Video Surveillance for Road Traffic Monitoring

Stanislaw Kajetan Morawski, Marc Perez, Claudia Baca and Joaquim Comas
Autonomous University of Barcelona
Barcelona, Spain
{stanislaw.morawski,marc.perezq,claudiabaca.perez, joaquim.comas}@e-campus.uab.cat

Abstract

In recent years, intelligent transportation systems (ITS) have become a fast-growing research topic. Furthermore, the rise of new technologies and computer vision have boosted important improvements in traffic management, playing a crucial role in transportation evolution. To accelerate the research and development of techniques in traffic road monitoring, the AI City Challenge has been created. Although the entire challenge is composed of four different tasks, in this paper, we deal with one particular task. The goal of this task is the tracking of vehicles across multiple cameras both at a single intersection and across multiple intersections. In this work, we present a novel spatial-metric approach for vehicles tracking in a multi-camera traffic scene. Moreover, we have adopted other well-established techniques for solving the tracking vehicle paradigm through a single and multi-camera approach.

1. Introduction

In recent years, research on intelligent transportation system (ITS) has become an attractive topic in academia and industry. To accelerate the research on the development of smarter transportation systems, the 4th edition of CVPR 2020 AI City Challenge is focused on 4 different challenges: vehicle counts by class at multiple intersections, city-scale multi-camera vehicle re-identification, city-scale multi-camera vehicle tracking and traffic anomaly detection. The target of this work is the city-scale single and multi-camera vehicle tracking (Track 3 of AI City Challenge). Specifically, the contributions of our work are as follow:

1. We represent a set of handcrafted feature-based and deep learning-based methods for the Multi-Target Single Camera Tracking (MTSC).
2. We propose several approaches to deal with the Multi-Target Multi-Camera Tracking (MTMC) introducing the strengths and weaknesses of each method and the different attempts.
3. We present our novel spatial-metric approach for vehicles tracking in a multi-camera traffic scene providing our results on the Track 3 of AI City Challenge.

2. Related work

In this section, we will provide a brief overview of existing works on different aspects of MTMC. Most of the current approaches to tracking follow the track-by-detection scheme in which objects are first detected, then associated within a single camera across frames, and finally those tracks are linked across cameras based on a plethora of features.

Object Detection. Since the emergence of [23], which framed object detection as regression as opposed to a classification problem, various similar approaches were proposed. Together with their tweaked variations, they constitute the current state of the art [12, 18, 24, 17]. We build our solution on top of fine-tuned Mask-RCNN [12] detections.

MTSC and **Re-Identification**. Most of the current state of the art tracking solutions are build on top of [5], which uses simple and well-studied techniques such as Kalman Filter and Hungarian algorithm. [31, 28, 26] provide a way to incorporate visual features into a system similar to [5]. [30] additionally takes temporal visual features into account.

MTMC. Thanks to advances in MTSC the multi-camera tracking is becoming more and more feasible. Apart from re-identification based purely on visual features, most solutions exploit spatial relations between cameras. Some works require completely unobstructed and overlapping views [1, 2, 6], others ignore overlaps or are able to leverage fusing data from partially overlapping views [34]. Some works map objects to ground plane [7, 8, 15, 22]. Travel time is also being exploited in [29, 33, 8].

3. Methodology

In this section, we will explain the dataset used in our experiments, including our data pre-processing steps. Further, we describe our proposed methods in order to to solve the multi-target tracking through single and multiple cameras.

3.1. Datasets

The provided dataset [27] contains 3.25 hours of videos collected from 40 cameras spanning across 10 intersections

in a mid-sized U.S. city. The dataset covers a diverse set of location types, including intersections, stretches of roadways, and highways. The dataset consists of a total of 666 vehicles annotated across five different scenarios. From these five scenarios, three of them are used for training and the remaining two scenarios are used for testing. In this work, because of computation capability, we only used the training set with a total length of 58.43 minutes and composed by scenarios 1, 3 and 4.

The second dataset consists of the same data as the previous dataset, however we created a new set of samples to train our metric model. For this task, we used the annotations files to extract the cropped image of each detected car from scenarios 1 and 4. Then, each vehicle was labelled as a different class along with each camera and scenario. By combining the scenarios 1 and 4, a total of 103 classes were created and utilized in the training of our metric model, which will be described in detail on section 3.3.3.

3.2. Multi-target single-camera tracking

As most of the state-of-the-art MTSC tracking methods, we have followed the tracking-by-detection paradigm. Initially, we generated the detected bounding boxes of the vehicles in all the scenes; we had several methods, on the one hand, background estimation methods that include our adaptive background estimation method, MoG(Mixture of Gaussian) method and KNN (K-Nearest-Neighbours) method. On the other hand, well-known object detection methods such as RetinaNet [17], Yolov3 [24], Mask-RCNN [12] and our fine-tuned Mask-RCNN. Additionally, we had to apply filtering to delete non-moving cars detected, since these cars were not going to be tracked. Once we had the detections of the moving vehicles, two different tracking methods were applied, our implementation of an Overlap method and a Kalman Filter for tracking.

Figure 1: Multi-target single camera example using Mask R-CNN fine tune technique with Kalman filter tracking.

3.3. Multi-target multi-camera tracking

3.3.1 Features matching

Feature matching is used in many applications such as three dimensional (3D) reconstruction, robotics, medical imaging. Among those applications, we also found object tracking; therefore, we have proposed a model to solve the AI City Challenge using a feature-based approach. Our proposal consists of computing the features of each car in one camera, so the key-points that represents the vehicle in that frame, and try to find that representation in the same frame of another camera, obtaining then the re-identification of the car in different scenes. The representation of each car has been done using SIFT [20], ORB [9] or SURF [11] descriptors, to found the correspondence between different cameras we have used a brute-force matching algorithm with which we obtained that quantity of matches between the vehicles. By computing this quantity of matches and the total key-points that represents a car, we extract the similarity between the different representations, then we threshold this parameter to dismiss matching errors. Although the model use comparison frame to frame from different cameras and we used a similarity threshold, many matches are detected wrongly making our method fail in most of the cases, we have realized that our proposal is too sensitive to the perspective shown of the car and his trajectory. Due to that and the constraints of the key-point representation, so the lack of robustness against illumination changes, occlusions, and blur. We considered that this feature-based approach was not the right solution for the re-identification task.

3.3.2 Color Histogram

Color histogram based tracking is one of the most simple methods of object tracking; it is being proved to be a computationally efficient and robust model for many cases. The goal in our case is to achieve the same robustness but tracking the same object from different views, then for this, we have followed the hypothesis of the previous proposal but using the histogram as features instead. The model, as mentioned, uses histograms as the representation of each car, which can be in 3D or 2D using the colorspaces; RGB, HSV, Grayscale and YUV. Then each car histogram is compared with all the car histograms in the same frame but in other cameras, by computing the distance which can be chi-squared, Bhattacharyya or by computing the correlation or intersection between them. We considered that the same car is shown in two perspectives if the distance between them is lower than a threshold established, we repeat this process to achieve our goal, the re-identification of a car in multiple cameras. Although using histograms as features is a really efficient way to represent the cars and it gives us a lot of information about the object of interest, it is not generalized enough to differentiate the same car in different scenes, so we have the same problem as the previous method, the proposal is too sensitive to the perspective shown of the car and the trajectory of it. Due to that and regardless, that was a more fitted method than feature matching; this approach was also dismissed as an accurate method for our task.

3.3.3 Metric learning

Metric learning is a common method employed in computer vision tasks such as image retrieval [16] [10], facial recognition [14] [19] or person and vehicle re-identification [13]

[4]. The goal of metric learning approach is to learn a feature space, also well-know as embedding space, in which the samples of the same class are close to each other, and the samples from other classes are far away. To differentiate each class from other different classes, we can distinguish two main loss functions for this goal: the contrastive loss and the triplet loss. For our vehicle re-identification task, we considered two different frameworks using both loss functions. In our first attempt, we adapted and trained a facial detection framework [21] based on a straightforward siamese network with a contrastive loss. During the training, we used the dataset mentioned in section 3.1. The objective of this first approach was the computation of the dissimilarity score to maximize the embedding distance between instances from different classes and minimize the distance of samples from the same class. After training, some experiments were tested to check the dissimilarity score between a set of samples. Although the tuned framework worked for some classes, the encoding distance was not entirely accurate to establish a global threshold to differentiate the different classes. Furthermore, the simplicity of the model caused other problems such as the loss of consistency when adding more cameras from different points of view. Due to the lack of generalization of this first approach, we considered exploring a more robust method which leads us to the second attempt using triplet loss [25]. In this second approach, we chose a Keras implementation for image classification, called EmbeddingNet [32]. The adopted model is composed by two main training steps: softmax pretraining and data mining. The first training stage correspond to Resnet-50 training using a cross-entropy loss followed by a semi hard online learning data mining which is used to improved the previous accuracy results. In table 1, we have summarized the most important parameters for each training stage: After training the model using the dataset

Param.	Training stage	
	Softmax pretrain	Data mining
Epochs	30	100
Learning rate	0.001	0.001
Batch size	8	8
Iterations	500	200
Pretrain weights	True	False
Loss func.	Cross-entropy	Triplet
Optimizer	Adam	rAdam
Early stopping	50 epochs	5 epochs

Table 1: EmbeddingNet training parameters for softmax pretrain and data mining.

explained on section 3.1 with the previous parameters we obtained a train accuracy of 100 % and 85,56 % on the validation set. In figure 2, is shown the embedding space before and after the explained training where each vehicle class is represented with a different color. As we can appreciate on the figure, after the training we notice a significant improvement on the encoding representation of each class which are differentiated from different classes.

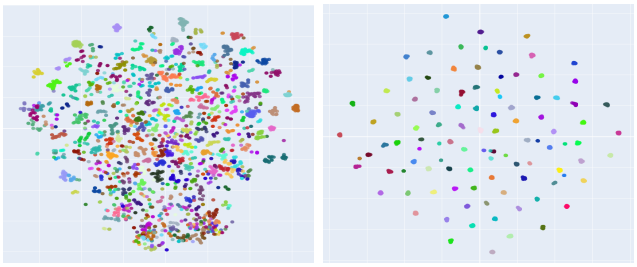

Figure 2: Embedding space before training (on the left) and after training (on the right).

3.3.4 Spatial coordinates

In this approach to the multi-camera tracking we use the homography matrices provided with the dataset to convert each detection to GPS coordinates. For each bounding box representing a detection for a given camera at a given timestamp we take the pixel situated at the middle laterally and 80% bottom of the image to convert it to GPS coordinates and then the detection is represented by the tuple (latitude, longitude) and the timestamp, we don't need to take into account from which camera this detection originated. We then merge all the detections from all cameras in chronological order using an Extended Kalman Filter (EKF). This filter was implemented from scratch considering a constant velocity model. Detections get assigned to an existing track if the distance to the predicted position of the track at the time of the detection is less than a threshold (in meters). When a detection is assigned it updates that track. And if the detection is not assigned a new track is created. Tracks are removed if no detection is assigned to them after a give time. As for the mathematical formulation of the EKF, each track is represented at a given stage k by a state x_k

$$x_t = \begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix}$$

Then the **state prediction** is performed according to the following equation that depend on the time difference $\Delta t_k = t_k - t_{k-1}$:

$$x_{k|k-1} = f(x_{k-1|k-1}, u_k) + w_k = f(x_{k-1|k-1}) + w_k$$

Where w_k is the process noise, a zero mean gaussian with covariance Q_k. And $x_{k|k-1}$ is the predicted state.

And the predicted covariance estimate is

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k$$

$$f(\begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix}, t_k) = \begin{pmatrix} x + \Delta t_k v_x \\ y + \Delta t_k v_y \\ v_x \\ v_y \end{pmatrix}$$

$$F_k = \frac{\partial f}{\partial x}\bigg|_{x_{k-1|k-1}} = \begin{pmatrix} 1 & 0 & 0 & \Delta t_k & 0 \\ 0 & 1 & 0 & 0 & \Delta t_k \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

To update a track given a detection d_k, we need to calculate the predicted measurement z_k for the predicted track at the time instance of the detections:

$$z_k = h(x_{k|k-1}) + v_k$$

Where v_k is the process noise, a zero mean gaussian with covariance R_k. And h is the function that maps the state of a predicted track to the expected measurement. In our case, simply:

$$h(\begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix}) = \begin{pmatrix} x \\ y \end{pmatrix}$$

Then, the general equations for EKF are:
Innovation or measurement residual

$$y_k = d_k - h(x_{k|k-1})$$

Innovation (or residual) covariance

$$S_k = H_k P_{k|k-1} H_k^T + R_k$$

Near-optimal Kalman gain

$$K_k = P_{k|k-1} H_k^T S_k^{-1}$$

Updated state estimate and covariance estimate:

$$x_{k|k} = x_{k|k-1} + K_k y_k \qquad P_{k|k} = (I - K_k H_k) P_{k|k-1}$$

Using diagonal matrices as initial $P_{0|0}$ and for the covariances of the processes ensures diagonal covariances. And that the predicted state going directly from t_1 to t_3 is the same as from going from t_1 to t_2 and then to t_3 We set the following covariance matrices (here in meters, they have to be transformed into latitude/longitude distances):

$$Q_k = \Delta t_k \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad P_{0|0} = \begin{pmatrix} 5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 100 \end{pmatrix}$$

$$R_k = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \end{pmatrix}$$

3.3.5 Spatial-similarity approach

The last method has some problems differentiating cars that were too close even if they were very different in aspect, so we decided to combine the spatial approach with the similarity approach based on metric learning (seen on Figure 3). So, building on the EKF in GPS coordinates approach, we keep an encoding for every track, this is initially the encoding of the detection that initializes the track. Then to associate a detection to the track we use a distance based on weighted sum of the distance in GPS coordinates (in meters) and the euclidean distance in the encoding. Then, if associated, we update also the encoding by adding 0.8 times the encoding stored in the track and 0.3 times the encoding of the detection.

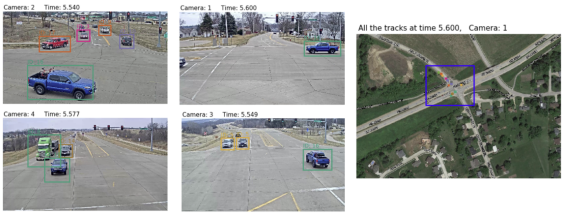

Figure 3: Multi-target multi-camera tracking using the proposed spatio-similar approach.

4. Results

In this section, we will briefly explain our final results obtained in MTSC and MTMC tasks. In MTSC, we used several methods which are shown in more detail in appendix A.In table 2 we can observe our best model for single-camera where we used our Mask R-CNN fine-tune as object detection followed by Kalman filter tracking and our filtering preprocessing. In this task, we only used the IDF1 metric, obtaining a 65.9% on average from the three tested scenarios. Although the results are quite similar, our best results where get on scenario four with 68.55% while scene three was the most challenging with a 61.61%.

	S01	**S03**	**S04**	**Avg.**
IDF1	67.54 %	61.61 %	68.55 %	65.9 %

Table 2: Multi-target single-camera tracking results

In the MTMC task, we used the same object detection as the MTSC, but our tracking was performed using our proposed spatial-embedding approach. In table 3, we appreciate the results obtained for each scene using the challenge metrics based on [3]. On average, our final result was a 51.2 % on IDF1 with similar precision and recall results around 75% approximately.

Seq.	**IDF1**	**IDP**	**IDR**	**Precision**	**Recall**
S01	57.7 %	68.1 %	67.3 %	81.9 %	61.4%
S03	43.5 %	48.5 %	43.3 %	73.2 %	73.9 %
S04	52.3 %	76.2 %	54.6 %	81.9 %	75.4 %
Avg.	51.2 %	64.3 %	55.1 %	79.0 %	70.2 %

Table 3: Multi-target multi-camera tracking results

5. Conclusions

We have presented many different approaches to building components of a traffic monitoring system, with object detection and tracking based on Kalman filters being the most important. We have built a traffic monitoring system from multiple cameras using an EKF on GPS coordinates that not only allows to retrieve the id of each detection, but it also provides a robust method to get the position of the detected cars in the real world merging the information from all cameras. This last feature is what makes our solution interesting for a real application. The system benefits from both spatial information and similarity information to improve the performance and robustness. At the same time it is very easy to add extra sensors, even lidars and radars, not only cameras.

Our method has some clear advantages over the Recurrent Neural Network (RNN) based solutions. Using Kalman filters for multiple object tracking makes it very easy to add more sensors while providing robustness and explainability required for production systems. It also dismisses the need of having a large dataset and expensive computational resources to train the model.

References

[1] M. Ayazoglu, B. Li, C. Dicle, M. Sznaier, and O. I. Camps. Dynamic subspace-based coordinated multicamera tracking. In *2011 International Conference on Computer Vision*, pages 2462–2469, 2011.

[2] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua. Multiple object tracking using k-shortest paths optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1806–1819, 2011.

[3] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008:1–10, 2008.

[4] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016.

[5] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. *2016 IEEE International Conference on Image Processing (ICIP)*, Sep 2016.

[6] M. Bredereck, X. Jiang, M. Körner, and J. Denzler. Data association for multi-object tracking-by-detection in multi-camera networks. In *2012 Sixth International Conference on Distributed Smart Cameras (ICDSC)*, pages 1–6, 2012.

[7] Yinghao Cai and Gérard Medioni. Exploring context information for inter-camera multiple target tracking. pages 761–768, 03 2014.

[8] Kuan-Wen Chen, Chih-Chuan Lai, Pei-Jyun Lee, Chu-Song Chen, and Yi-Ping Hung. Adaptive learning for target tracking and true linking discovering across multiple non-overlapping cameras. *IEEE Transactions on Multimedia*, 13:625–638, 08 2011.

[9] K. Konolige E. Rublee, V. Rabaud and G. R. Bradski. Orb: An efficient alternative to sift or surf. *International Conference on Computer Vision(ICCV)*, 2011.

[10] Weifeng Ge. Deep metric learning with hierarchical triplet loss. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[11] T. Tuytelaars H. Bay, A. Ess and L. V. Gool. Surf: Speeded up robust features. *Computer Vision and Image Understanding(CVIU)*, 110(3):346–359, 2015.

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. *arXiv arXiv:1703.06870*, 2017.

[13] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017.

[14] Junlin Hu, Jiwen Lu, and Yap-Peng Tan. Discriminative deep metric learning for face verification in the wild. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[15] Cheng-Hao Kuo, Chang Huang, and Ram Nevatia. Inter-camera association of multi-target tracks by on-line learned appearance affinity models. pages 383–396, 09 2010.

[16] Z. Li and J. Tang. Weakly supervised deep metric learning for community-contributed image retrieval. *IEEE Transactions on Multimedia*, 17(11):1989–1999, 2015.

[17] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*, 2017.

[18] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, page 21–37, 2016.

[19] Xiaofeng Liu, B. V. K. Vijaya Kumar, Jane You, and Ping Jia. Adaptive deep metric learning for identity-aware facial expression recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[20] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004.

[21] James Loy. *Neural Network Projects with Python: The ultimate guide to using Python to explore the true power of neural networks through six projects*. Packt Publishing Ltd, 2019.

[22] Dimitrios Makris, Tim Ellis, and James Black. Bridging the gaps between cameras. volume 2, pages II–205, 01 2004.

[23] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[24] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv arXiv:1804.02767*, 2018.

[25] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[26] Zheng Tang and Jenq-Neng Hwang. Moana: An online learned adaptive appearance model for robust multiple object tracking in 3d. *IEEE Access*, 7:31934–31945, 2019.

[27] Zheng Tang, Milind Naphade, Ming-Yu Liu, Xiaodong Yang, Stan Birchfield, Shuo Wang, Ratnesh Kumar, David Anastasiu, and Jenq-Neng Hwang. Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8797–8806, 2019.

[28] Z. Tang, G. Wang, H. Xiao, A. Zheng, and J. 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 Recognition Workshops (CVPRW)*, pages 108–1087, 2018.

[29] Jiuqing Wan and Liu Li. Distributed optimization for global data association in non-overlapping camera networks. pages 1–7, 10 2013.

[30] Gaoang Wang, Yizhou Wang, Haotian Zhang, Renshu Gu, and Jenq-Neng Hwang. Exploit the connectivity: Multi-object tracking with trackletnet, 2018.

[31] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric, 2017.

[32] Rauf Yagfarov, Vladislav Ostankovich, and Aydar Akhmetzyanov. *Traffic Sign Classification Using Embedding Learning Approach for Self-driving Cars*, pages 180–184. 04 2020.

[33] Shu Zhang, Elliot Staudt, Tim Faltemier, and Amit Roy-Chowdhury. A camera network tracking (camnet) dataset and performance baseline. *Proceedings - 2015 IEEE Winter Conference on Applications of Computer Vision, WACV 2015*, pages 365–372, 02 2015.

[34] Shu Zhang, Yingying Zhu, and Amit Roy-Chowdhury. Tracking multiple interacting targets in a camera network. *Computer Vision and Image Understanding*, 134, 05 2015.

A. Multi-target single-camera tracking results

Detector	Tracker	IDF1 - S03	IDF1 - S01	IDF1 - S04
Adap ours	Overlap	0.2559	-	-
	Kalman	0.4494	-	-
MOG	Overlap	0.3217	-	-
	Kalman	0.4907	-	-
KNN	Overlap	0.3521	-	-
	Kalman	0.4805	-	-
YOLO V3	Overlap	0.2815	0.2473	0.3263
	Kalman	0.4251	0.6162	0.4381
RetinaNet	Overlap	0.2678	0.2751	0.3058
	Kalman	0.4011	0.5801	0.3916
Mask R-CNN + filtering	Overlap	0.3701	0.2504	0.4565
	Kalman	0.5266	0.5499	0.6332
Mask R-CNN (FT) + Filtering	Overlap	0.2638	0.6754	0.4785
	Kalman	0.6161	0.6754	0.6856

Table 4: Multi-target single-camera tracking results