ROAD TRAFFIC MONITORING

Sergi Canyameres Masip, Dèlia Fernández Canellas, Jordi Frias Navarro, Arcadi Llanza Carmona

Universitat Politècnica de Catalunya (UPC)

ABSTRACT

This article presents a method for traffic monitoring, based on background subtraction and tracking of vehicles. We also study possible improvements when using video stabilization and introduce a method to compute vehicles velocity. Our best results are able to accurately monitor traffic recorded with camera jitter, in real time.

Index Terms— Foreground Estimation, Object Tracking, Kalman Filter

1. INTRODUCTION

Traffic congestion, safety and vehicle speed control are just a some example applications of traffic monitoring. This field of study has been growing on recent years due to the exponential increase of the number of vehicles on the roads and the importance and effect of them in our daily life. In response to this, many researchers are developing intelligent traffic systems [6], which can increase safety in traffic intersections [1], incident detection [5] and prevent traffic jams.

In this article we present a traffic monitoring system capable of tracking cars and control vehicles velocity. The tracking system is based on a first foreground estimation, which is presented on section 2. We also tested a video stabilization method with the objective better estimating the foreground, which is presented on section 3.

On section 4, the two methods we used for the car tracking are presented: Kalman filter and Mean Shift. Results from both methods are compared lately on section 6. Finally, the speed control method is described on section 5.

For testing the system we used three sequences from CDNET [8] with different recording perspective of the road and background noise problems: $1 - Highway$ (baseline), $2 - Fall$ (dynamic background) and $3 - Traffic$ (camera jitter). This allows us to evaluate if our system is resistant to all these interferences. For foreground estimation evaluation all three sequences are used, but for tracking and speed control only 1 and 3 are used.

2. FOREGROUND ESTIMATION

The motion estimation of the elements in the scene is the baseline for any traffic monitoring pipeline. A traffic monitoring system is supposed to be focused in static roads (highways, urban avenues, etc.), so the aim of these systems is to monitor highly controlled scenarios with constricted variability. Also, these systems are designed to work in real-time. Due to both assumptions, a complex and computationally expensive algorithm for foreground estimation [7] is useless in front of a simpler and much faster approach. In this paper we propose a fast and real-time foreground estimation system using a Non-Recursive Gaussian Modelling [2].

This technique is based on the hypothesis that the observed video sequence I consists on a fixed background B in front of which moving objects are observed. With the assumption that a moving object at time t has a color (or a color distribution) different from the one observed in B, and considering that all the pixels in B follow a Gaussian distribution $N(\mu_{i,j}, \sigma_{i,j})$, the method estimates the foreground F as:

$$F_{i,j} = \begin{cases} 1 \text{ if } |I_{i,j} - B_{i,j}| \geq \alpha \cdot (\sigma_{i,j} + 2) \\ 0 \text{ otherwise} \end{cases} \tag{1}$$

Where α allows us to choose the acceptance area of the Gaussian distribution, and the addition of 2 avoids low values of σ to react to very small variances and so be more robust to smooth light changes. The α value is optimized in terms of the F1-Score, using parameter sweep. Since the operation is done pixel-wise, our method is embarrassingly parallel and thus very fast to compute.

In order to correct the well known errors of the pixel-wise operations, like noise in the foreground estimation, partial estimations, etc. we perform a fast post-process of the foreground estimation. Initially, a 4-Connectivity hole filling is applied, in order to fulfill small holes in objects. Afterwards, we apply an area filtering of objects smaller than a given pixel size P. This method erases noise and small artifacts, keeping only the bigger and most relevant moving objects. Finally, a combination of an opening and a bigger closing allows us to reconstruct the non-estimated regions of the car. All the morphological operators sizes are optimized in terms of AUC, using grid search. Figure 1 shows the main effects of this post-process improvements.

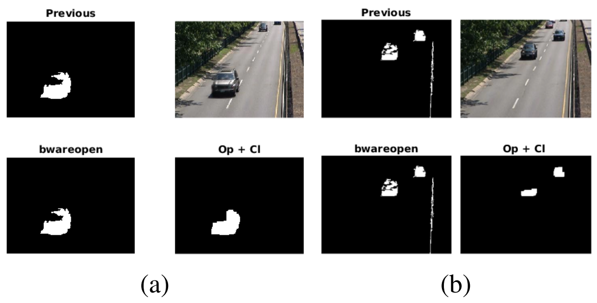


Fig. 1. Foreground estimation. Despite of the Non-Recursive Gaussian Modelling working pretty well, this post-processing allows us to achieve a better reconstruction of the cars (a) and a proper denoise of the estimations (b) without a remarkable impact in the real-time performance.

3. IMPROVE FOREGROUND ESTIMATION USING VIDEO STABILIZATION

As explained before, the goal of our system is to work in highly controlled scenarios. For example, a fixed camera recording from the top of a road. One of the assumptions made to perform the foreground estimation was a fixed background. So, light movements produced by windy conditions makes our background model not suitable. We propose a video stabilization technique in order to overcome this trouble.

Given two consecutive images in a sequence I^{t-1} and I^t, we perform an optical flow estimation of the whole image with a backward brute-force block matching algorithm. The differences between blocks are computed with a 3-channel MSE. This way, we can estimate for each frame t the x and y displacements D_x^t and D_y^t with respect to the $t-1$ frame.

The wind can move the camera making it to start and end in different positions, that is, $(X(0), Y(0)) \neq (X(T), Y(T))$. We propose to track the position $(X(t), Y(t))$ of the camera and try to follow it while avoiding the hard bounces:

$$X(t) = \int_0^t D_x^s ds, \quad Y(t) = \int_0^t D_y^s ds \tag{2}$$

We address this problem performing a smooth and locally linear regression of the accumulated averages of D_x^t and D_y^t and then correcting the difference between the actual and the smoothed positions (Fig. 2).

4. CAR TRACKING

At this point, we use two different approaches to track car trajectories. As detailed below, we obtain better results with a Kalman filter [4] than when using the Mean Shift method [3].

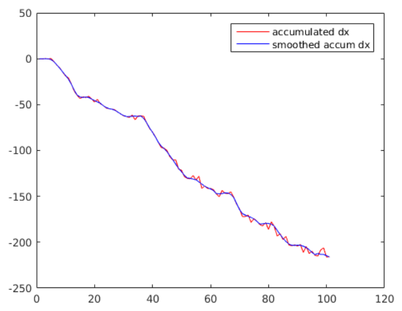


Fig. 2. Video stabilization. If the camera is moving towards an (or various) edges, correct the x and y displacement at each frame could be not useful. The goal of our method is to correct the gaps between the actual (red, estimated with block matching) and the estimated (blue, smoothed) position.

The main car trajectory estimator processes a video sequence by using the following pipeline:

```
while videoHasFrames()
    detectObjects();
    predictTrack();
    updateAssignedTracks();
    updateUnassignedTracks();
    deleteLostTracks();
    createNewTracks();
    updateSpeeds();
end
```

Given this guidelines, we adjusted the different parameters of the trackers in order to fit our sequences requirements and boost their performance.

4.1. Kalman filter

The Kalman filter is a recursive algorithm that uses observed estimations over time. This measurements are assumed to contain some Gaussian noise. Hence, the tracker can infer the most probable object positions over time. As the assumption is usually correct in natural sequences, the Kalman filter is the optimal method to achieve our goals. Moreover, it can run in real time, which makes it totally appropriated for traffic-monitoring applications.

4.2. Mean shift

The Mean shift algorithm is an iterative method that creates a confidence map in the new image based on the histogram of the object in the previous image. We use this algorithm to find the peak of a confidence map near the object's old position.

The confidence map is a probability density function of the image I_t, representing the likelihood of each pixel to appear in the object at I_{t-1}. A histogram-based implementation incorporates the continuously adaptive mean shift (CAMShift) algorithm for object tracking. It uses the histogram of pixel values to identify the tracked object.

In Figure 3 we show how Mean Shift tags car objects inside a bounding box. We add an identifier to count how many vehicles are in the road.

To be able to detect cars in the road, we use our own implementation of background subtraction algorithm. We compute a blob analysis system object to find connected groups of foreground pixels. In Figure 4 we can see an example of how the tracker works correctly.

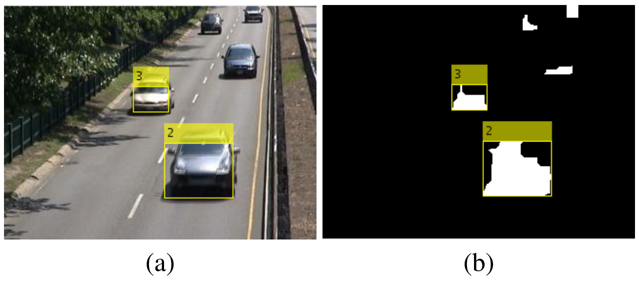


Fig. 3. We highlight the detected cars with a yellow bounding box and an identifier (a) Image of cars in a highway. (b) Foreground image of cars in a highway.

5. SPEED CONTROL

A common goal of many traffic monitoring systems is to increase the security in roads and urban regions. A simple way to achieve this goal is to perform an accurate estimation of the cars speeds.

Under the assumption of a local lineal and not biased transformation between meters in the real scene and pixels in the recorded images, we are able to estimate properly the speed of the cars on the scene:

$$v_{real} = v_{estimated}(\tfrac{pix}{frame}) \cdot f_r(\tfrac{frame}{sec}) \cdot \alpha(\tfrac{m}{pix})$$

Where f_r is the frame-rate and α is the local relationship between meters and pixels.

We consider that all the cars in the scene are being tracked properly throughout all their path. Under this consideration, knowing the f_r parameter and defining a region A in the image with its associated α, we can estimate the speeds of the cars in this region. The speed of a single car is computed only taking into account the length l_A (in pixels) of the car's path inside A and the number of frames t_f it has taken to travel this path:

$$v_{estimated} = \frac{l_A}{t_f}\frac{pix}{frame}$$

Figure 5 shows the process of the speed computation in a visual way.

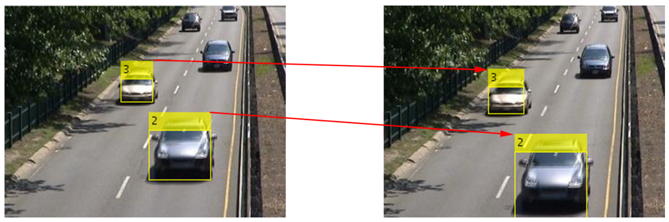


Fig. 4. Image of cars in a highway (frame 10, frame 20). We highlight the detected cars with a yellow bounding box and an identifier. We show with a red arrow how the tracker is working in the temporal domain.

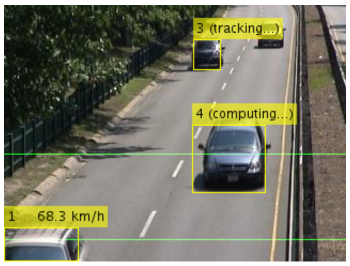


Fig. 5. Once an object is consistently tracked, an estimation of its velocity can be computed in a given region A (between green lines). This computation is possible thanks to the assumption of a local linear conversion between pixels and meters.

6. RESULTS

6.1. Foreground Estimation

The foreground estimation results are highly improved using morphological operators. As can be seen on Table 6.1, this operations allow to increase the AUC gain 0.26 points. For all sequences we get high AUC, which allows an accurate foreground detection. We decided not to use stabilization in further steps, because as seen in results, when tested in the sequence with camera jitter we are not able to improve estimation results with respect to the not stabilized sequence.

On Fig.6 the precision and recall curves for the three sequences and the averaged curved are shown. It can be noticed how best results are get from $Fall$ sequence, where we are able to erase the artifacts from tree leaves moving. Surprisingly good results are obtained from $Traffic$, despite of camera jitting. On the other hand $Highway$ sequence results worst than what could expected from the simplicity of the scene. On $Highway$ cars are seen from a top perspective, so the groundtruth includes car windows,which are harder to detect because of the illumination.

6.2. Car Tracking

We achieved good results adapting and tuning the Kalman filter and Mean shift algorithms. We were able to learn the main

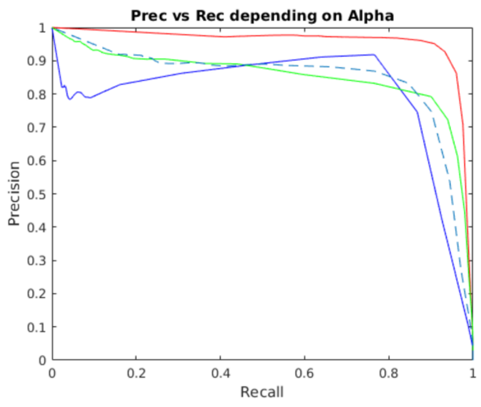


Fig. 6. Precision vs Recall curve depending on α from Foreground estimation after morphological operations.

Area Under the Curve (AUC)			
	Before Morph.	**After Morph.**	**Stab.**
1	0.554	0.821 (+0.266)	-
2	0.756	0.957 (+0.201)	-
3	0.527	0.8549 (+0.260)	0.672(+0.144)
Avg.	0.612	0.873 (+0.260)	-

Table 1. Area Under the Curve (AUC) results comparison before and after morphology and using stabilization from Foreground Estimation and the corresponding gains, from the 3 video sequences and the Average results.

difference of those methods and we could perform the system to improve our results.

6.3. Speed Control

We record two different sequences of videos to ensure that our system is working fine. With those news videos we could quantize the error.

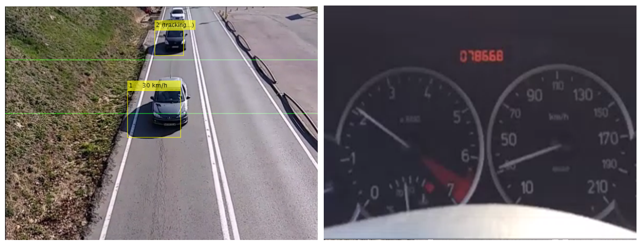
We realize that the system is working pretty good taking into account that we applied a really simple approach.

In Fig.7 we observe how the system properly estimates the speed of the vehicle in when driving at 30km/h. The biggest reported estimation error is only $\pm$ 2 km/h in the fastest movement sequence, at 50km/h.

7. CONCLUSIONS AND FUTURE WORK

We have presented the design, implementation, and evaluation of computer vision techniques for traffic monitoring.

As seen on section 6 our method proved accurate results for static environments. Nevertheless our system has many limitations that should be improved to work in real scenarios. Non-recursive modeling for foreground estimation does not



(a) (b)

Fig. 7. Example of good velocity estimation. On the left, (a) the correct prediction. On the right, (b) the corresponding groundtruth is shown.

adapt to light changes. Future work should improve the background model so it could work in real time with a technique like Gauss-Seidel, which would overcome these limitations. Also, car tracking should be improved, as our method is not capable to discriminate between consecutive cars if they are too close.

8. REFERENCES

[1] Marco Gruteser Baik Hoh and Hui Xiong. Enhancing security and privacy in traffic-monitoring systems. *Pervasive Computing*, 2006.

[2] Y. Benezeth, P. M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger. Review and evaluation of commonly-implemented background subtraction algorithms. In *In Proceedings of the IEEE ICPR*, 2008.

[3] Keinosuke; Larry D. Hostetler Fukunaga. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21:32–40, 1975.

[4] R. E. Kalman. A new approach to linear filtering and prediction problems. *ASME Basic Engineering*, 1960.

[5] Katsushi Ikeuchi Fellow IEEE Shunsuke Kamijo, Yasuyuki Matsushita and Masao Sakauchi. Traffic monitoring and accident detection at intersections. *IEE Transactions on Intellicent Transportation Systems*, 2000.

[6] Nivedita Patra Amartya Mukherjee SSatya Priya Biswas, Paromita Roy and Nilanjan Dey. Intelligent traffic monitoring system. *ICCC*, 2016.

[7] Chris Stauffer and W. Eric L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2000.

[8] Jodoin; Fatih Porikli; Janusz Konrad; Yannick Benezeth; Prakash Ishwar. Yi, Wang; Pierre-Marc. Cdnet 2014: An expanded change detection benchmark dataset. *CVPR*, 2014.