M6 - Video Surveillance for Road Traffic Monitoring[*]

Ignacio Galve Ceamanos, Brian Guang Jun Du, Eric Henriksson Martí, José Manuel López Camuñas
Universitat Autònoma de Barcelona
{ignacio.galve, brian.guang, eric.henriksson, josemanuel.lopezcam}@autonoma.cat

Abstract

This report presents an overview of various methods to deal with the problem of monitoring road traffic based on video footage. Multi-target single camera tracking is first addressed. Techniques like Gaussian modelling and modern object detection CNNs are proposed as means to detect vehicles, while maximum overlapping and the Kalman filter are considered as a way to assign the obtained detections to tracks. The challenge of performing said tracking across different cameras is additionally studied, where several learning and non-learning based methods to match vehicles are considered and tested, including the use of color histograms and metric learning.

1. Introduction

The growth in the worldwide population over recent years has among other things led to unprecedented levels of traffic density, as the number of vehicles around the globe has been increasing exponentially during the past decade [1]. Road traffic monitoring is thus not only a necessity in current times, but is to become increasingly important and challenging in the next years.

The application of image processing and computer vision techniques to the analysis of road video footage has become the most convenient alternative in traffic monitoring. These tackle the drawbacks of other methods like inductive loop, sonar and microwave detectors, which have a hard time detecting slow or stationary vehicles and are expensive to set up and maintain [2]. They nevertheless still need to present short processing times, low processing costs and high reliability. While their basic functionality has been accomplished, they still experience certain difficulties, especially in terms of handling uncertainty and fusing information from different cameras.

This report specifically deals with traffic monitoring conducted through video footage from static cameras. This kind of footage is subject to irregularities like progressive and sudden changes in light levels or vibrations and altered camera positions. It is therefore important for systems to maintain their functionality regardless of them.

Multi-target single camera (MTSC) tracking is here first addressed. Once this functionality is implemented, multi-target multi-camera (MTMC) tracking is tackled.

[*]Supported by Universitat Politècnica de Catalunya.

2. Related work

Video-based traffic monitoring methods can involve the spatial, temporal, or spatio-temporal analysis of video sequences [2]. This section covers some of the most common techniques employed in this field.

2.1. Vehicle detection

The first step to address when designing a traffic monitoring system depending on camera footage is to determine a way to identify vehicles in the video frames, preferably in a manner that provides the approximate regions of the images that correspond to said vehicles.

2.1.1 Background/foreground differentiation

Assuming vehicles have different color characteristics than the background allows using thresholding to differentiate between the two. A common way of implementing this is through Gaussian modelling, which involves establishing a Gaussian distribution model of the background for each pixel based on information from a certain interval of frames [3]. Adaptive thresholding can also be employed in order to make the system more robust against irregularities like changes in light, which introduces the updating of the mean and variance of the background distribution models as the footage progresses. These approaches are nevertheless still prone to mistakes like the detection of shadows or missing parts of vehicles that resemble the background.

There are several libraries implementing methods of this kind. *OpenCV* has several Gaussian mixture-based algorithms available, MOG and MOG2. *BGSLibrary* [4] also introduces several background subtraction algorithms, including adaptive background learning, frame difference, static frame difference and weighted moving mean/variance.

2.1.2 Object detection

Modern convolutional neural networks (CNNs) designed for object detection can be used as an alternative to the mentioned methods. Some notable examples include Faster R-CNN and RetinaNet.

Faster R-CNN [5]: This architecture incorporates R-CNN [6] with a region Proposal Network (RPN) with the goal of sharing computation with the object detection network. The implementation uses a small network sliding over the feature maps of the last layer of a CNN, generating 9 different anchors at each position. The RPN itself has a classifier and a regressor, where the classifier determines the probability of an anchor containing the target object while the regression predicts the coordinates of the anchor.

RetinaNet [7]: In contrast to two-stage models like Faster R-CNN, RetinaNet is a one-stage object detection model that utilizes a focal loss function to address class imbalance during training. Focal loss applies a modulating term to the cross entropy loss in order to focus learning on hard negative examples.

2.2. Vehicle tracking

Several alternatives exist to establish relationships between detections across frames in order to determine tracks. Two of the most common ones are the following:

Maximum Overlapping: Bounding box detections in consecutive frames are matched through the evaluation of the intersection over union (IoU) between pairs, linking detections where this value is maximized and above a certain threshold. It is a simple algorithm that performs well in scenarios with no cluttered areas. Its main drawback is that as soon as a detection is missing in one frame the track is lost and its supposed continuation is assigned a new ID.

Kalman Filter: This approach is based on the Kalman algorithm [8], which is in general terms used to predict the position of each bounding box in the next frame. This technique is known to help achieve a more robust tracking, as it can mitigate the negative effects of aspects like occlusions or missing detections.

The use of optical flow has also been proposed in some works as a mean of improving object tracking through an estimation of motion [9, 10].

2.3. Tracking across cameras

MTMC tracking involves the re-identification of cars that appear in more than one camera. Most approaches to solve this problem pass through the use of some kind of feature representation obtained via object classification or metric learning [11]. To avoid having to construct delicate feature spaces, some work proposes text recognition (e.g., reading license plates) as an alternative [12]. The latest trends in the field of computer vision have also been heavily based on attention mechanisms [13, 14].

3. CVPR 2022 AI City dataset

This project is constructed around the CVPR 2022 AI City Challenge, and therefore based on its provided dataset. This dataset consists of multiple sequences, each composed of several videos of road footage taken from different cameras. Testing has been carried out on sequences 1, 3 and 4 of said dataset, with the characteristics detailed in Table 1. In the ground truth of the dataset, only cars in movement that appear in at least two cameras are labeled.

4. Methodology

The steps followed to solve the presented problem are here explained. The section is divided in two parts: MTSC tracking and its extension to perform MTMC tracking.

4.1. Multi-target single camera tracking (MTSC)

MTSC tracking is handled following the general process shown in Figure 1 and explained in the next subsections.

4.1.1 Vehicle detection

Three different approaches have been tested for detection. The first one has been using simple adaptive Gaussian modelling that first models the background of every pixel based on the first 25% of frames and continuously updates their distribution on the go. This method has been combined with a succession of opening and closing morphological operations to get more consistent foreground detections.

The other two approaches involve the use of object detection networks. A RetinaNet pre-trained on COCO [15] model with a ResNet-101 backbone has for every sequence first been trained on the other two sequences and then used to infer detections. In addition, a Faster R-CNN also pre-trained on COCO has been used to infer with no training.

Details of the specific adjustments for each of these approaches are available in the project's repository [16].

Table 1. AI City Challenge dataset.

Seq.	Time [min.]	# Cams	# IDs	Scene type
1	17.13	5	95	highway
3	23.33	6	18	residential
4	17.97	25	71	residential

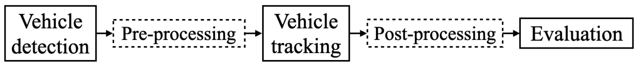

Figure 1. General steps of the process for MTSC tracking.

4.1.2 Pre-processing

Before establishing tracks, unwanted detections have been dealt with by removing cases in the zero regions of the ROI masks provided in the dataset. Non-maxima suppression has also been employed to avoid cluttered detections in delicate areas, where whenever two box detections have an IoU higher than 0.8 only highest confidence one is kept.

4.1.3 Vehicle tracking

Both maximum overlap and Kalman filter methods have been used to set tracks. The SORT variant [17] of the Kalman filter has been implemented, and a permissive IoU threshold of 0.3 has been used for both approaches.

4.1.4 Post-processing

Several measures have been taken to correct tracks by removing parts of these that do not fulfill conditions set by the ground truth. First, static parts of tracks have been discarded, as well as tracks shorter than 5 frames. Next, detections smaller than 0.7 of the smallest detection boxes in the ground truth (in width or height) have been disregarded.

4.2. Multi-target multi-camera tracking (MTMC)

Our implementation of MTMC extends the MTSC pipeline with a matching algorithm for comparing tracked cars across different cameras and a final relabeling stage before evaluation. The core task is to establish identity correspondences across different cameras.

4.2.1 Matching algorithms

Two main methods have been considered for matching vehicles: metric learning and color histograms.

Metric Learning: A custom dataset is generated for training with the purpose of distinguishing different vehicles and match ones that are the same. It is created from cropping the video frames using ground truth detections with each unique track ID as a class label.

A triplet network [18] with ResNet18 [19] as backbone has been used. Grid search has then been employed to obtain the optimal model with the lowest triplet loss. The last used training configuration is as follows: batch size of 32, learning rate of 1e-3 with a decay of 0.1 every 3 steps, 20 epochs, loss margin of 0.5, SGD as the optimizer and triple margin loss as the loss function.

The trained model outputs the embedding when given a pair of input images. A similarity score is calculated by computing the Hellinger distance [20] between embeddings. To establish whether images are a match, a threshold is calculated through computing the average of the distances between true positives retrieved at 1.

Color histogram: Color histograms are used as feature embeddings. For each detection, a 3D histogram is computed and matched with other histograms using Hellinger distance. To test the reliability of color histograms different color spaces have been tested: RGB, HSV and Lab.

4.2.2 Tracked cars representation

The measured similarity differs depending on the viewpoint. Selecting good frame or frames for matching is thus crucial. The solution presented here is to construct a viewpoint rich image representation for each tracked car across all cameras and sequences. Given the detections and tracked IDs from MTSC, the first frame of a tracked car is saved as an image representation that corresponds to it. Each following frame is then compared with frames within the image representation using the trained triplet network, and if the frames are dissimilar then the new frame is added to the image representation. As the algorithm iterates through the frames the image representation grows by including images that have different viewpoints and resolutions.

4.2.3 Cross camera matching

First, a new global ID list is established based on the information from the first camera of the sequence. Then, the image representations between the first two cameras are compared by computing their similarity with one of the matching algorithms. Tracks with the lowest distance are considered as a match only if the distance is also below the threshold set earlier. When a match is found the global ID list is updated with added viewpoints from the new camera. Otherwise, a new global track entity is added. At the same time the track IDs from MTSC are updated to the global IDs. This process is repeated until all cameras in sequential order have been checked. Finally, tracks that only appear in one camera are removed.

4.3. Evaluation metrics

The identification precision (IDP), identification recall (IDR) and IDF1 metrics from py-motmetrics [21] are used to determine the correctness of tracks both in MTSC and MTMC tracking. IDP and IDR provide information regarding tracking trade-offs, while IDF1 presents a balanced value between the other two through their harmonic mean.

5. Results

The main MTSC and MTMC tracking results for the conducted tests are summarized in this section. Only results for sequence 3 are provided in this report, as the insights that can be taken from these are quite representative for those of other sequences. Additional results can be found in the slides uploaded in project's Github repository [16].

Table 2. MTSC IDF1 tracking results for sequence 3 using different detection and tracking methods.

		Camera						Average
Detection method	**Tracking method**	**c010**	**c011**	**c012**	**c013**	**c014**	**c015**	
Adaptive Gaussian Modelling	Max. overlapping	0.333	0.494	0.095	0.279	0.444	0.444	0.348
	Kalman filter	0.348	0.492	0.088	0.282	0.522	0.522	0.376
RetinaNet trained on AI City Dataset	Max. overlapping	0.342	0.232	0.250	0.242	0.302	0.005	0.228
	Kalman filter	0.360	0.238	0.255	0.366	0.345	0.006	0.262
Faster R-CNN pre-trained on COCO	Max. overlapping	0.754	0.337	0.667	0.869	0.545	0.127	0.550
	Kalman filter	**0.768**	**0.472**	**0.824**	**0.767**	**0.742**	**0.129**	**0.617**

Table 3. MTMC tracking results for sequence 3 using the best MTSC tracking configuration and different matching methods.

Method		**IDF1**	**IDP**	**IDR**	**Precision**	**Recall**
Baseline		0.271	0.217	0.357	0.770	0.767
Metric Learning		**0.385**	**0.392**	**0.385**	**0.770**	**0.767**
Color Histogram	RGB	0.365	0.370	0.345	0.770	0.767
	HSV	0.371	0.372	0.371	0.770	0.767
	Lab	0.370	0.372	0.368	0.770	0.767

5.1. MTSC results

Table 2 displays the tracking results for various detection and tracking methods. The pre-trained Faster R-CNN appears to yield the best results by far, especially when using Kalman filter for tracking. Particularly low scores like those of camera c015 have to do with how most of the determined tracks belong to vehicles that only appear in that camera and are thus not considered by the ground truth.

The poor performance of the adaptive Gaussian modelling can probably be explained by its lack of robustness in situations of sudden change (e.g., changes in lighting) and its general sensitivity to noise, while the low scores of the RetinaNet are most likely related to non-optimal training conditions and inconsistencies in the ground truth.

5.2. MTMC results

Results for the MTMC tracking when using the best MTSC tracking configuration are shown in Table 3. The baseline results correspond to computing the metrics of MTMC with the results of MTSC without re-ID or matching. All matching experiments achieve better results, meaning that despite being far from perfect, the matching is working to some extent.

The best results are obtained with the metric learning generated features, probably due to how this approach is making sure different cars are placed far apart in the feature space. Still, the training process seems to need a better parameter adjustment and the addition of improvements like online hard negative mining.

The slightly lower performance of color histograms is probably attributed to some cars having similar colors and to disturbances like reflections, camera characteristics, etc.

6. Conclusions

This report has studied a variety of methods for detecting vehicles in video footage. It has been proven how pre-trained object detection neural networks can generate reliable detections if the addressed data is somewhat similar to that of the datasets used to pre-train the network. Also, the need for pre-processing to eliminate cluttered detections and cases outside of the ROIs has been emphasized.

It has been observed how Kalman filter helps establish more robust tracks compared to maximum overlapping. The importance of post-processing has also been stressed, especially for non learning-based methods, as it is essential to adapt tracks to the peculiarities of the ground truth at hand.

A new solution for selecting frames for matching has been proposed, presenting a richer image representation than regular approaches such as averaging. Metric learning has demonstrated being the best matching algorithm for this particular task. Setting an appropriate threshold has been the biggest challenge regarding the matching algorithms, and it is the deciding factor in limiting the results.

Moving forward, a more robust strategy should be considered for determining the matching algorithm thresholds. In an attempt to make the road monitoring algorithm more practical, it should also be tested under ”online” conditions (only past frames are available), as well as in real-time.

References

[1] N. K. Jain, R. Saini, and P. Mittal, "A review on traffic monitoring system techniques," in *Soft Computing: Theories and Applications*, pp. 569–577, Springer, 2019. 1

[2] V. Kastrinaki, M. Zervakis, and K. Kalaitzakis, "A survey of video processing techniques for traffic applications," *Image and vision computing*, vol. 21, no. 4, pp. 359–381, 2003. 1

[3] D. R. Magee, "Tracking multiple vehicles using foreground, background and motion models," *Image and vision Computing*, vol. 22, no. 2, pp. 143–155, 2004. 1

[4] A. Sobral and A. Vacavant, "A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos," *Computer Vision and Image Understanding*, vol. 122, pp. 4–21, 2014. 1

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015. 2

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014. 2

[7] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017. 2

[8] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960. 2

[9] A. Agarwal, S. Gupta, and D. K. Singh, "Review of optical flow technique for moving object detection," in *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*, pp. 409–413, IEEE, 2016. 2

[10] H.-H. Nguyen, P.-T. Tran-Huu, H. M. Tran, and S. V.-U. Ha, "Improved optical flow estimation in traffic monitoring system," in *2013 Third World Congress on Information and Communication Technologies (WICT 2013)*, pp. 165–169, IEEE, 2013. 2

[11] Y. Zhang, D. Liu, and Z.-J. Zha, "Improving triplet-wise training of convolutional neural network for vehicle re-identification," in *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1386–1391, IEEE, 2017. 2

[12] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 11, pp. 2298–2304, 2016. 2

[13] H. Guo, K. Zhu, M. Tang, and J. Wang, "Two-level attention network with multi-grain ranking loss for vehicle re-identification," *IEEE Transactions on Image Processing*, vol. 28, no. 9, pp. 4328–4338, 2019. 2

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. 2

[15] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014. 2

[16] I. Ceamanos, B. Guang Jun Du, E. Henriksson Martí, and J. Lopez Camuñas, "mcv-m6-2022-team5." `https://github.com/mcv-m6-video/mcv-m6-2022-team5`, 2022. 2, 3

[17] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3464–3468, 2016. 3

[18] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International workshop on similarity-based pattern recognition*, pp. 84–92, Springer, 2015. 3

[19] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-first AAAI conference on artificial intelligence*, 2017. 3

[20] R. Beran, "Minimum hellinger distance estimates for parametric models," *The annals of Statistics*, pp. 445–463, 1977. 3

[21] C. Heindl, "py-motmetrics." `https://github.com/cheind/py-motmetrics`, 2022. 3