Video Surveillance for Road Traffic Tracking

Alba Herrera
albaherrerapalacio@gmail.com

Jorge López
jorge.lopezfu@e-campus.uab.cat

Marc Nuñez
marc.nunezu@e-campus.uab.cat

Nilai Sallent
nilai.sallent@e-campus.uab.cat

Abstract

This paper discusses car tracking with different cameras and sequences. Different methods are studied for id tracking in the same camera and sequences such as maximum overlap, optical flow and Kalman filters. Deep learning techniques, that relate the id's of the cars tracked in different cameras and sequences, have also been used. The goal of this project is to design, implement and evaluate these methods on the AICity Challenge dataset.

1. Introduction

Object detection has become an important part of deep learning. From autonomous driving to road traffic monitoring, differentiating and identifying objects allows those processes to be performed properly. On the latter, object tracking consists in the identification of an object through video frames. All objects in a frame are detected and an ID is given to each one. On the following frames the object is searched, if it disappears the ID is dropped and if a new object appears a different ID is assigned to it. This allows the detection and tracking of surrounding vehicles on an autonomous driving system.

Our work consists in vehicle tracking from diverse surveillance cameras located in an intersection. The vehicles are identified and tracked through the different cameras with diverse computer vision techniques.

2. State of the art

On-road vehicle tracking has been extensively modelled. In [17] vehicles are modelled with Haar-like features and static AdaBoost is trained for observation. [14] handles both occlusions and varying viewpoint, training is done with a static SVM that uses a Deformable-part-based model for occlusions and multi-viewpoint models for the latter problem. Both in [1] and [2] implementations of the Kalman filter are used for the object tracking. The former uses an extended version and interest points are used to model the vehicles. The latter implementation models vehicles with correlation filters and a unscented version of the Kalman filter. Particle filter is used by Danescu *et al.* [5] and Hermes *et al.* [9] for the vehicle detection and viewpoints variation. Kristoffersen *et al.* [10] use a part-based model for vehicle detection and a SVM holistic model by means of a Markov Decision Process and Track Learn Detect for tracking. In [19], the scenario consists of a set of surveillance cameras. Vehicles are modelled using semantic parts and Markov Random Fields and Kalman filters are used for tracking.

3. Dataset

The dataset provided from AICity Challenge [18] comes from multiples traffic cameras placed in a city in United States, as well as from state highways in Iowa, Figure 1.

- Nearly 3 hours of synchronised videos synchronously captured from multiple vantage points at various urban intersections and along highways.
- The videos are captured at 960p or better, 10fps.
- Detections for each frame provided by object detection Neural Networks: Mask RCNN, YOLOv3, SSD512.
- Metadata about the collected videos, including GPS locations of cameras, camera calibration information and other derived data from videos.
- Ground Truth for each frame and region of interest for each camera.

4. Method

4.1. Multi-object single-camera tracking

In this subsections, the methods implemented for single-camera tracking and a filter to remove static cars from the tracking are explained.

4.1.1 Kalman filter based tracking

Alex Bewley *et al.* [3] implemented the pragmatic approach of Simple Online and Realtime Tracking. Although their

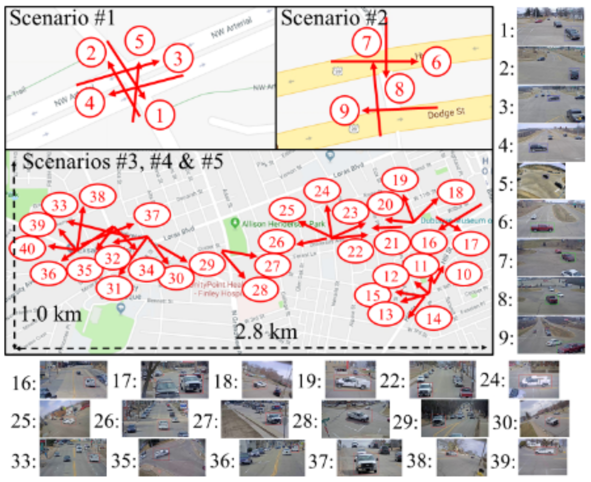


Figure 1: Urban environment and camera distribution, red arrows denote location and direction of cameras. Some examples of camera views are shown. Source: [18]

approach only uses a rudimentary combination of Kalman Filter and Hungarian algorithm for the tracking, they are able to achieve results comparable to state-of-the-art techniques.

The pre-computed bounding box detections of the vehicles are associated based on the IoU with the Hungarian algorithm.

4.1.2 Overlap based tracking

A simple tracking method that compares the intersection over Union (IoU) [15] to identify detections by their overlap between frames Figure 2. The following algorithm has been implemented:

For each detection of the current frame (bounding box):

1. Compute the IoU with each of the previous frame detections, Figure 2.
   - If IoU $>$ threshold assign the same id.
   - Otherwise generate a new id.

4.1.3 Optical flow based tracking

In order to track cars using optical flow, the following algorithm has been implemented:

For each frame with its detections (bounding boxes):

1. Obtain the best features to track inside the detections, using Shi-Tomasi corner detector.

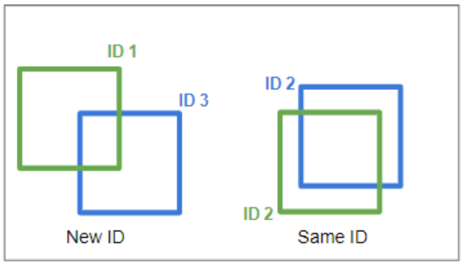


Figure 2: Identification system by overlap. Green bounding boxes correspond to detected objects at frame N, while blue ones are at frame N+1

2. Compute the optical flow of the features with the next frame using multi-scale Lucas Kanade algorithm, Figure 3.
3. Compute the average of the optical flow vectors inside each detection.
4. Perform the bounding box movement in the direction of the optical flow.
5. Assign tacking indices using the overlap:

   5.1. Compare the displaced bounding box with the detected one in the next frame, Figure 2.
   - If IoU $>$ threshold assign the same id.
   - Otherwise generate a new id.

Lucas-Kanade Optical Flow [12, 13] a differential method for optical flow estimation, developed by Bruce D. Lucas and Takeo Kanade. It solves the basic optical flow equations for all the pixel in a local neighbourhood assuming that the flow is constant over it.

4.1.4 Filter to remove static cars

Since the ground truth only considers moving cars, the system should be able to distinguish and remove not moving cars, to do so the following algorithm has been implemented:

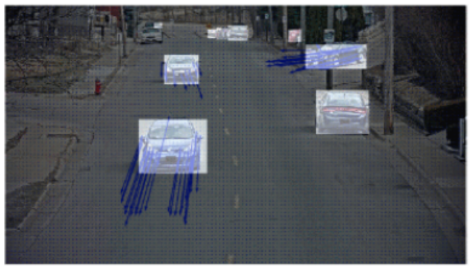


Figure 3: Optical flow for each detection.

For each frame with its detections (bounding boxes):

1. Obtain the best features to track inside the detections, using Shi-Tomasi corner detector.
2. Compute the optical flow of the features with the next frame using multi-scale Lucas Kanade algorithm.
3. Compute the average of the norm of the optical flow vectors inside the detection.
4. If average $<$ threshold, ignore the detection.

4.2. Multi-object multi-camera tracking

For each camera, the same methods implemented for Single-Camera tracking are used, Section 4.1.

In this subsections, a re-identification system to assign the same id of cars that appear in more than one camera using a siamese network is explained.

4.2.1 Siamese network

A Siamese neural network, as defined by Jane Bromley *et al*. [4], consists of two identical sub-networks joined at their out-puts. During training the two sub-networks extract features from two signatures, while the joining neuron measures the distance be-tween the two feature vectors. Verification consists of comparing an extracted feature vector ith a stored feature vector for the signer. Signatures closer to this stored representation than a chosen thresh-old are accepted, all other signatures are rejected as forgeries.

The network is composed of a basic ResNet18 network, by Kaiming He *et al*. [7], which was modified to add a fully connected layer at the end that returns the desired number of dimensions. In Figure 4 the evolution of the train loss can be observed .

The siamese was also tested with several networks, for example ResNet152 slightly improved the results, but at the cost of a huge increase in the training time.

4.2.2 Mining

As discussed by Alexander Hermans and Lucas Beyer [8], Siamese networks still are the best system to perform Re-Identification (ReID), and defines how to improve the training by performing online mining.

The process consists of dividing the dataset in P classes and K images per class, both 10 in our case. We then compute the embedding of each image by using the neural network. With this $(P \times K)xD$, being D the number of dimensions of the embedding, we compute the online loss. This is done by pairing each image with the one of the same class that is the furthest away, the most different equal image, and with the closest one of a different class, the most similar different image, as seen in Figure 5. This greatly improves the training by always using the hardest possible cases.

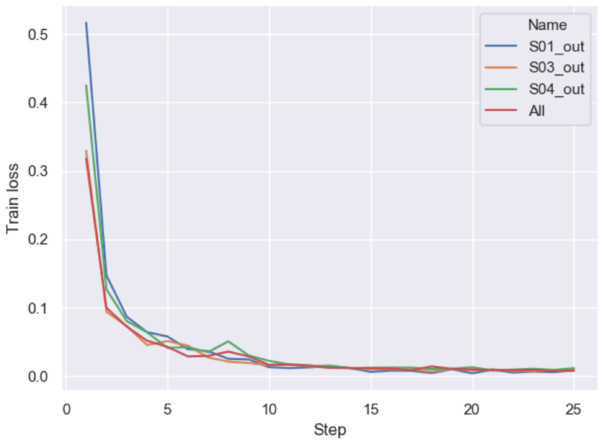
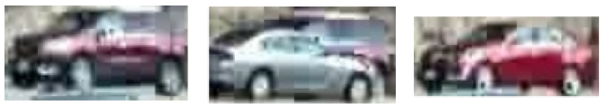


Figure 4: Evolution of the training loss during the first 25 epochs



(a) Anchor (b) Positive (c) Negative

Figure 5: Data mining to train the triplet network.

4.2.3 Re-Identification system

The re-identification system (ReID) is based on the previously explained siamese network. The idea is that, when we perform a the detection of a new car, we must decide whether to assign it a new id or reuse an already existing one. Whenever a new detection is made (something not previously tracked) we query the siamese database, we do the following:

1. Crop the image using the detection(s).
2. Send the image(s) to the siamese network to obtain the embedding(s).
3. Search the most similar image in the database with Unsupervised Nearest Neighbours.
4. Compute its distance.
   - If $distance < 1$ Return the class of the closest neighbor
   - Otherwise Return -1
5. Update the ID of the detection

If the detection has already been tracked, what is done instead is:

1. Obtain cropped images from each successfully tracked detection with their class.

2. Send the image(s) to the siamese network to obtain the embedding(s).
3. When the whole video is processed, add them to the database.

5. Results

The results were obtained with the detections provided by YOLOv3 [16]. Also the minimum intersection threshold was tested and the optimal vale was 0.5, which then has been used to compare the overlap of bounding boxes for all the methods that use it.

The optimal threshold to remove parked cars is 1.25, which has been determined though tests as shown in Figure 6.

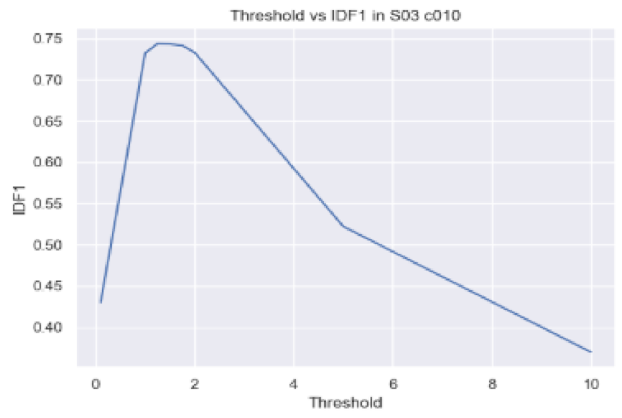


Figure 6: Parked cars removal threshold test results.

5.1. Metrics

In order to compute the metrics, the library py-motmetrics[1] has been used. The metrics IDF_1 is calculate using the Equation (1).

$$IDF_1 = \frac{2IDTP}{2IDFTP + 2IDFP + IDFN} \qquad (1)$$

where after global min-cost matching ID measures are,

- **IDTP:** true positive matches.
- **IDFP:** false positive matches.
- **IDFN:** false negatives matches.

5.2. Single camera

The results can be seen in Table 1, Table 2 and Table 3. In most of the scenarios the robustness of Kalman tracking make it the best method to track cars. Anyways there are sequences in which the movement of the objects are not linear (accelerating, turning or braking), so other methods like Overlap Tracking performs slightly better.

Nevertheless, some cameras in sequence 1 obtain better results using Kalman without removing static cars, the reason why there is no improvement removing static cars is that the sequence is a road junction, as seen in Figure 1, so static cars are not parked and they are considered in the ground truth.

5.3. Multi camera

The multi camera tracking presents a difficult task since a car detected at different cameras will appear with a different perspective which hinders the identification. Therefore, multi camera approaches achieve worst results than single camera tracking, as can be seen in Table 4, Table 5 and Table 6. The recall results show that the majority of cars are detected in all sequences and the precision to identify correctly the vehicles is around 60% in all the tested methods removing parked cars.

6. Conclusions

The implementation of one method that evaluates the detections of the objects in order to analyse if one car is stationed or not, was key to increase the tracking results, specially for single camera.

Kalman outperforms other methods for single camera tracking. Generally is the best in multi camera, but the re-identification system is underperforming, since the library used has only been slightly modified and more changes are needed to adapt it to our id system. Anyway Re-Identification is much harder than tracking.

In multiple camera tracking, when a road junction (as in sequence 1) is evaluated:

- The removal of static cars worsened the results, as cars without preference tend to stop, but they should be tracked.
- Kalman tracking obtains worse results as cars have non-linear movement.

More 40 high computation executions were done in the cluster provided by UAB, but more than 200 would be needed for all the necessary tests to be ran. Because of lack of time:

- Although, as state of the art shows, there is not much change using different detections, methods should be tested with detections from SSD512 [11] and mask RCNN [6].
- The triplet network performance should have been tested using different values for the number of the dimensions of the embedding.

[1]`https://github.com/cheind/py-motmetrics`

References

[1] Alexander Barth and Uwe Franke. Estimating the driving state of oncoming vehicles from a moving platform using stereo vision. *IEEE Transactions on Intelligent Transportation Systems*, 10(4):560–571, 2009.

[2] Massimo Bertozzi, Luca Castangia, Stefano Cattani, Antonio Prioletti, and Pietro Versari. 360 detection and tracking algorithm of both pedestrian and vehicle using fisheye images. In *2015 IEEE intelligent vehicles symposium (iv)*, pages 132–137. IEEE, 2015.

[3] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468, 2016.

[4] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744, 1994.

[5] Radu Danescu, Florin Oniga, and Sergiu Nedevschi. Modeling and tracking the driving environment with a particle-based occupancy grid. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1331–1342, 2011.

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

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

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

[9] Christoph Hermes, Julian Einhaus, Markus Hahn, Christian Wöhler, and Franz Kummert. Vehicle tracking and motion prediction in complex urban scenarios. In *2010 IEEE Intelligent Vehicles Symposium*, pages 26–33. IEEE, 2010.

[10] Miklas S Kristoffersen, Jacob V Dueholm, Ravi K Satzoda, Mohan M Trivedi, Andreas Mogelmose, and Thomas B Moeslund. Towards semantic understanding of surrounding vehicular maneuvers: A panoramic vision-based framework for real-world highway studies. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 41–48, 2016.

[11] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[12] Bruce D Lucas and Takeo Kanade. Optical navigation by the method of differences. In *IJCAI*, pages 981–984. Citeseer, 1985.

[13] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. In *Proceedings DARPA Image Understanding Workshop, April 1981*, pages 121–130. Vancouver, British Columbia, 1981.

[14] Hossein Tehrani Niknejad, Akihiro Takeuchi, Seiichi Mita, and David McAllester. On-road multivehicle tracking using deformable object model and particle filter with improved likelihood estimation. *IEEE Transactions on Intelligent Transportation Systems*, 13(2):748–758, 2012.

[15] Sebastian Nowozin. Optimal decisions from probabilistic models: the intersection-over-union case. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 548–555, 2014.

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

[17] Sayanan Sivaraman and Mohan Manubhai Trivedi. A general active-learning framework for on-road vehicle recognition and tracking. *IEEE Transactions on Intelligent Transportation Systems*, 11(2):267–276, 2010.

[18] 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. *arXiv preprint arXiv:1903.09254*, 2019.

[19] Bin Tian, Ye Li, Bo Li, and Ding Wen. Rear-view vehicle detection and tracking by combining multiple parts for complex urban surveillance. *IEEE Transactions on Intelligent Transportation Systems*, 15(2):597–606, 2014.

Appendices

Appendix A. Result tables

Camera	Overlap	Kalman	Optical Flow	Kalman with static
c01	0.5586660548	0.6212290503	0.53014385	0.3810287241
c02	0.581871345	0.614768174	0.5587828492	0.6094982879
c03	0.3846153846	0.4865089249	0.3974832215	0.5249632493
c04	0.472061657	0.5325179577	0.4555566493	0.5950241457
c05	0.5113393239	0.5572193682	0.5315934066	0.3400020107
AVG	0.5017107531	0.562448695	0.4947119953	0.4901032835

Table 1: MTSC results for sequence 1.

Camera	Overlap	Kalman	Optical Flow	Kalman with static
c10	0.6913365259	0.7437588318	0.7307869305	0.2656641604
c11	0.6045548654	0.6088449979	0.449978894	0.06965212675
c12	0.4256880734	0.5069860279	0.4312267658	0.03029699023
c13	0.6021505376	0.7102803738	0.5594322886	0.6670252824
c14	0.5676613463	0.574204947	0.5801635265	0.5681674065
c15	0.0295202952	0.03698811096	0.02222222222	0.004749266657
AVG	0.4868186073	0.5301772149	0.4623017713	0.2675925388

Table 2: MTSC results for sequence 3.

Camera	Overlap	Kalman	Optical Flow	Kalman with static
c16	0.7414248021	0.793220339	0.7330195024	0.7421777222
c17	0.6095717884	0.6507304117	0.6375176305	0.30472103
c18	0.4389534884	0.6783625731	0.3032159265	0.3827683616
c19	0.6100628931	0.6205787781	0.596214511	0.5868358446
c20	0.3644605621	0.397338403	0.2883211679	0.1543334293
c21	0.7313915858	0.7596236099	0.7319838057	0.3513931889
c22	0.8266452648	0.7679603633	0.8212560386	0.6047058824
c23	0.4010462075	0.4424460432	0.2694775435	0.158208244
c24	0.2857142857	0.5	0.2888888889	0.5569620253
c25	0.4057971014	0.543956044	0.4319852941	0.167903525
c26	0.6971428571	0.7406749556	0.6971851009	0.4433363554
c27	0.5642512077	0.6366366366	0.5990291262	0.2576797976
c28	0.6253229974	0.5929919137	0.6778711485	0.2427860697
c29	0.5461358314	0.563229572	0.5485981308	0.3083645443
c30	0.6118855466	0.5982314494	0.6449636155	0.2681051587
c31	0.2116182573	0.4688172043	0.2288329519	0.0156323571
c32	0.6120122753	0.582462517	0.4843524314	0.2303215926
c33	0.7104666969	0.7441101003	0.7165677773	0.623903177
c34	0.4932088285	0.5584358524	0.6002946748	0.4386137225
c35	0.8211731044	0.8504053058	0.7106283941	0.7077106993
c36	0.7228799227	0.7337951509	0.7065665132	0.5509946321
c37	0.5111583422	0.5638418079	0.521270867	0.7422680412
c38	0.5251450677	0.5205761317	0.4479840717	0.4275092937
c39	0.6155555556	0.5985576923	0.6158940397	0.4543034605
c40	0.6062322946	0.5609031491	0.6286366229	0.3427017226
AVG	0.5715702706	0.6187154402	0.557222231	0.4025695951

Table 3: MTSC results for sequence 4.

Metric	Overlap	Kalman	Optical Flow	Kalman with static
IDF1	0.2568477397	0.2726518455	0.2842597808	0.3026350967
IDP	0.2431512272	0.2506748034	0.2847903464	0.2051920622
IDR	0.2956865011	0.308492201	0.3158578856	0.6267090314
Precision	0.6323279367	0.6468570122	0.6460811343	0.3182549533
Recall	0.8235605623	0.8169170037	0.7897169266	0.9998555748

Table 4: MTMC results for sequence 1.

Metric	Overlap	Kalman	Optical Flow	Kalman with static
IDF1	0.3195140281	0.3041217007	0.3081481481	0.1097346172
IDP	0.2625295873	0.2555446302	0.2599717422	0.06450610893
IDR	0.4131843213	0.3788467768	0.3874311629	0.6054421769
Precision	0.5578925873	0.5867723719	0.5738423698	0.0993801853
Recall	0.8850016197	0.8751214772	0.8691286038	0.9972465177

Table 5: MTMC results for sequence 3.

Metric	**Overlap**	**Kalman**	**Optical Flow**	**Kalman with static**
IDF1	0.3571619336	0.3261435394	0.350087826	0.2176959568
IDP	0.3359852609	0.2999333944	0.3438011411	0.174038301
IDR	0.4637614149	0.4164258467	0.4492544215	0.7227488152
Precision	0.5832790445	0.5980432276	0.5889905549	0.1769672996
Recall	0.9314530112	0.9291411398	0.9226678997	0.998092706

Table 6: MTMC results for sequence 4.