# Video Surveillance for Road Traffic Monitoring

Guillermo Torres[1] ⋆, Ivan Caminal[2], and Cristina Maldonado[2] Marc Górriz[2]

[1] Universidad Nacional del Comahue, 8300 Neuquén, Argentina
[2] Universidad Politècnica de Catalunya, 08034 Barcelona, España
guille.torres,ivancaminal72,cmmsmcgma,algayon2}@gmail.com

**Abstract.** This work proposes a framework for road traffic surveillance using computer vision techniques. After a foreground estimation, post processing techniques are applied to the detected vehicles in motion to generate blobs. Then, a tracking approach based on Kalman filters is used to extract instantaneous information throughout a video sequence, including speed and trajectory estimation and imprudent driving detection. The system has been developed in Python and can be launched in real-time using a standard CPU. The code is available at github: `https://github.com/mcv-m6-video/mcv-m6-2018-team3`.

**Keywords:** traffic control · road monitoring · foreground segmentation · vehicle tracking · kalman filter · speed estimator.

## 1 Motivation

Full roads and congested traffic relates a common problem nowadays. Consequently, there is a need to develop intelligent traffic surveillance systems that play an important role in road monitoring. Its purpose is to provide numerical data in real time about traffic activity and to point out potentially anomaly situations, such as accident detection or dangerous driving. Video analytics is defined as surveillance based computer vision algorithms and systems to extract appropriate information from video. This can assist the detection of events of interest, which is useful for traffic management due to additional data available. Video cameras have been used for a long time for traffic monitoring purposes, because they provide a rich information for human understanding. Automatic preprocessing allows efficient control for the operators to process and collect statistics from live videos, with the aim to improve traffic flow.

The article starts describing the related work in section 2, followed by a detailed explanation of the methodology in section 3 and finally, the overall framework evaluation at section 4.

## 2 Related work

Detecting vehicles in images acquired from roads is a challenging problem and has become increasingly popular in the last years, replacing current approaches

⋆ This research is part of the project Altas Prestaciones Parte III (04/F012) of the Universidad Nacional del Comahue, Argentina.

such as RADAR or mobile surveillance configurations. The solutions have to deal several adverse situations and uncontrolled environment with a high variability (background, illumination), and in a reduced processing time.

Regarding the background subtraction, several approaches provides very good results, including frame differencing [5], gaussian mixtures [3], median filter [7] or non-parametric kernel-density estimation [16]. For the case of vehicle tracking, deep learning has overcome the state of the art of the methodology, both for prediction quality and computational efficiency. Some powerful approaches for detection include Yolo [10] and Faster R-CNN [11], while specifically for tracking, Zhang [14] proposed multi-layer occlusion detection and processing framework for the mutual occlusion between two vehicles and Faro [13] proposes a further improvement by introducing curvature scale space to segment occluded region accurately.

## 3 Methodology

In this section we will present the different datasets, methods, configurations and applications that enabled us to obtain the final pipeline, see figure 1.

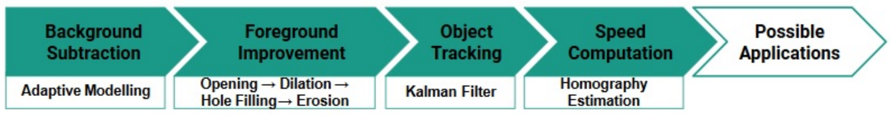


Fig. 1: Pipeline followed to make our application.

### 3.1 Datasets

During this project we worked with one downloaded video [3], named ”Custom” in this paper, and with three other datasets obtained from *ChangeDetection* [15] corresponding to the year 2014: Traffic, Fall and Highway. The Custom has no annotations while the ones from *ChangeDetection* have. Hence, we used the annotated ones to asses quantitatively the quality of the obtained masks in subsections 3.2 and 3.3. Regarding the results obtained in subsections 3.4 and 3.5 all datasets could be used because we only asses them qualitatively, however, only Traffic, Highway and Custom were finally used.

A split was done to the four sequences to obtain train and test sets, see Table 1.

[3] `https://www.youtube.com/watch?v=nt3D26lrkho`

## 3.2 Background subtraction

Background subtraction is the first step in order to detect moving objects in a sequence. For this task was tried three approaches: *Non adaptive*, *Adaptive* and *MOG*. After considering the results, that are presented in the section 4, the *Adaptive* one was selected for the final pipeline.

**Non adaptive** This approach, models the background of the sequence using a single Gaussian for each pixel in the train set. When the model is estimated, we threshold every pixel in the test set by (1) to obtain binary masks.

$$|I_i - \mu_i| \geq \alpha \cdot \, (\sigma_i + 2) \tag{1}$$

Where $i$ is the pixel position, $\mu$ is the mean, $\sigma$ is the standard deviation and $\alpha$ is a constant. The result is a binary mask where high pixels belong to foreground and low ones to background. The optimum value of $\alpha$ was found by maximizing f1-score metric.
Illumination changes and errors in the scene are the main limitations of this approach. Therefore, is only useful in high-supervised applications.

**Adaptive** This other approach, is equal to the previous but the parameters of the model evolve over time following (2) and (3) after each prediction.

$$\mu_i = \rho \cdot I_i + (1 - \rho) \cdot \mu_i \tag{2}$$

$${\sigma_i}^2 = \rho \cdot (I_i - \mu_i)^2 + (1 - \rho) \cdot {\sigma_i}^2 \tag{3}$$

Where $\rho$ is a constant within [0,1] that works as a trade-off between the current and accumulated value of mean and variance. Grid search was used maximizing again f1-score metric to find the optimal combination of $\alpha$ and $\rho$ parameters. Adaptive methods, are chosen for more applications due to the limitations of non-adaptive ones.

| | **Train** | **Test** |
|---|---|---|
| Highway | 1050-1200 | 1201-1350 |
| Fall | 1460-1510 | 1511-1560 |
| Traffic | 950-1000 | 1001-1050 |
| Custom | 1-359 | 1-359 |

Table 1: Split (in frames) of the used sequences to obtain test and train sets.

**MOG** MOG [4] is an improved adaptive Mixture Model (implemented in *OpenCV*). It uses a method to model each background pixel by a mixture of K Gaussian distributions (K = 3 to 5). The weights of the mixture represent the time proportions that those colors stay in the scene. The probable background colors are the ones which stay longer and more static.

### 3.3 Foreground improvement

In this part, we will try to improve the previous masks with some video preprocessing and mask post-processing.

As pre-processing, we used two stabilization methods only in the Traffic set (which had camera jitter). The first was a own implementation using block matching algorithm and the second was a web application [12].

For mask post-processing we tried shadow removal using an available *OpenCV* function called MOG2 [4] that detects shadows. Furthermore, some morphological operators were used in different combinations and structuring elements.

After applying different combinations of the mentioned processing methods, we discarded the ones that did not improve the results. The final improvement pipeline can be seen in Table 2.

| | Highway | Traffic | Custom |
|---|---|---|---|
| 1-Opening | s.e. 4x4 | s.e. 11x11 | s.e. 3x3 |
| 2-Dilation | x4 iter. | x8 iter. | x3 iter. |
| 3-Hole Filling | 4 connect. | 4 connect. | 4 connect. |
| 4-Erosion | x4 iter. | x8 iter. | x3 iter. |

Table 2: Best improvements applied to the masks obtained in section 3.2 ordered up to down. The opening was done with a square structuring element (s.e.). The iterations (iter.) in dilation and erosion were done with a square 3x3 structuring element.

### 3.4 Object tracking

In order to identify a vehicle along the sequence and extract real time features we used different tracking algorithms: Kalman filter [1], Kernelized Correlation filter (KCF) [2], Median Flow filter [8] and Boosting filter [9]. We implemented the first approach while we took the *OpenCV*[4] implementations [cite] for the others. Once the foreground masks are generated, tracking is performed following the pipeline described in figure 2. First, connected components are extracted from

[4] `http://opencv.org`

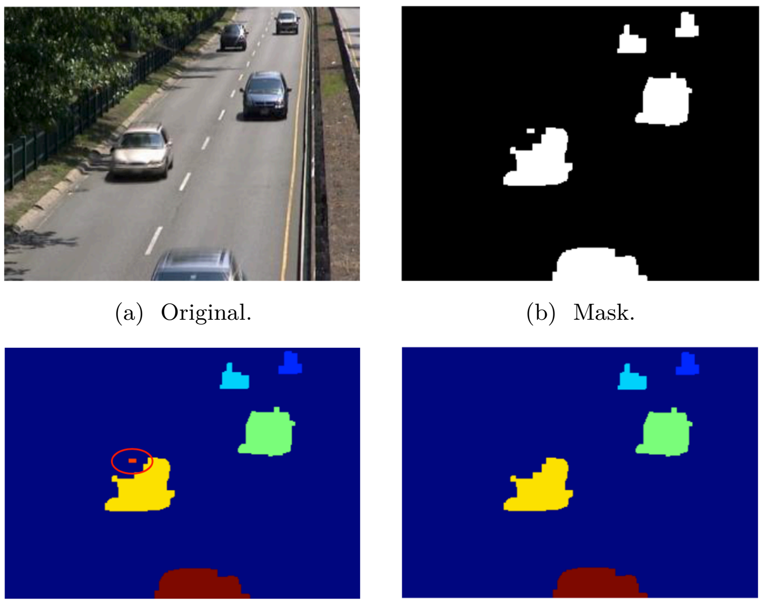

(a) Original. (b) Mask. (c) Connected comp. (d) Area filtering.

Fig. 2: Tracking preprocessing pipeline.

the masks. Then, area filtering is applied in order to remove the noisy objects. Finally, each foreground object is tracked throughout the sequence.

The filters are able to do tracking taking blobs as input and determining the next position of the objects using spatial measurements. After some test results (see section 4), Kalman filter seems to be the most accurate approach. Is robust against illumination changes in shadows and in difficult situations such as horizontal displacements when the vehicle changes the land.

### 3.5 Speed computation

For the speed estimation, in the first step we use a projective transformation to obtain a bird's-eye perspective and compute the new location of the car centroids. In this way, the 8-point algorithm [6] was implemented using only 4 points to obtain a fundamental matrix, which is computed only once. It is important to point out that the fundamental matrix is only applied to the position of the centroids, in contrast to be applied to the whole image, to save computation resources [5].

[5] Team 1 credits from last year.

In the second step, the speed computation takes advantage of the birds'-eye perspective and assume that the movement of the vehicles is only in y-direction, which is shown in the figure 3.

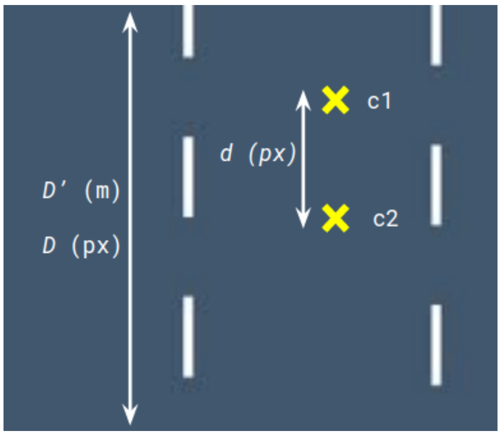


Fig. 3: Measures used to compute the speed estimation: $D$ and $D'$ represent the relationship between pixels and meters respectively, and $d$ represents the movement of a centroid from the position $c1$ to $c2$

Thus, the speed is computed in (4). Where *fps* stands for frames per seconds and 8 denotes the frequency in which speed is updated. This frequency was arbitrary selected to decrease the speed variation of the vehicles, specially, when a new vehicle begins to be tracked.

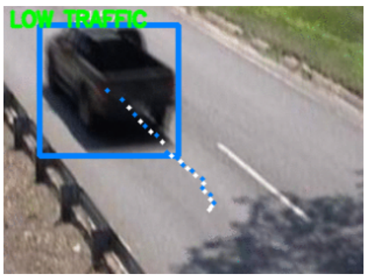
$$Speed = \frac{fps}{8} * \frac{D'}{D} * d \tag{4}$$

## 3.6 Application

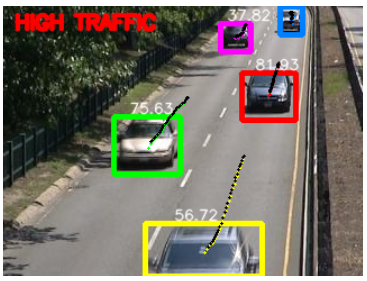
With all the techniques described before, we build our Road Traffic Monitoring system. This application is the end part of the pipeline showed in the figure 1 and was built with the best methods and configurations evaluated in the previous sections.
For this reason, the application was configured to use the Kalman filter. The sets employed were Traffic, Highway and Custom, which are described in section 3.1.

This application does the following:

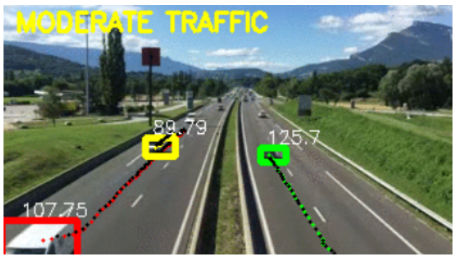
- density of the road: considering 1 or 2 vehicles as low traffic, 3 or 4 vehicles as moderate traffic, and 5 o more as high traffic;



(a) Traffic dataset.



(b) Highway dataset.



(c) Custom dataset.

Fig. 4: The Road Monitoring Traffic in action using the Kalman filter in the (a) Traffic, (b) Highway, and (c) Custom datasets. The tracked vehicles are closed with a color bounding box and its speed is shown over it. Also the vehicles's trajectory are drawn with color for the centroids and with black/white for the filter predictions. And a label with low/moderate/high traffic is shown for the density of the road.

– detects when vehicles exceed speed limit; and
– obtain a trajectory estimation of the vehicles.

A picture of the application in action is shown in the figure 4.

## 4 Results

In this section, we present the quantitative results for the two first parts of the pipeline. The table 3 shows the best results obtained for each background subtraction. The adaptive approach get highest maximum F1-Score for the three datasets evaluated, and in second place it is followed by non adaptive. The particular parameters values used for the non adaptive and adaptive are listed in table 4.

Figure 5 shows the foreground improvement (green) reached by the adaptive approach in background subtraction (black) for the traffic and highway test sets. In the figure 5a along the recall axis, always it is obtained a precision improvement. And the figure 5b shows that the precision get an improvement from 0.5 onwards in the recall axis. Below that value it works worst (red), which means that the approach has a negative impact in the foreground and in the forward pipeline steps it will worsen the vehicles detection.

Also, a qualitative results for the other parts can be seen in the figure 4.

| | *Non Adaptive* | *Adaptive* | *MOG* |
|---|---|---|---|
| Highway | 0.45 | **0.73** | 0.35 |
| Fall | 0.67 | **0.71** | 0.39 |
| Traffic | 0.48 | **0.68** | 0.57 |

Table 3: Maximum F1-Score background subtraction approaches from section 3.2

| | *Non Adaptive* | *Adaptive* | |
|---|---|---|---|
| | $\alpha$ | $\alpha$ | $\rho$ |
| Highway | 1.9 | 2.89 | 0.21 |
| Fall | 4.5 | 3.2 | 0.05 |
| Traffic | 1.8 | 3.55 | 0.16 |

Table 4: Optimal parameters, sections 3.2 and 3.2

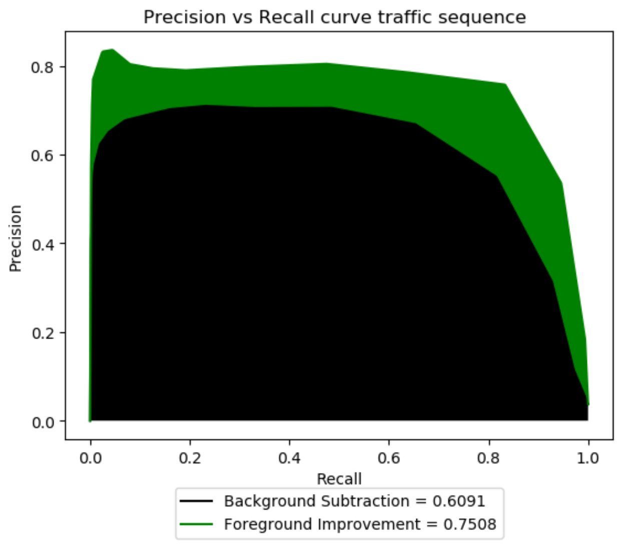
.

## 5 Conclusions

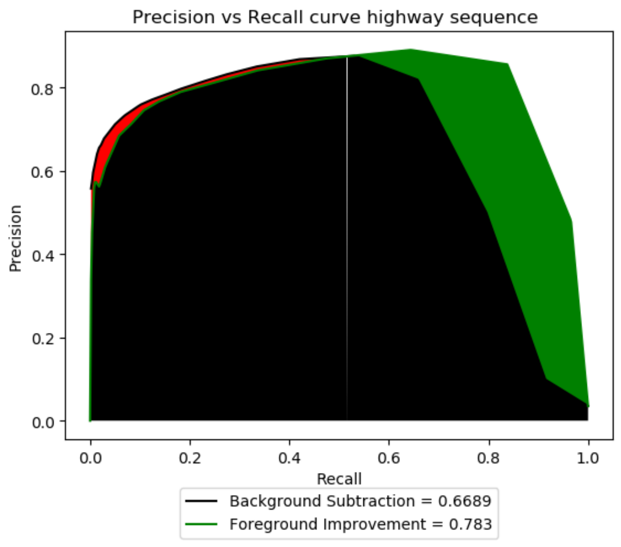
We developed a traffic monitoring system modularized by a pipeline to track cars, count them, estimate their trajectory and speed. We learned that errors in the first stages of the pipeline can produce bad results in subsequent stages. It is very important to have a robust foreground detection system in order to avoid errors in object tracking.

For object tracking we used different algorithms and we found that Kalman filter is the best method to track the cars in the video and gives the best qualitative results. With this technique there are less erroneous detections and the trajectories are estimated with more precision than other filters.

As future work, the application could incorporate a metric of imprudent driving taking into account the distance between vehicles, its trajectory and



(a) Traffic test set



(b) Highway test set

Fig. 5: Visual improvement of the precision and recall curves and area under the curve of two of the analyzed test sets in sections 3.2 and 3.3.

speed. Also is interesting to advance in direction of deep learning to track the vehicles using a neural network like Yolo [10] or other. A measurement of the required time for each step and in the over all process is needed to know the viability of the techniques, because it is a real time application.

## References

1. Bishop, G., Welch, G., et al.: An introduction to the kalman filter. Proc of SIGGRAPH, Course **8**(27599-23175), 41 (2001)
2. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. IEEE Transactions on Pattern Analysis and Machine Intelligence **37**(3), 583–596 (2015)
3. Jin, C., Zhang, Y., Balakrishnan, S., J. Wainwright, M., Jordan, M.: Local maxima in the likelihood of gaussian mixture models: Structural results and algorithmic consequences (09 2016)
4. Kaewtrakulpong, P., Bowden, R.: An improved adaptive background mixture model for realtime tracking with shadow detection (2001)
5. Kartika, I., Mohamed, S.S.: Frame differencing with post-processing techniques for moving object detection in outdoor environment. Signal Processing and its Applications (CSPA), 2011 IEEE 7th International Colloquium on (March 2011)
6. Longuet-Higgins, H.C.: A computer algorithm for reconstructing a scene from two projections. Nature **293**(5828), 133 (1981)
7. Mallikarjuna Rao, G., , C.: Object tracking system using approximate median filter, kalman filter and dynamic template matching **6** (04 2014)
8. Oflazer, K.: Design and implementation of a single-chip 1-d median filter. IEEE transactions on acoustics, speech, and signal processing **31**(5), 1164–1168 (1983)
9. Okuma, K., Taleghani, A., De Freitas, N., Little, J.J., Lowe, D.G.: A boosted particle filter: Multitarget detection and tracking. In: European conference on computer vision. pp. 28–39. Springer (2004)
10. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You Only Look Once: Unified, Real-Time Object Detection. ArXiv e-prints (Jun 2015)
11. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015)
12. Unknown: Video Stabilize, `https://video-stabilize.com/`
13. W. Zhang, Q.W.e.a.: Adaptive background modeling integrated with luminosity sensors and occlusion processing for reliable vehicle detection. IEEE Transactions on Intelligent Transportation Systems **12**(4), 1398–141 (Dec 2011)
14. W. Zhang, Q.W.e.a.: Multilevel framework to detect and handle vehicle occlusion. IEEE Transactions on Intelligent Transportation Systems **9**(1), 161–174 (Feb 2008)
15. Wang, Y., Jodoin, P.M., Porikli, F., Konrad, J., Benezeth, Y., Ishwar, P.: CDnet 2014: An Expanded Change Detection Benchmark Dataset `https://www.cv-foundation.org/openaccess/content{_}cvpr{_}workshops{_}2014/W12/papers/Wang{_}CDnet{_}2014{_}An{_}2014{_}CVPR{_}paper.pdf`
16. Zanin Zambom, A., Dias, R.: A Review of Kernel Density Estimation with Applications to Econometrics (Dec 2012)