M6: Video Analysis for traffic monitoring

Jonatan Poveda, Joan Francesc Serracant, Ferran Pérez, Martí Cobos
Universitat Autònoma de Barcelona
Barcelona, 08193, Spain
{jonatan.poveda, joanfrancesc.serracant, ferran.perez, marti.cobos}@e-campus.uab.cat

Abstract

This document summarizes the research performed and algorithms developed by our team regarding the Video Analysis for traffic monitoring project developed during the Video Analysis *module in the framework of the* Master in Computer Vision Barcelona*. The main objective of the project is to develop a computer vision algorithm capable of tracking vehicles from multiple surveillance cameras placed on the street. This document describes the algorithms developed, their evaluation, conclusions and further discussion.*

1. Introduction

This document is a report of the project developed as the practical part of the *Video Analysis* module from the *Master in Computer Vision Barcelona*. The goal of this project is to implement and assess state-of-the-art algorithms to perform multi-target multiple camera tracking (MTMC), in other words, track vehicle position from multiple cameras and time-stamps by analyzing video streams from surveillance cameras. In particular, video sequences from traffic surveillance cameras available for the *2019 AI CITY CHALLENGE* have been analyzed. These sequences have already been analyzed using several state of the art algorithms [24], which serve as a benchmark for the challenge.

MTMC consists on the detection and tracking of vehicles from several cameras. It can be formed by a combination of 2 algorithms: (1) Multi-target single camera tracking (MTSC) and (2) Image-based vehicle re-identification based on vehicle detections with added information from spatio-temporal domain. MTSC could be considered as a simplification of this task as it consists on tracking vehicles within a single camera. This task is usually implemented by algorithms which follow the tracking by detection paradigm. CNN based detectors (e.g.: YOLO, SSD...) are used in combination and tracking methods, such as Kalman filter based tracking algorithms, to generate the vehicle track from the video stream from one camera.

This paper is organised as follows. In section 2, the state-of-the art architectures for vehicle tracking and re-identification are presented, sections 3, 4 and 5 layout the work done in MTSC and MTMC, respectively, explaining the methodology followed and tests done. Finally, in section 6 we extract conclusions for the implemented computer vision task and the overall system and propose some ideas for further improvements.

This project is developed in Python using mainly the OpenCV library, additionally a neural network for distance metric learning has been fine tuned using PyTorch framework. The code developed during this course, is fully published on GitHub [3] under a GPL-3.0 License.

2. Related Work

Over the past years, the problem of object tracking has been tackled first to track multiple object seen by a single camera (MTSC). Only recently, some research has been done on tracking multiple objects captured by multiple cameras (MTMC). Let's review some previous work and research on both topics.

2.1. Single camera tracking

First attempts to track vehicles like [13] followed a *Model based tracking* approach where a 3D model of vehicles to be tracked was needed and occlusions were a serious problem.
A different approach is *Region based tracking* as in [12]. In this approach, first, background subtraction is performed following a Kalman filter-based model of the background that adapts to changes on illumination. Connected regions in the foreground are associated to each vehicle using a cross-correlation measure. This approach experienced problems on occlusions as well.
An improvement to previous approaches was introduced in [14] where *Active Contours* were used to enforce tracked objects to have a specific shape of the bounding contour. Again partially occluded objects posed a problem, specially during initialization of contours with objects usually entering the scene partially occluded.

Feature based tracking, as in [18], identifies features (corners) on each frame and tracks them across time, backed up by a Kalman filter to reduce errors. In this approach, an additional step is needed in order to group features belonging to the same vehicle. Using features proved to have good performance in congested traffic with occlusions.

2.2. Multiple camera tracking

Tracking objects on multiple cameras seems to be a field of limited research in recent years. Latest attempts, like [25] and [19], follow a similar two step approach. First, state-of-the-art *Deep Convolutional Neural Networks* perform detection of desired objects at frame level for each camera separately. Detections of the same vehicle are grouped into tracklets using spatio-temporal constraints. In a second step, a task known as *Object Re-identification* tries to associate the appearance of each object instance in all the cameras using several techniques such as feature extraction, histogram comparison, or recognition of unique parts like license plates in the case of vehicles.

3. Multi-target single camera tracking

In this section we detail the algorithms tested for multi-target single camera tracking. To perform tracking of several cars from a video recorded by a single camera, we propose a two-step solution: first, *Object Detection*, consists in a frame-by-frame detection of all cars resulting in a set of regions of interest (ROI) containing cars for each video frame; second, *Object Tracking*, tries to establish correspondences between ROIs throughout consecutive video frames. Two ROIs are related if they belong to the same object. Finally, a set of ROIs for the same object ordered along time is denoted as a *tracklet*.

3.1. Object detection

As mentioned above, all tracking methods presented use detections given by state-of-the-art object detectors such as YOLO3 [20], Mask-RCNN [6], RetinaNet [15] and SSD512 [16]. With the exception of RetinaNet, detections are pre-computed for all sequences so we use them directly. It would like to compute RetinaNet for all cameras to see if it improves the performance of our methods given that it uses a more modern backbone ResNeXt but we run it only for one camera due to time constrains.

3.2. Object tracking

3.2.1 Region Overlap

Given frame-by-frame detections, region overlap searches for each region of interest (ROI) in the current frame the best matching ROI on the previous frame using the maximum Intersection over Union (IoU). If a match is found, the ROI is assigned the same track ID. Otherwise, we assign it a new ID. We repeat this procedure for each pair of frames in the sequence. Figure 1 shows an example of the maximum IoU for the ROIs of two consecutive frames.

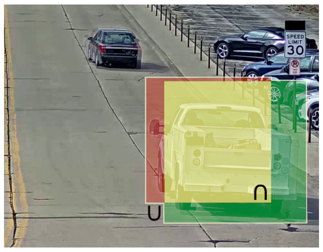
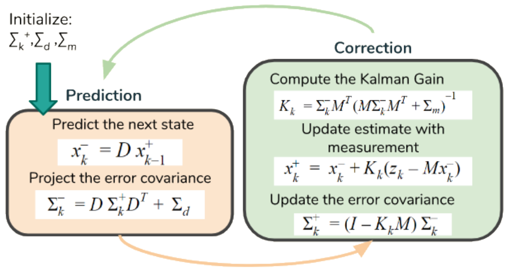


Figure 1. Example of ROIs with maximum IoU. In red, green and yellow, ROI at current and previous frame and their intersection.

3.2.2 Kalman filter

Secondly, we implemented a method based on the Kalman filter [11] which is formulated as a two-step estimation problem. First, we predict the position of the vehicle in the next frame; secondly, given a new observation, we refine the estimation by combining it with the prediction. By doing so and assuming that the measurements are close to the initial model (i.e.: same variance), the more iterations we do, the smaller the variance of the prediction becomes (the more certain we are).

The base Kalman filter can only deal with linear models which is compatible with highway-like traffic (mostly linear) but not so appropriate for residential traffic where vehicles often accelerate, brake or turn. Moreover, with the Kalman filter we have to explicitly define a motion model for the object being tracked. In our case, we test two models: constant velocity and constant acceleration.



Initialize:
$\Sigma_k^+, \Sigma_d, \Sigma_m$
Prediction
Predict the next state
$x_k^- = D x_{k-1}^+$
Project the error covariance
$\Sigma_k^- = D \Sigma_k^+ D^T + \Sigma_d$
Correction
Compute the Kalman Gain
$K_k = \Sigma_k^- M^T (M \Sigma_k^- M^T + \Sigma_m)^{-1}$
Update estimate with measurement
$x_k^+ = x_k^- + K_k(z_k - M x_k^-)$
Update the error covariance
$\Sigma_k^+ = (I - K_k M) \Sigma_k^-$

Figure 2. Kalman filter steps prediction and correction

3.2.3 Optical Flow

The apparent motion between consecutive frames (Optical Flow) can be used to estimate the position of a bounding box in the next frame. There are several methods to estimate optical flow, from classical approaches (variational-based) like Horn-Schunk [8], Lucas Kanade [17], PyFlow [1], TV-L1 [28], Epicflow [21] to other methods which employ self-supervised CNNs to learn to predict the flow such as: PWC-Net [23], FlowNet [2], FlowNet2 [10], LiteFlowNet [9].

Particularly, given two consecutive frames and the detections at the current frame, we follow these steps: 1) Estimate the optical flow between $I_t \to I_{t+1}$ (PyFlow); 2) Consider only the flow inside the ROI. To ensure the entire vehicle is included in the box, we give a margin of 20% to the ROI bounding box; 3) Binarize flow magnitude in the bounding box to only account for the motion of foreground pixels (vehicle); 4) Estimate the vehicle center on the next frame using the mean optical flow in the foreground region. An illustration of this procedure is shown in Figure 3.

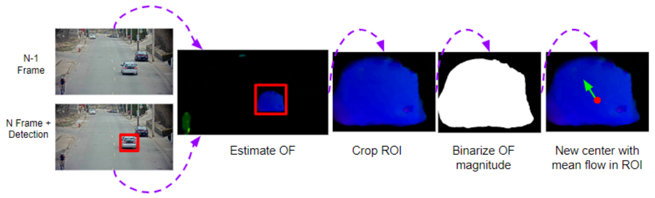


Figure 3. Using Optical Flow for object tracking

3.3. Results

Initially we run our tracking algorithms on sequence *S03*, camera *c010*. Each team annotated a part of this sequence including parked cars (not included in the challenge annotations). However, to assess our algorithms on all training cameras (*S01, S03, S04*), we use the challenge annotations. We show results for the tracking methods presented above for the training cameras with different combinations of detectors. Additionally, we provide as a reference the best-performing method shown in the AICity challenge paper which employs SSD detections and the Track Clustering (TC) tracker algorithm. This method uses motion, temporal and appearance queues of tracks to decide whether they should be merged/split. It does so by minimising a loss which is composed of the weighted sum of some terms: colour cost, split cost, temporal smoothness, etc.

In order to properly evaluate our methods against the challenge annotations, we remove parked cars by simply thresholding the absolute difference between the ending and starting coordinates of the bounding box of a tracklet. To cope with the different video resolutions present in the dataset, we define the threshold as a percentage of the image width and height instead of a number in pixels. To simplify things, we use one threshold for all the cameras of the sequences to test. To select an optimal threshold, we do a coarse grid search in a validation sequence different that the one to test. Figure 4, shows the IDF1 score for a range of threshold values in sequence *S01*, in order to later test on *S03*.

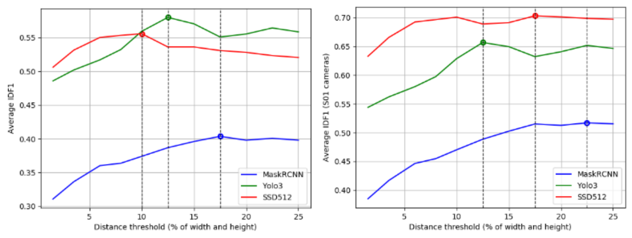


Figure 4. Validation of the optimal threshold in sequence S01

	IDF1						
Method/Camera	c010	c011	c012	c013	c014	c015	avg
RetinaNet+Kalman[1]	45.6%	–	–	–	–	–	–
RetinaNet+Flow[1]	45.7%	–	–	–	–	–	–
RetinaNet+Overlap[1]	46.5%	–	–	–	–	–	–
RetinaNet+Overlap[2]	75.6%	–	–	–	–	–	–
YOLO3+Overlap[3]	31.6%	10.7%	0.6%	35.1%	55.5%	2.0%	22.6%
YOLO3+Kalman	78.2%	55.3%	21.5%	65.4%	63.3%	4.3%	48.0%
SSD+Overlap	**87.3%**	69.6%	**56.9%**	79.0%	68.6%	**15.8%**	**62.9%**
TC+SSD(reference)	80.0%	**72.1%**	10.4%	**86.6%**	**70.8%**	3.9%	54.0%

Table 1. IDF1 score on sequence S03. [1] denotes methods tested with team-annotated sequences (w. parked cars); [2] specifies the method was *not* validated to find an optimal threshold but a default one was used; [3] refers to methods without thresholding.

Note that the simple Region Overlap method outperforms the Kalman constant velocity model. This is mainly due to Kalman's initialization (zero acceleration) and further adaptation, which is less capable to follow the vehicles movement, specially when they accelerate, brake or turn. The best threshold is 17.5%, which is quite large, implying that in S01, tracks are relatively long. Moreover, the best-performing detector is SSD, conforming with the results shown in the challenge paper [24].

In Table 1 we compute the IDF1 metric for sequence *S03*. Notice that the simple thresholding scheme on top of accurate detections and region overlap gives very competitive results. In fact, we obtain better average IDF1 score than TC+SSD [26] (best-performing method on the challenge's paper [24]. Nevertheless, we obtain worse results in sequence *S01* (Table 2). After an in-depth side-by-side analysis, we observe that S03 contains more static and parked cars than S01. Moreover, despite the benefits of the thresholding, it sometimes removes short tracks (below thresh.) of the same object that should be merged with a longer track (above threshold) based on their similarity. We also report worse results on the 25 cameras in S04 (7% below reference) where more critical scenarios for the thresholding are present which yield worse results. Finally, we must consider that a fixed threshold for an entire sequence is only optimal using mean and maybe fine-tuning it based on the mean/median track motion would be helpful. One idea to explore is adding a clustering scheme for tracks similar to TC cluster. One drawback of TC cluster we observed by testing the source code in Matlab [4] is that it takes a lot of time to compute the tracks given the detections (about 1.5 seconds per frame so about 54 minutes for each camera).

	IDF1					
Method/Camera	c001	c002	c003	c004	c005	avg
SSD+Kalman[1]	27.8%	17.7%	23.7%	18.5%	–	21.9%
MaskRCNN+Kalman[1]	32.2%	44.8%	33.8%	48.4%	23.1%	36.5%
SSD+Overlap[1]	48.4%	33.0%	35.8%	33.9%	–	37.8%
YOLO3+Kalman[1]	39.3%	58.6%	52.8%	37.5%	24.5%	42.5%
MaskRCNN+Overlap[1]	40.7%	51.1%	39.8%	55.7%	29.4%	43.3%
YOLO3+Overlap[1]	39.3%	59.3%	53.9%	39.9%	25.5%	43.6%
SSD+Overlap	**87.3%**	**77.2%**	**70.8%**	62.2%	**53.1%**	66.4%
TC+SSD(reference)	**72.2%**	73.6%	64.8%	**77.3%**	**53.1%**	**68.3%**

Table 2. IDF1 score on sequence S01. [1] denotes methods without filtering of parked cars by thresholding

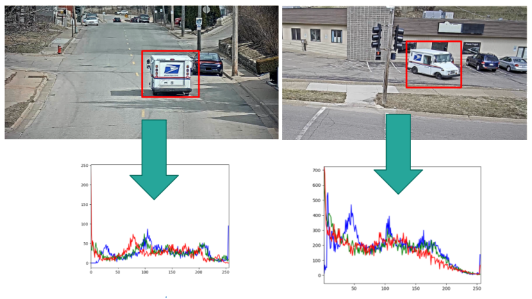


Figure 5. Colour histograms computed from two instances of the same object from different cameras

For the tested sequence, if we only output a text file with all the tracks (without plotting detections) for one camera, our best-performing method takes about 5 seconds to process the *whole* sequence, that is, around 2.33 ms per frame.

4. Improvements on single camera tracking

Improvements on single camera tracking are presented in this section. Discontinuities on the detection of objects throughout frames cause long trajectories of the same object instance to be split into different tracklets, thus being considered as different objects. In order to solve this problem, we propose to merge tracklets based on a metric distance on objects.

We used the image of largest region of interest of each tracklet (ensuring the maximum spatial information available of that object) to compute distance between tracklets using two different metric distances: colour histogram comparison (Fig. 5) and a learnt metric distance.

4.1. Colour histogram comparison

A cross-correlation between RGB colour normalized histograms is used (Eq. 1).

$$(H_1, H_2) = \frac{\sum_I (H_1(I) - \bar{H_1})(H_2(I) - \bar{H_2})}{\sqrt{\sum_I (H_1(I) - \bar{H_1})^2 \sum_I (H_2(I) - \bar{H_2})^2}}, \qquad \bar{H_k} = \frac{1}{N} \sum_J H_k(J) \tag{1}$$

4.2. Learnt metric distance

The other approach is to train network to learn the distance between car instances. There are multiple choices of nets candidates to perform this operation, for example FaceNet [22] which is based on Siamese Networks, [7] based on Triplet Networks, or [5] and Proxy-NCA [27] which are in addition tested on Car196 dataset. We selected the last work, Proxy-NCA [27], as it has a performance on the state-of-the-art on a related dataset, the authors claimed fast convergence and its available an implementation on Pytorch hosted in GitHub `https://github.com/dichotomies/proxy-nca`.

In short, networks based on Distance Metric Learning for images are in general convolutional neural networks trained using weak labels. The network is trained to decide if some samples are similar o dissimilar. In our case, a triplet network, the input are sets of three samples: an anchor or reference, a positive and a negative. Given an anchor, a positive sample is anyone belonging to same class as the anchor, and just the opposite on negative samples. The network learns a projection of the input samples into an embedding space where a distance, denoting similarity, between them can be computed.

We used all the five cameras from Sequence01 to train the network. As we had a limited processing capabilities, we trained the network by five stages of fine-tuning. We used the pre-trained weight of BNInception and then we fine-tuned the network for each camera, consecutively. We used 60 different classes of cars for training and the other for validation, on each train stage. All the other parameters of the net were left by default. The distance between two samples is done by projecting two crops into the learnt embedding space and then computing the distance between them in that space.

	IDF1						
Method/Camera	c010	c011	c012	c013	c014	c015	avg
SSD+Overlap	**87.3%**	**69.6%**	**56.9%**	**79.0%**	**68.6%**	**15.8%**	**62.9%**
SSD+Overlap+Hist	**87.3%**	52.1%	21.3%	54.3%	65.8%	**15.8%**	49.0%
SSD+Overlap+PNCA	85.9%	63.2%	28.7%	51.5%	**68.6%**	**15.8%**	52.0%

Table 3. IDF1 score on sequence S03 using tracklet merging at single camera level

4.3. Results

We applied our tracklet merging techniques to our best results from section 3 with SSD detections and tracking by region overlap. As shown in Table 3, implemented distances fail as a valid metric to compare vehicles. As expected, learnt metric distance performs better than histogram comparison.

5. Multi-target multiple camera tracking

In this section we detail the algorithms tested for multi-target multi camera tracking. Following the same procedure as in single camera tracking, we first detect objects and them relate them to get their tracklets. Then we add an additional step which relate various tracklets from different cameras.

Method	IDF1
Histogram	**54.4%**
PNCA	50.1%

Table 4. Multi-cam tracking results (IDF1) on the sequence S03 on cameras c010 and c011

We used our implementation of metric distances used in Section 4 in order to establish correspondences between objects from different cameras.

5.1. Results

Results are shown in Table 4 and again prove that implemented distances fail to solve the problem of re-identification for multiple camera tracking. Nevertheless, we are very confident that better results can be achieved by learning a metric if we continue working on this problem.

6. Conclusions

During this project our team has implemented an algorithm for multi-target multiple camera tracking. This algorithm is based on two main blocks: multi-target single camera tracking and image based vehicle re-identification. The *2019 AI CITY CHALLENGE* dataset has been used to assess system performance.

For multi-target single camera target, several methods have been implemented successfully: region overlap, Kalman filters and optical flow based tracking. Optical flow based tracking has been discarded due to the fact that it is computationally expensive and cannot be computed in real-time. Region Overlap yields slightly higher IDF1 scores, however Kalman filter based generated tracks are smoother as measurement variability is taken into account. Optimal results could potentially be obtained by fine-tuning the Kalman filter models.

As for vehicle re-identification, histogram comparison and Deep Metric learning methods have been implemented. Histogram comparison is a simple method to match vehicles (either between different cameras or within the same camera) which yields acceptable results with little threshold training effort. On the other side, Deep Metric learning is more powerful method to re-identify vehicles, however the training stage is far more complex as a neural network must be trained. We were not able to implement Deep Metric learning successfully on the MTMC task, additional work on the neural network training shall be performed. Considering related work in Section 2, implemented techniques are not enough to solve re-identification on their own and must be used together with other techniques such as feature extraction and comparison, tracklet clustering, license plate recognition, etc.

MTMC for vehicle tracking is a very challenging task. Non-homogeneous set of cameras with different lighting conditions and disjoint views of the 3D scene add complexity to the problem. Extracting good features to uniquely identify vehicles is not an easy task and may be useless if those features cannot be seen from another cameras point of view with another 3D pose of the same vehicle.

As a future work, Kalman filters cold be fine tuned to obtain better tracklets. We could also use feature based tracking and tracklet clustering to refine object trajetories. Finally we could take advantage of spatio-temporal information (camera calibration and frame time-stamp) from the video-sequences for the vehicle re-identification task.

References

[1] Thomas Brox, Andrs Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. pages 25–36. Springer, 2004.

[2] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[3] fperezgamonal, fserracant, jonpoveda, and marticobos. mcv-m6-video/mcv-m6-2019-team4, Apr. 2019.

[4] GaoangW. Gaoangw/tc_tracker, Apr. 2018.

[5] Ben Harwood, B. G. Vijay Kumar, Gustavo Carneiro, Ian Reid, and Tom Drummond. Smart Mining for Deep Metric Learning. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[6] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2980–2988, 2017.

[7] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9370(2010):84–92, 2015.

[8] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *ARTIFICAL INTELLIGENCE*, 17:185–203, 1981.

[9] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *CVPR*, pages 8981–8989. IEEE Computer Society, 2018.

[10] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[11] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[12] K. . Karmann. Time-recursive motion estimation using dynamical models for motion prediction. In *[1990] Proceedings. 10th International Conference on Pattern Recognition*, volume i, pages 268–270 vol.1, June 1990.

[13] D. Koller, K. Daniilidis, and H. H. Nagel. Model-based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer 11263on*, 10(3):257–281, Jun 1993.

[14] Dieter Koller, Joseph Weber, and Jitendra Malik. Robust multiple car tracking with occlusion reasoning. In Jan-Olof Eklundh, editor, *Computer Vision — ECCV '94*, pages 189–196, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

[15] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2999–3007, 2017.

[16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In *ECCV*, 2016.

[17] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[18] Philip F. McLauchlan, David Beymer, Benjamin Coifman, and Jitendra Malik. A real-time computer vision system for measuring traffic parameters. In *CVPR*, 1997.

[19] Hang Qiu, Xiaochen Liu, Swati Rallapalli, Archith J. Bency, Kevin S. Chan, Rahul Urgaonkar, B. S. Manjunath, and Ramesh Govindan. Kestrel: Video analytics for augmented multi-camera vehicle tracking. *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 48–59, 2018.

[20] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[21] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow. In *Computer Vision and Pattern Recognition*, 2015.

[22] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A unified embedding for face recognition and clustering. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June:815–823, mar 2015.

[23] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. 2018.

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

[25] Zheng Tang, Gaoang Wang, Hao Xiao, Aotian Zheng, and Jenq-Neng Hwang. Single-camera and inter-camera vehicle tracking and 3d speed estimation based on fusion of visual and semantic features. pages 108–1087, 06 2018.

[26] Zheng Tang, Gaoang Wang, Hao Xiao, Aotian Zheng, and Jenq-Neng Hwang. Single-camera and inter-camera vehicle tracking and 3d speed estimation based on fusion of visual and semantic features. 2018.

[27] Xianghua Ying and Hongbin Zha. No Fuss Distance Metric Learning using Proxies. *Proceedings - International Conference on Pattern Recognition*, 1:535–538, 2017.

[28] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l1 optical flow. In *Proceedings of the 29th DAGM Conference on Pattern Recognition*, pages 214–223, Berlin, Heidelberg, 2007. Springer-Verlag.