal length cliques [2] in a graph structure to match cars in different cameras.

3.3. Pipeline

The pipeline designed to solve this problem is composed of four building blocks.

- Single camera tracking and filtering. Apply the same techniques as in the MTSC challenge, and remove static tracks.
- Feature extraction. Assign a feature vector to every car per camera.
- Graph-based matching. Find correspondences between different cars in different cameras, based on a graph built from neighborhood relations between cars in different cameras.


Tracking
Filter tracks
Tracks representation
Graph-based matching
Use best MTSC method output as starting point for MTMC.
Remove static tracks. With the same logic as in MTSC
Assign 1 single embedding to a single track per camera. Given an ID in a camera C, we can obtain one embedding per frame where it appears.
Obtain a single embedding by averaging them.
Formulate the track matching problem as a maximum multi-clique problem.
Figure 1. Pipeline used in our method

3.4. Feature Extraction

In order to find correspondences between objects in different cameras, we need some measure of similarity, and to get similarity measures we need to describe objects in a way so that we can compute distance metrics between them. We have used two different techniques to obtain these descriptions using tools to generate feature vectors.

- Histograms: Use histogram based vectors as feature representation.
- Metric-learning: Learn embeddings that separate instances in the latent embedding space

In order to validate the assumption that feature vectors play a vital role in our pipeline, we have compared our results with the usage of random embeddings.

3.4.1 Histogram

Histograms have been widely used as feature representation for images [12]. We try to use histograms as a source of our embeddings under the assumption that color distribution inside the bounding boxes of the car could uniquely identify them.

In order to use information coming from the 3 color channels, histograms obtained from different channels are concatenated and normalized to form a single vector descriptor. Two hyper-parameters have been tweaked while using histograms.

- Number of bins: The number of bins used in every channel's histogram.
- Distance metric: Different distances metrics have been considered in the experiments (euclidean, correlation and hellinger).

3.4.2 Metric Learning

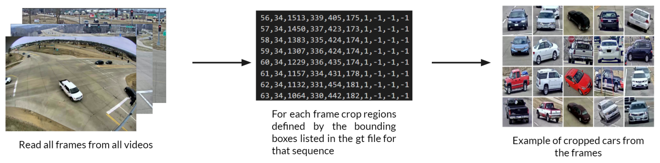
We used an existing implementation of Deep Metric Learning that can be found in the Torchvision repository[10]. By default the implementation provided uses Resnet50[1]. However, we decided to pick an small version of Resnet, the Resnet18[1]. The default implementation uses online triplet mining with "batch hard". This method creates triplets with the 'hardest' positive (farthest positive) and negative (closest negative). The loss metric used was:

$$L = max(d(a,p) - d(a,n) + margin, 0)$$

. Where the distance is computed using

$$||a-b||^2 = ||a||^2 - 2 < a, b > + ||b||^2$$

. The Figure 2 shows the process to create the database to feed the net.


Read all frames from all videos
For each frame crop regions defined by the bounding boxes listed in the gt file for that sequence
Example of cropped cars from the frames

```
56,34,1513,339,405,175,1,-1,-1,-1
57,34,1450,337,423,173,1,-1,-1,-1
58,34,1383,335,424,174,1,-1,-1,-1
59,34,1307,336,424,174,1,-1,-1,-1
60,34,1229,336,435,174,1,-1,-1,-1
61,34,1157,334,431,178,1,-1,-1,-1
62,34,1132,331,454,181,1,-1,-1,-1
63,34,1064,330,442,182,1,-1,-1,-1
```

Figure 2. The caption below each figure explains each of the steps to create the database used by the deep metric learning network.

Not all the detections have the same size and this is required by our net, therefore we have to apply a resize on all cropped cars. We decided apply a resize of 128 x 128. The Table 1 summarizes the database content divided by train an test.

	Train Seqs [1,4]	**Test Seq 3**
# Images	25290	4443
# ID	164	18

Table 1. Database statistics used to fed the deep metric learning model.

The Table 2 contains the hyperparameters used to train the deep metric learning network.

Optimizer	**Lr**	**Decay**	**P**
Adam	1e-3	0	[10,20,30]
K	**Margin**	**Epoch**	
[20,50,80]	[0.1,0.2,0.5,0.8,1.0]	10	

Table 2. Hyperparameters used to train the deep metric learning network.

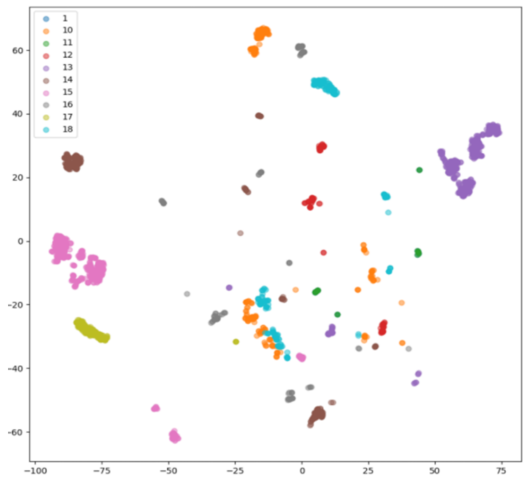


Figure 3. Train TNSE plot.

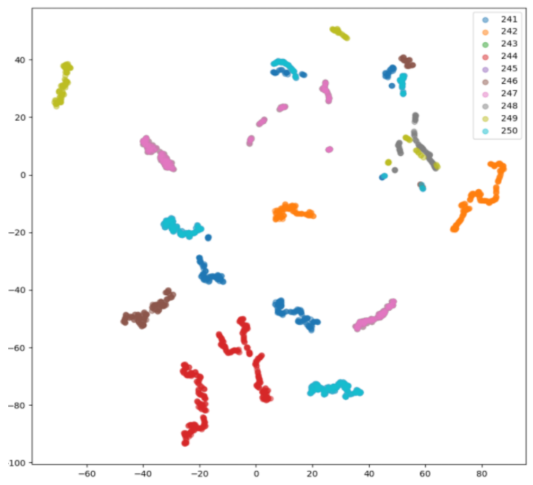


Figure 4. The test data is almost fully clustered, however the net was not able to cluster the cars with id 248, 250 and 249. One possible reason is that those cars appears in the same frame multiple times. Or the margin is too high and therefore those cars are set to another car id.

With the previous configuration we achieve a 93% accuracy.

3.5. Graph based identification

Once every object detection in every camera can be assigned a feature vector, the challenge is reduced to find correspondences and identify the same car in different cameras. This challenge gets harder with the presence with more cameras.

Inspired by [2], we propose a graph-based approach to tackle this problem. We create a graph where every node is an object ID for a camera. These nodes are connected to their nearest neighbors in every other camera and the edge is weighted by the inverse of the distance in the corresponding embedding space. Then, we find cliques in this graph, a clique in a graph is a subgraph inside a graph where every node is directly connected to every other node. This means, that if we find a clique we can be very certain that all the nodes correspond to the same car instance. All the nodes in the subgraph have each other as nearest neighbor in every other camera. The procedure is as follows:

- Track representation: For a given camera and car, assign a unique vector representation by averaging each feature vector detected from every frame. Obtain a set of nodes (cam, id) where cam corresponds to a camera id, and id corresponds to an object identified in that camera.

- Given each (cam, id), find the nearest neighbour id in every other camera. Build an edge connecting these two nodes and weighted by the inverse of the distance.

- Iteratively, find cliques of maximal length in this graph and remove them from the graph. Run until cliques of length one are only found. All the nodes in each of these cliques are considered to correspond to the same car observed in different cameras.

This framework offers a robust algorithm to solve any kind of MTMC challenge without the need of relying on specific heuristics that cannot be extrapolated to different scenarios.

4. Results

4.1. Results for MTSC

The results for MTSC can be found in the appendix. The table 4 summarizes the results obtained with sequence 1. The tables 5 and 6 are for sequences 3 and 4 respectively. The technique found in MTSC with highest IDF1 is use the baseline with TC and Yolo3. This detector will be used as a base for MTMC.

4.2. Results for MTMC

The **precision** for all techniques is: **81.06** and the **recall** is: **89.79**. All the techniques had the same precision and recall because we used the same detection file: **mtsc_tc_yolo3** based on the best detector found in MTSC.

Techniques	IDF1	IDR	IDP
Histograms, 16 bins, euclidean	46.80	44.75	49.04
Histograms, 32 bins, euclidean	51.21	51.94	50.49
Histograms, 16 bins, correlation	44.03	46.40	41.89
Histograms, 32 bins, correlation	43.43	45.77	41.32
Histograms, 32 bins, hellinger	49.44	52.10	47.03
Histograms, 64 bins, hellinger	**52.59**	**55.42**	**50.04**
Metric Learning, euclidean	51.66	54.43	49.14
Metric Learning, cosine distance	41.82	54.61	49.30
Baseline, Random Embeddings	35.90	33.67	38.38

Table 3. MTMC Results applying different techniques

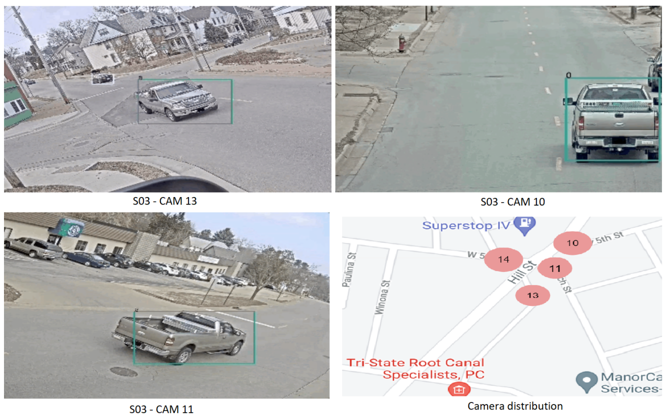


Figure 5. Qualitative results from MTMC. The car surrounded by a green bounding box corresponds to the car with ID 0. The net is capable to detect that car even if the cameras are not synchronized.

5. Conclusions

Metric learning techniques have proven to be ideal for identifying the same car in different scenarios and separating the different instances and works well with non-synchronized cameras. Surprisingly the metric learning has lower IDF1 compared to histogram. Whereas histogram feature representations also performed well but depend a lot of the parameters (number of bins) and the distance used. The hellinger distance proved to be a good distance when comparing histograms. Approaching the assignment problem as a graph-based problem has provided a robust approach that could potentially work in many scenarios.

MTSC and MTMC, despite its similar name, have very different challenges and are both complicated tasks. With MTSC we can use basic computer vision tools to get a high IDF1 whereas with the MTMC we're facing the problem to identify exactly the same car in different times and places.

References

[1] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[2] WenQian Liu, Octavia I. Camps, and Mario Sznaier. "Multi-camera Multi-Object Tracking". In: *CoRR* abs/1709.07065 (2017). arXiv: 1709.07065. URL: http://arxiv.org/abs/1709.07065.

[3] A. Bewley N. Wojke and D. Paulus. "Simple online and realtime tracking with a deep association metric". In: (Sept. 2017).

[4] Milind Naphade et al. "The 4th AI City Challenge". In: (June 2020).

[5] Ergys Ristani and Carlo Tomasi. "Features for Multi-Target Multi-Camera Tracking and Re-Identification". In: *CoRR* abs/1803.10859 (2018). arXiv: 1803.10859. URL: http://arxiv.org/abs/1803.10859.

[6] Ergys Ristani et al. "Performance Measures and a Data Set for Multi-Target, Multi-Camera Tracking". In: (Sept. 2016).

[7] "Single-camera and inter-camera vehicle tracking and 3d speed estimation based on fusion of visual and semantic features". In: (June 2020).

[8] Z. Tang and J. Hwang. "Moana: An online learned adaptive appearance model for robust multiple object tracking in 3d". In: (Mar. 2019).

[9] Jack Valmadre and Christoph Heindl. *py-motmetrics*. https://github.com/cheind/py-motmetrics. 2018.

[10] Vasilis Vryniotis and Ben Weinstein. *Similarity Learning Using Triplet Loss*. https://github.com/pytorch/vision/tree/master/references/similarity. 2018.

[11] Yandong Wen et al. "A Discriminative Feature Learning Approach for Deep Face Recognition". In: vol. 9911. Oct. 2016, pp. 499–515. ISBN: 978-3-319-46477-0. DOI: 10.1007/978-3-319-46478-7_31.

[12] Yue Zhang, Chuancai Liu, and Jian Zou. "Histogram-based embedding for learning on statistical manifolds". In: (May 2016).

A. Results for MTSC

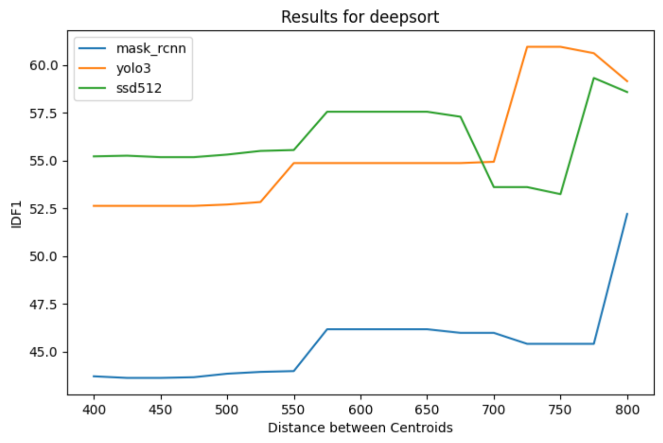


Figure 6. Results for deepsort

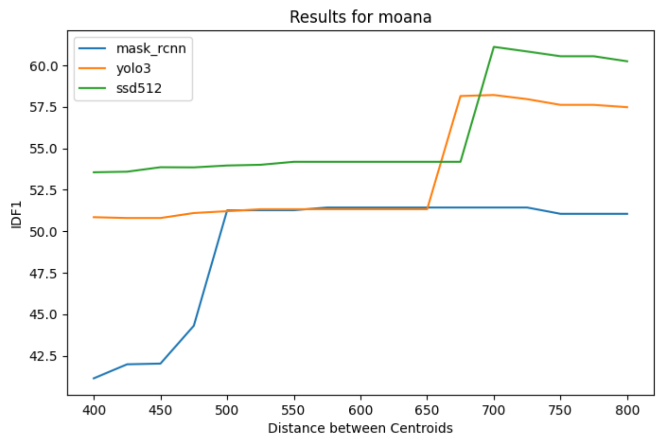


Figure 7. Results for moana

IDF1 (SEQ 1)						
Techniques	**C01**	**C02**	**C03**	**C04**	**C05**	**Average**
Max Overlap, Mask-RCNN	30.28	60.25	70.16	65.66	18.94	49.06
Max Overlap, Yolo3	23.84	49.29	71.29	60.89	19.85	45.03
Max Overlap, SSD512	20.41	50.71	67.53	50.14	15.55	40.86
Kalman Filter, Mask-RCNN	57.46	62.06	64.54	61.93	40.40	57.28
Kalman Filter, Yolo3	51.83	69.28	66.27	**69.21**	35.59	58.44
Kalman Filter, SSD 512	59.64	69.85	68.72	65.91	**49.18**	**62.66**
Max.Overlap+Opt.Flow Yolo3	23.91	46.25	**71.92**	60.98	19.66	44.55
Baseline TC, Yolo3	**70.37**	**75.03**	66.51	66.85	32.21	62.20

Table 4. MTSC Results using the Complete sequence 1. Kalman Filter using SSD512 detector is slightly better than the baseline TC with Yolo3 and outperforms the rest of out of the box detectors.

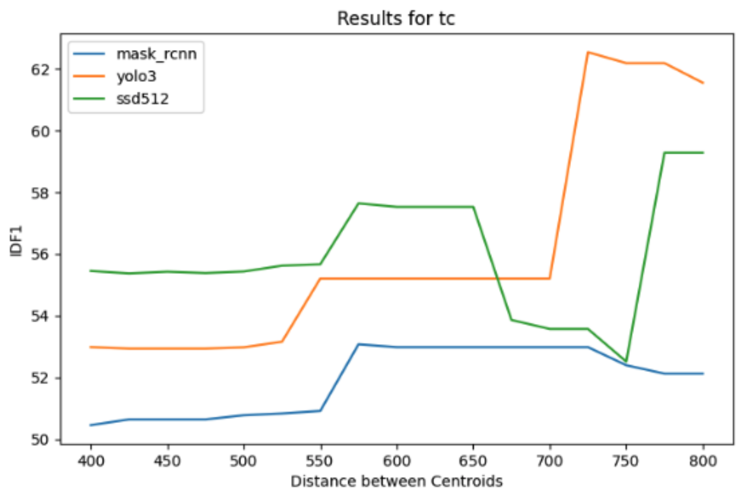


Figure 8. Results for tc

IDF1 (SEQ 3)							
Techniques	**C10**	**C11**	**C12**	**C13**	**C14**	**C15**	**Average**
Max Overlap, Mask-RCNN	22.83	11.42	7.75	9.8	20.11	20.00	15.32
Max Overlap, Yolo3	25.72	10.46	6.65	15.49	19.15	20.98	16.40
Max Overlap, SSD512	24.05	11.41	8.62	14.90	21.18	0.0	13.36
Kalman Filter, Mask-RCNN	59.67	43.50	56.32	50.35	61.02	21.25	48.69
Kalman Filter, Yolo3	74.07	27.54	56.46	72.71	73.93	**24.43**	54.86
Kalman Filter, SSD 512	72.14	24.28	**58.57**	65.69	55.98	**0.0**	46.11
Max.Overlap+Opt.Flow Yolo3	25.71	10.45	6.65	15.50	19.07	20.98	16.39
Baseline TC, Yolo3	**80.26**	**62.07**	48.72	**85.60**	**77.62**	20.99	**62.55**

Table 5. MTSC Results using the Complete sequence 3. The baseline TC with Yolo3 is the best by far. With sequence 1 the Kalman Filter using SSD512 was the best choice, however this technique is still a good option compared to the others.

IDF1 (SEQ 4)								
Techniques	**C16**	**C17**	**C18**	**C19**	**C20**	**C21**	C22	**C23**
Max Overlap, Mask-RCNN	95.57	61.53	24.90	97.42	29.45	61.21	69.66	15.83
Max Overlap, Yolo3	63.22	49.12	25.38	96.51	24.07	49.09	63.83	18.21
Max Overlap, SSD512	63.46	**52.47**	28.67	**99.11**	19.17	60.17	60.65	14.50
Kalman Filter, Mask-RCNN	73.43	46.45	42.80	8.12	30.64	71.89	**72.85**	33.57
Kalman Filter, Yolo3	83.69	44.74	54.50	89.05	39.49	90.38	46.57	**35.56**
Kalman Filter, SSD 512	71.37	46.57	56.27	6.64	43.67	67.79	31.65	26.45
Max.Overlap+Opt.Flow Yolo3	63.22	49.12	25.39	96.51	22.45	49.08	63.94	18.21
Baseline TC, Yolo3	**84.84**	48.73	**74.57**	91.79	**44.29**	**93.97**	70.06	35.20

IDF1 (SEQ 4)								
Techniques	**C24**	**C25**	**C26**	**C27**	**C28**	**C29**	**C30**	**C31**
Max Overlap, Mask-RCNN	11.21	17.23	49.71	54.88	30.84	44.85	31.25	8.21
Max Overlap, Yolo3	12.63	14.32	49.72	38.91	32.75	36.60	41.29	5.89
Max Overlap, SSD512	13.95	15.82	42.93	58.96	48.65	33.45	37.03	4.71
Kalman Filter, Mask-RCNN	**56.25**	**59.08**	79.11	51.73	46.19	52.73	62.25	18.84
Kalman Filter, Yolo3	52.3	48.17	78.58	42.71	**59.04**	40.35	68.99	18.96
Kalman Filter, SSD 512	42.30	40.38	70.42	52.14	57.14	40.57	61.49	18.20
Max.Overlap+Opt.Flow Yolo3	12.63	17.86	49.70	42.68	32.91	41.42	41.75	5.14
Baseline TC, Yolo3	29.03	55.95	**83.63**	**82.49**	58.93	**57.82**	**73.61**	**63.84**

IDF1 (SEQ 4)										
Techniques	**C32**	**C33**	**C34**	**C35**	**C36**	**C37**	**C38**	**C39**	**C40**	**Average**
Max Overlap, Mask-RCNN	16.75	97.07	95.25	36.32	76.31	49.03	26.47	42.90	44.87	46.39
Max Overlap, Yolo3	19.37	85.18	79.32	38.92	72.49	43.19	19.15	40.36	45.11	42.59
Max Overlap, SSD512	14.58	89.17	92.91	34.85	70.74	52.93	24.07	29.18	**63.51**	45.15
Kalman Filter, Mask-RCNN	31.70	71.57	61.15	72.45	62.33	44.50	58.39	67.35	49.62	52.99
Kalman Filter, Yolo3	47.60	76.94	68.96	66.61	62.83	38.20	53.20	**80.73**	24.72	56.55
Kalman Filter, SSD 512	**54.43**	53.99	64.98	71.13	59.35	44.76	54.94	71.79	53.56	50.48
Max.Overlap+Opt.Flow Yolo3	18.87	85.15	73.33	38.62	72.47	43.20	20.77	37.93	46.20	42.99
Baseline TC, Yolo3	45.24	67.25	69.36	**77.61**	66.94	31.38	**61.11**	73.92	57.10	**63.96**

Table 6. MTSC Results using the Complete sequence 4. The baseline TC with Yolo3 is again the best option to use in sequence 4. Like in sequence 3, kalman filter is a very good alternative to the baseline.