Video surveillance for road traffic monitoring

Marc Carne
Polytechnic
University of Catalonia,
Spain

Sergio Sancho
Autonomous
University of Barcelona,
Computer Vision Center,
Spain

Alejandro Nespereira
Pompeu Fabra
University,
Spain

Axel Barroso
Open
University of Catalonia,
Spain

Keywords—car, real–time, tracking, Kalman, Python, speed.

***Abstract*—We propose a method for real time tracking and motion estimation for traffic environments. Combining stabilization techniques,background modeling, morphology filters and a very simple Kalman Filter we are able to reliably detect moving cars on a highway and obtain an estimation of their speed. We present our code in Python.**

I. Introduction

While the number of cars in our roads is expected to evolve with the population, the methodology of traffic control has not evolved. While it is true that the number and quality of sensors increase every day, they still yield a margin of speed rather than a fixed speed. Besides, their nature only allows them to give instant speed measurements that can (and are) easily fooled by drivers. On top of that, the systems usually trigger when a speeding car approaches, closing the door to more elaborate systems.

In order to fix these disadvantages, we present a tracking system that is able to extract information on real time from all the passing vehicles. This information could be used as a cornerstone of complex systems such as intelligent traffic, accident detection, license plate recognition, or calculating time between locations.

This article starts with the study of the related work at section II, the pipeline description on section III, followed by a more deep explanation of our algorithms at sections *Foreground subtraction*, *Morphology operations*, *Tracking and speed estimation* and *Stabilization*. We present results at section IV.

II. Related work

Car tracking in road environments has become increasingly popular in the last years, slowly replacing current RADAR systems for speed estimation and making an appearance in autonomous cars. Surveillance systems are widely popular as well, and the current trend of incorporating computer vision algorithms to improve its performance and/or replacing the human factor has favoured the rise of novel approaches.

Foreground substraction has provided very good results when used for video surveillance, as the authors of [6] and [3] have proved. These algorithms provide a reliable segmentation between background and foreground that can be used as a starting point for other algorithms. These methods are often improved with hand crafted approaches, such as morphology filtering or shadow removal [5] The authors of [2] present a method for motion tracking of cars using LBP features and an extended Kalman filter. Others, such as [4] make use of Bayesian tracking with good results.

III. Car tracking and Speed Estimator

As mentioned, the proposed method is able to count moving cars in a highway and estimate their speed. To do so, we have explored the well known approach of background modelling to retrieve the moving objects. Using morphological operations we are able to substantially improve both the numerical and visual performances of our system. We conducted a study of stabilization to improve the image quality reducing the effect of wind and pulse that were adding jitter to the recordings. Finally, by using a own Kalman filter implementation we are able to estimate the position and speed over the road for each car. More deep explanation is in the following subsections.

A. Stabilization

As a first stage of our pipeline comes the stabilization. Stabilization plays a key role in tracking. By estimating and correcting the camera movement, we can make the measured motion closer to the real. As we want to build a robust vehicle tracking system able to deal with real conditions, i.e. high frequency jitter. We decided to apply block matching for study the optical flow, and compensate the whole frame with the global motion
We propose three approaches.

1) ***Block matching in all the image having fixed reference frame****:* . For doing so, take first frame as a reference and compensate all frames respects to the first one. Good point is simply to code, but not too robust.

2) ***Block matching in a certain region of the frame****:* . Similar than previous technique. Take a region of the first frame as a reference and study the optical flow of this single region over all frames. We then compensate all frames respects to the first region. It is fast to compute, but we have to assume a really strong assumption that can not be true always.

3) ***Block matching in the whole image having variable reference frame****:* . This last technique is the one implemented in our project. We change the reference frame in every pair of frames, when one frame is compensated, it becomes the reference. And the next frame will be compensated respect to this one. It gives very good results, but it has a really high computational cost.

B. Background estimation and foreground subtraction

Once the frames are stabilized, we count with the minimum variation in each pixel, as ideally background pixels will only be affected by smooth lightning changes. The apparition of a foreground object will produce a drastic change in the pixels intensity. Using this simple concept we can segment the foreground from our background with a very simple model trained over a very low number of frames. We have used a Gaussian model for each of the pixels in our frames, and an adaptative version.

1) Non adaptative Gaussian model: Starting with a training sequence of our scene, we can study the intensity values in each pixel in order to infer some knowledge about the background. As the foreground objects are supposed to be moving, their total contribution on each pixel is minimized. Thus, we can create a Gaussian curve for each pixel, characterised by its median μ and variance σ, representing our background. After we have trained the model, we can compare new frames from our video source to see if they fit in our background model. To do so, we check if the pixels in the image fit in the range $[\mu - \sigma - \alpha, \mu + \sigma + \alpha]$, where α is a custom parameter that needs to be fine tuned. The goal of this parameter is to add some range to what values we consider as background.

Changing the color space of our images plays a key role in this model, as the results vary greatly when using a single gray scale image or multiple color channels. Selecting a suitable color space can lead to substantial improvements on the results, specially when using color spaces where the colors are not interlaced, as HSV or HSL. This makes color discrimination more robust and precise, which in turns boosts the performance.

Fig. 1. The result of our foreground estimation algortihm. Altough it is very noisy, it provides a very good starting point for the rest of the algorithms.

2) Adaptative Gaussian model: While the previous approach yields very good results, it presents some issues that need to be addressed. These issues arise from the simplicity of the model and the nature of the problem. Obtaining a model of our scene background is a reliable and robust approach when objects in the scene are expected to keep moving. However, for some scenes as streets, rooms, or landscapes, we can expect some foreground items, i.e. cars, people or clouds to remain quiet in a position for unknown lapses of time. Obviously, our model is too naive to learn that foreground objects can become background and therefore it will fail to classify them.

To solve this, we have used an adaptative approach where we constantly update our model with the data that our whole system provides. At each frame, we update the pixel Gaussians that have been identified as background with the new information. The weight of the new value against the model depends of a new parameter ρ that we have fine tuned. This approach, however, is very sensitive, and every wrong update from a foreground pixel classified as background contributes to new errors in the future. This is why we have decided to update our models with the final result of our system. After applying morphology filters we have found the foreground detection much more reliable, which in turn results in better updates. This is specially true for the foreground pixels that are classified as background, that can be easily corrected by simple hole filling.

C. Morphology operations

To improve the foreground mask we applied the following filters:

- Hole filling: some zones detected may have holes of non-detected pixels. Filling those holes removes false negatives but can increase false positives. We studied the effects of 4 and 8 connectivity.
- Area filtering: since the holes are already filled, it is time to process the frames to remove noise. As the vehicles are quite large, we can eliminate any connected component smaller than a set area.
- Closing and opening: closing is the erosion of the dilation, it is useful to fill holes. Opening is the dilation of the erosion, effective to remove small objects that are not part of any foreground object. We applied both at the end of the pipeline to improve the results.

D. Tracking and speed estimation

Tracking and speed estimation are a fundamental part of this project. We present different ideas of how to track and how to estimate cars and velocity. Those ideas are presented here below.

1) Tracking: For the tracking stage, we implement different methods. Kalman filters and coherence between centroids are the final winners. We must say that in the last stage, Kalman filter performance was better.

Kalman Filters[1] are a family of algorithms that helps us estimate predict new samples from the statistical distribution of previous ones. We use this technique to predict car positions over frames. The position that is predicted by our Kalman filter can be corrected after the prediction is done. After a short period of time we come with a powerful predictor, that becomes more robust and accurately over time.
We create an instance of a Kalman filter for each detection. Thus, in each frame we compare all detected cars with series of Kalman filter estimations, if the lower distance between an estimation and a real car is lower than a threshold, we decide that we are dealing with the same car. If we have a match, we update the Kalman filter value. Otherwise, we create a new one.
To ease memory issues, we remove Kalman filter instances if there are not been detected over the last 10 frames. We only

deal with objects that are supposed to be in the scene in the current time.

Coherence between centroids uses the information a priory of the environment to map the more possible car positions between frames. Once we detect cars in two different frames we search for the relationship more likely. If distance between more likely cars is lower than a threshold, we mark these two cars as one. The big drawback of this technique is that we need to study the environment we are dealing with.

2) Speed estimation: One of the project challenge was to compute the speed of cars.

After review the state of-the-art for this topic, many works deal with the problem using two parallel lines and computing the time for crossing both lines for a car. This approach is simple, easy to implement and have good results. Despite that requires have a good tracking and not allow a real time application because the car is analyzed in two concrete points.

The approach we will present deals with the problem with a technique that allow a real time speed-computation using optical flow estimation for each object. One particularity of our approach is that it not need a tracking with a large precision as the speed estimation is computed between two frames closed in time (consecutive or with a little stride, for example, each 5 frames).

Pipeline for speed estimation in our approach consists in look at all the cars (objects) detected in a certain frame (for example, with a foreground subtracting algorithm) as a first step. Once we have located each car, we look at the previous frame (or applying a stride, for example 5 frames before) and we look for the most similar block in a searching region. For our approach we used square error to compute that difference. Once we matched the block we computed the motion vector. This motion vector can has a small error due to the global image motion that can be compensated by a stabilization algorithm. Although is a good option we decided not to apply it as its time consuming. To deal with the global motion effect in the speed estimation we increased the stride between the frames we used to compute the speed. As more time passes between both frames, the error in the motion is lower as we assumed the car has a motion higher than the jitter effect for example, that provoques global image motion.

Having the motion vector for the object to estimate the speed and we know the pixel-meter relation in the real point for a pixel location in the image we can multiply the number of pixels in the image with the relation, but we have to take into account the image is a 2D projection of the reality, a 3D scene, so there is a projection effect. By this reason, objects that are far in the real scene are represented by a lower number of pixels. For that reason if a car have a uniform movement (constant speed) the optical flow estimated seems to be larger and larger when the car are coming to the camera position. To deal with that problem we proposed a couple of equations, one for each dimension to compute real distances for pixels taking into account the projection effect. This equations consists in two differentiated parts: by one way there is a term that transforms an amount of pixels to meters, from a known relation, but we still having the same perspective problem. By other way we have a term that introduced an offset of pixels, a correction applied to the first term distance estimation. The second term add or remove meters depending on the distance of the object to the reference point (the point ore region we had used to define the pixel to meter relation.

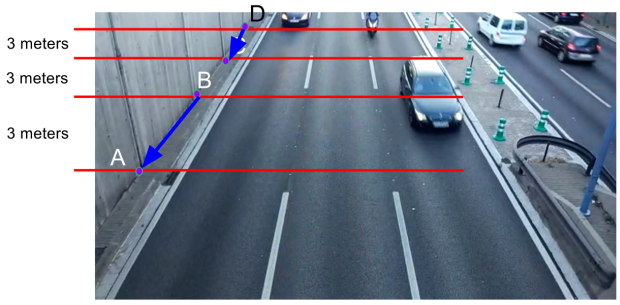

Fig. 2. An insight in the projection used to determine speed. As the distance from the camera to the road increases in our scene,the amount of pixels that represent the same distance is smaller. Therefore, we need to take this into account to estimate the speed. The distance in pixels between the red lines represent the same distance in the real world.

The approach we presented allows a real time implementation as requires two consecutive (of with an stride) frames, so as minimum 1/30 seconds if we consider a frame rate of 30Hz. Although, we have to say that as less time pass between frames the error in the optical flow estimation increases. A good point of our approach is that if the tacking become lost at some point, the speed still being computable as the algorithm look at detected objects and take a window from previous image to search the object.

Another interesting think is that the real time application is because the block search is done for all the bounding box of the object, so instead of grid the bounding box and compute an optical flow by voting the sub-block movement, we each for all the bounding box in the previous image. On problem can appear is that due also to the 2D projection, object size are not kept between frames, but considering a small stride between frames, the size change is not too much important.

IV. Results

As a groundtruth is needed to evaluate our approach results, we are not able to check if the stimated speed are precise or not. Although this speed is just a linear operation over optical flow where the car has been detected, so what we will evaluate is the correct detection of the car as is the main part in the speed estimation, more important than the tracking as in the previous section was explained. For that evaluation we used the groundtruth of the object position in the Traffic and Highway datasets and we will compare different strategies for perform the detection, giving the AUC (area under the curve) for the precision and recall curve. In the tables, FS means the result for the optimal foreground substraction without apply any other technique to the result, HF is the hole filling method applied, MF are the results for the morphological filters application and with SR, the best shadow removal approach explored was applied. To end, for the real enviroment application of this project we will give some visual results of the algorithm, and

other visual examples as video sequence can be seen in the Github repository.

A. Traffic dataset

Next table shows results for the evaluation of the object detection in the Traffic dataset considering the approaches explained before.

Method	AUC PR	Gain
FS	0.5560	-
FS + HF	0.5613	+0.0053
FS + HF + MF	0.7463	+0.1903
FS + HF + MF + SR	0.674	+0.118

TABLE I. EVALUATION RESULTS FOR TRAFFIC DATASET

B. Highway dataset

Next table shows results for the evaluation of the object detection in the Highway dataset considering the approaches explained before.

Method	AUC PR	Gain
FS	0.658	-
FS + HF	0.6696	+0.0116
FS + HF + MF	0.733	+0.075
FS + HF + MF + SR	0.853	+0.195

TABLE II. EVALUATION RESULTS FOR HIGHWAY DATASET

From the results we can observe hole filling and apply morphological filters are useful in both cases, while shadow removal decrease the AUC of the PR curve for the Traffic dataset, maybe due to their image properties or as some dark cars appears, all these cars can be considered as shadows, loosing a lot of objects, so the recall is dropped.

C. Real environment examples

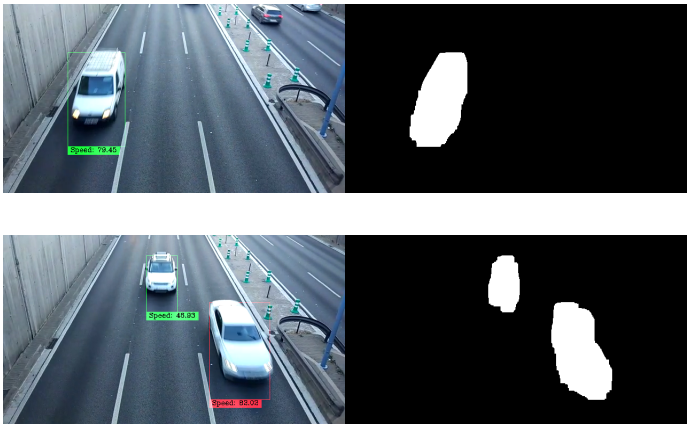

Fig. 3. Our system is able to estimate the speed of the vehicles and evaluate if they are infringing the law.

V. CONCLUSION

This work introduced a real–time approach for video surveillance based on Python using a single camera. The results show how the approach works on different scenarios. The results also shows that using the method a good tracking and speed estimation can be performed in real environments. Furthermore, if the high computational cost of having stabilization is critical, we show that using it without stabilization we are able to get a good performance.
We provided all the code for all the variants at: https://github.com/mcv-m4-video/mcv-m4-2017-team3

VI. FUTURE WORK

Many different adaptations, tests, and experiments have been left for the future due to lack of time. In what follows we list them all:

- Develop the application in a police car instead of having a static camera.
- Fine-tune the car tracking in order to weight better the penalizations in x-axis and y-axis (instead of penalizing almost in the same way).
- Fine-tune the speed estimator to be more robust to time differences and motion values (instead of computing the speed between five frames).
- Perform the speed estimator giving as an input a color image instead a grey-scaled image.
- Implementation of a car speed estimator using CNN (SpeedNet) giving as an input as least two frames and having as an output the predicted speed. As temporal information is computed a good option can be the use of a RNN.
- Perform the calibration using a marker on a motorcycle driver (helmet).

ACKNOWLEDGMENT

This work was supported by video analysis professors of the master in computer vision imparted in the UPC.

REFERENCES

[1] Rudolph Emil Kalman et al. “A new approach to linear filtering and prediction problems”. In: *Journal of basic Engineering* 82.1 (1960), pp. 35–45.

[2] Kristian Kovacic, Edouard Ivanjko, and Niko Jelusic. “Measurement of Road Traffic Parameters Based on Multi-Vehicle Tracking”. In: *CoRR* abs/1510.04860 (2015). URL: http://arxiv.org/abs/1510.04860.

[3] Chris Stauffer and W. Eric L. Grimson. “Learning patterns of activity using real-time tracking”. In: *IEEE Transactions on pattern analysis and machine intelligence* 22.8 (2000), pp. 747–757.

[4] YingLi Tian et al. “Robust detection of abandoned and removed objects in complex surveillance videos”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 41.5 (2011), pp. 565–576.

[5] Hanzi Wang and D. Suter. “A re-evaluation of mixture of Gaussian background modeling [video signal processing applications]”. In: *Proceedings. (ICASSP ’05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.* Vol. 2. Mar. 2005, ii/1017–ii/1020 Vol. 2. DOI: 10.1109/ICASSP.2005.1415580.

[6] Jie Zhou, Dashan Gao, and David Zhang. “Moving vehicle detection for automatic traffic monitoring”. In: *IEEE transactions on vehicular technology* 56.1 (2007), pp. 51–59.