VIDEO SURVEILLANCE FOR ROAD TRAFFIC MONITORING

Arnau Baró, Pau Cebrián, Victor Campmany and Guillem Cucurull

Universitat Politècnica de Barcelona
Master in Computer Vision
Barcelona

ABSTRACT

With this work we introduce an affordable computer vision based traffic monitoring system. It aims to predict speed and track vehicles along the road. We use different computer vision techniques such as video stabilization, foreground segmentation and region tracking.The system is capable of running in real time using a regular CPU.

***Index Terms*—** Road Traffic Monitoring, Video Surveillance, Video Stabilization, Foreground Segmentation, Region Tracking

1. INTRODUCTION

Road accidents are one of the main mortality causes in our society. Lots of efforts and money have been put in order to mitigate this problem. However, still nowadays we did not find an ultimate solution. Velocity radars are one of the most common systems to warn drivers and avoid them to drive over the speed limits. Though, RADARs are expensive and they need the right installation.

In this work we present a computer vision based speed estimation system that just needs a camera sensor and the code that we provide. By using computer vision we introduce a affordable and cheap system with minimal need of infrastructure. For example, our system could be placed in a bridge over the road, in traffic signs or even in police cars.

We have used several techniques in order to obtain a reliable software: (1) video stabilization; (2) background modeling; (3) foreground segmentation and; (4) region tracking and speed estimation.

The rest of the paper is organized as follows: section 2 introduces the related work; section 3 explains the used algorithms; and we end with section 5 summarizing the work.

2. RELATED WORK

Road traffic monitoring has been an active research topic during the last two decades. Several vision-based works appeared obtaining good results [1, 2, 3]. However, we do not base our research on the previously cited works. We combine a bunch of well know computer vision algorithms and create a video surveillance system out of them.

Video stabilization is a key factor of road traffic monitoring, some works use feature tracking [4] and some others use optical flow based techniques, such as Lukas-Kanade or Horn-Schunck [5, 6].

Some of the approaches successfully complement the system with shadow removal techniques such as the ones introduced in [7, 8, 9].

Most of the works also include a vehicle tracking system by using techniques such as Kalman Filters [10]. Another approach is to use mean shift to perform tracking [11]. Recent techniques include a deep learning approach introduced by Wang [12].

3. SYSTEM AND METHODS

In this section we describe the algorithms and methods that we have used in order to develop the system. It is divided in 4 basic steps illustrated in figure 1 in form of a pipeline.

3.1. Video stabilization

Captured videos could be jittery since the cameras are usually placed in traffic signs, poles or cars. It is important to smooth the sequence in order to obtain better results.

We performed the video stabilization using optical flow. We compute the optical flow with the Block Matching algorithm, and, as output we obtain a couple of vectors fields corresponding to the movement in the x and y axis. Then, we determine the global motion of the image; to do so, we compute the mode of each of the vectors fields and, finally we translate the image according to the mode values.

3.2. Background modelling

In order to model the scene we use a single gaussian approach, where each pixel is modeled as a gaussian distribution, this means that each pixel will have an associated mean (μ_p) and variance (σ_p). To obtain the parameters a training sequence is used. We claim that the best scenario is using a still sequence of the road to model the background (no cars passing by).

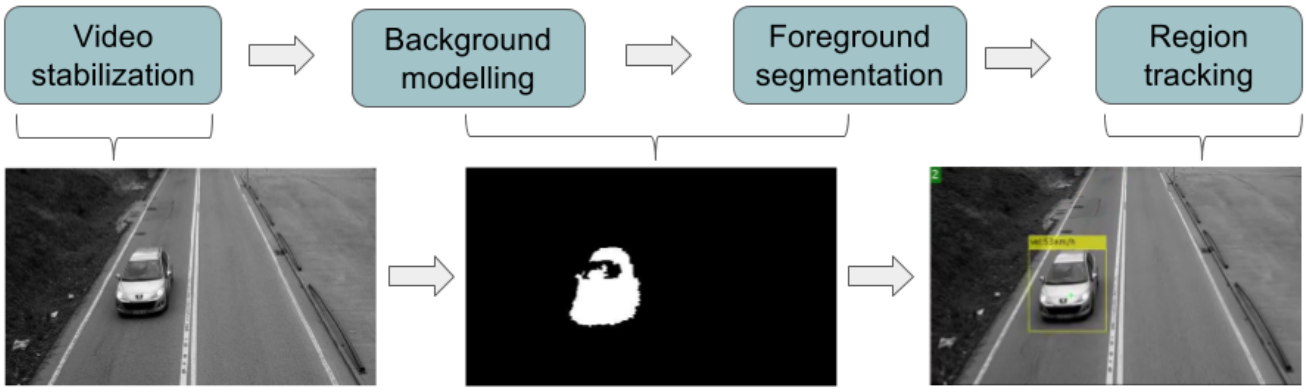


Fig. 1: Basic steps of the system in order to perform the tracking and speed estimation

Initial μ_p and σ_p for each pixel are estimated from the training set.

3.3. Foreground Segmentation

We have tried two different approaches to do the background segmentation. The non-adaptive where the background model is constant and the adaptive where the model is updated according to incoming pixels. The parameter ρ controls the strength of this update, the bigger the more aggressive is the modification. Equations 1 and 2 show how this update is performed.

$$\mu_p(t) = \begin{cases} (1-\rho)\mu_p(t-1) + \rho I_p(t) & p \in bg \\ \mu_p(t-1) & p \in fg \end{cases} \quad (1)$$

$$\sigma_p^2(t) = \begin{cases} (1-\rho)\sigma_p^2(t-1) + \rho(I_p(t) - \mu_p(t))^2 & p \in bg \\ \sigma_p^2(t-1) & p \in fg \end{cases} \quad (2)$$

In both methods the segmentation is performed the same way; each pixel(I_i) is labeled as foreground if the following condition is met:

$$|I_i - \mu_i| \geq \alpha(\sigma_i + 2) \text{ for every pixel i} \quad (3)$$

Where the parameter alpha (α) controls the tolerance. Best results where obtained using the adaptive model, and we found the best α and ρ parameters with a grid search.

Once the foreground is segmented, we improve our segmentation by trying different morphological techniques. We applied area filtering to remove noise, a closing to join areas of the same object that could be separated and hole filling to fill gaps.

3.4. Object Tracking

Once we have segmented the frames and performed motion segmentation to segment the moving objects on the video is when we apply the object tracking algorithm. Our method consists on the following pipeline:

1. Once a frame is segmented we perform a connected components analysis to detect blobs and get a bounding box around each detected object. In this step we discard objects with an area of less than 300 pixels, equivalent to a square of size 17×17 roughly. We also compute the filling ratio of each bounding box, which is defined by $filling\ ratio = \frac{\text{\# white pixels}}{\text{\# pixels}}$, and bounding boxes with $filling\ ratio < 0.3$ are discarded. This filtering allows to filter a lot of spurious detections.

2. Once we have detected the moving cars in a frame, we try to assign the detections to an initialized Kalman Filter. To do so, the distance between each detection and each Kalman Filter is computed. Then, if a detection does not match to any of the objects being tracked, we initialize a new Kalman Filter for that detection. Here another filtering step is done, and only the objects that are tracked for at least 8 frames are counted as real detections.

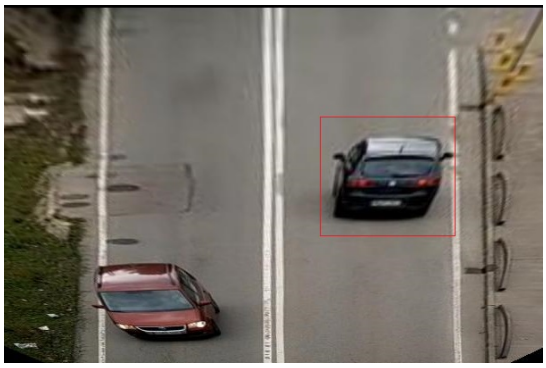
3. Repeat steps 1. and 2. for all the frames in the sequence.

With this method we are able to identify the trajectory of each car along the sequence, so we can count how many distinct vehicles cross the road and we can estimate their velocity. As it will be seen in the Results section this tracking methods works well and is capable of tracking multiple vehicles in real-time with a CPU.

3.5. Speed Estimation

As explained before, by tracking the position of the cars across the sequence we can try to estimate their velocity. The following is an schematic pipeline of the process we use to estimate the speed of a car:

(a) Image distorted by perspective. The distance in meters between two pixels is not equivalent in the whole image.



(b) Projected image where the distance in meters between pixels is the same in all the image.

Fig. 2: Projection for speed estimation

1. Get the current and previous position of a car being tracked.

2. Compute the distance between the current and previous position.

3. Translate this distance in pixels to an equivalent distance in meters.

4. By knowing the time elapsed since the last frame we can easily compute the velocity as $vel = \frac{distance}{time}$ where distance is the difference in meters between current and previous position and the time is the time in seconds since last frame.

The previous pipeline has one important problem in step 3. The problem is that the distance in pixels is distorted by perspective, so the same distance in pixels does not correspond to the same distance in meters in all the image, it depends on the region of the image where the distance is being measured. See Figure 2a for an example. To overcome this problem we propose to project the two positions in a plane where the distances are equivalent, so that the relation between pixels and meters is the same everywhere. To do so, we learn a transform matrix T that projects the points of a distorted image X to a distortion-free image Y by computing and homography, see Figure 2b for an example. Once we have this matrix, both the current and the previous position are projected to this new space, and the distance between them is computed. With this trick we are able to have a constant relation between pixels and meters in the whole image.

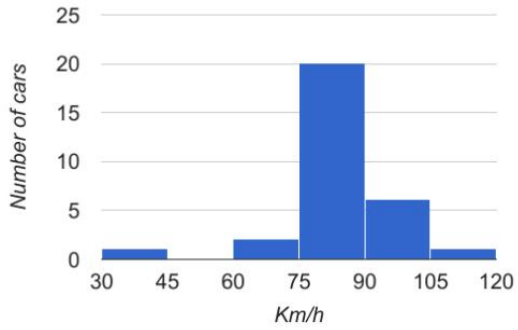


Fig. 3: Speed histogram distribution on highway sequence

In order to add more robustness to the speed estimation method, we average the observed estimation of the current frame with all the previous computed velocities of the same car.

4. RESULTS

Now we are going to analyze the results of our system on three specific sequences, Highway, Traffic from the ChangeDetection dataset [13] and our own sequence recorded at Universitat Autónoma de Barcelona. Our study is focused on the histogram of speed distributions of the detected vehicles.

4.1. ChangeDetection Sequences

We focus on the speed histogram distribution instead of the specific speed values, because they are related to our system calibration, but the histogram is independent from this issue.

In Figure 3 we found a reasonable distribution for the highway sequence, with only one outlier obtained from a bad estimation, but still, it is a reasonable speed so it can not be filtered by our system.

Figure 4 shows how traffic sequence performs pretty bad due two main reasons, the camera position, which difficulties our homography, and a poor segmentation due to jittering.

We have some speeds that are clearly bad estimated, with values around 300 Km/h. The presence of this outliers makes our system unsuitable for speed traffic detection on this case, but it can be still useful for traffic monitoring and analytics after filtering them.

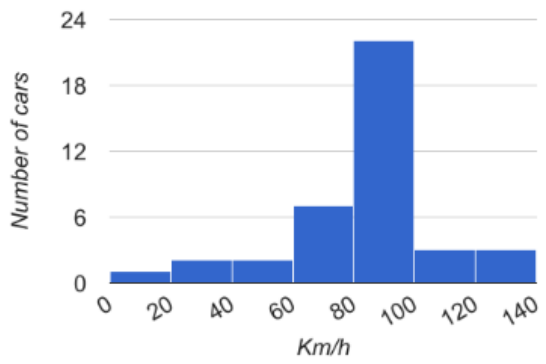


Fig. 4: Speed histogram distribution on traffic sequence after filtering outliers

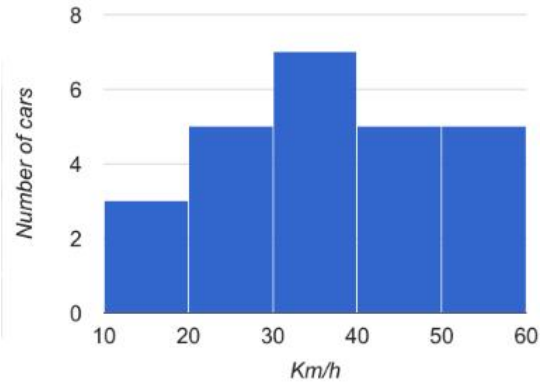


Fig. 5: Speed histogram distribution in our own sequence

4.2. Own Sequence

Finally, for our study, we have recorded a short sequence, using the black car in figure 2a for calibration purposes.

Using Google maps, we have used a reference distance to get the equivalence between number of pixels on our image and real distance meters. Combined with our camera fps specifications and the control car, we have everything that our system needs in order to be calibrated.

In the sequence, shown in Figure 5, we found two remarkable weak points, one pedestrian tracked as a vehicle, and unexpected behavior when tracking large vehicles such as buses or trucks.

This could be solved filtering elements by speed, shape or location, for the pedestrian case; and placing our camera far away from the road for the large vehicles case.

5. CONCLUSIONS

We described how we did build a cheap system for traffic monitoring and speed estimation.

Also, we have spotted the weak points of this approach, like the relevance of foreground segmentation, which can be a real challenge depending on the sequence.

In general, with right settlement of the cameras, like a straight and measurable road, stable position and good illumination, our system can be used in order to act as traffic monitor and speed estimator.

6. REFERENCES

[1] Antonio Fernández-Caballero, Francisco J Gómez, and Juan López-López, "Road-traffic monitoring by knowledge-driven static and dynamic image analysis," *Expert Systems with Applications*, vol. 35, no. 3, pp. 701–719, 2008.

[2] C Setchell and EL Dagless, "Vision-based road-traffic monitoring sensor," *IEE Proceedings-Vision, Image and Signal Processing*, vol. 148, no. 1, pp. 78–84, 2001.

[3] Rita Cucchiara, Massimo Piccardi, and Paola Mello, "Image analysis and rule-based reasoning for a traffic monitoring system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 2, pp. 119–130, 2000.

[4] Sebastiano Battiato, Giovanni Gallo, Giovanni Puglisi, and Salvatore Scellato, "Sift features tracking for video stabilization," in *Image Analysis and Processing, 2007. ICIAP 2007. 14th International Conference on*. IEEE, 2007, pp. 825–830.

[5] Bruce D Lucas, Takeo Kanade, et al., "An iterative image registration technique with an application to stereo vision," 1981.

[6] Berthold KP Horn and Brian G Schunck, "Determining optical flow," *Artificial intelligence*, vol. 17, no. 1-3, pp. 185–203, 1981.

[7] Andres Sanin, Conrad Sanderson, and Brian C Lovell, "Shadow detection: A survey and comparative evaluation of recent methods," *Pattern recognition*, vol. 45, no. 4, pp. 1684–1695, 2012.

[8] Ahmed Elgammal, David Harwood, and Larry Davis, "Non-parametric model for background subtraction," in *European conference on computer vision*. Springer, 2000, pp. 751–767.

[9] Rita Cucchiara, Costantino Grana, Massimo Piccardi, Andrea Prati, and Stefano Sirotti, "Improving shadow suppression in moving object detection with hsv color information," in *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*. IEEE, 2001, pp. 334–339.

[10] Rudolph Emil Kalman et al., "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[11] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer, "Real-time tracking of non-rigid objects using mean shift," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*. IEEE, 2000, vol. 2, pp. 142–149.

[12] Naiyan Wang and Dit-Yan Yeung, "Learning a deep compact image representation for visual tracking," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., pp. 809–817. Curran Associates, Inc., 2013.

[13] Yi Wang, Pierre-Marc Jodoin, Fatih Porikli, Janusz Konrad, Yannick Benezeth, and Prakash Ishwar, "Cdnet 2014: an expanded change detection benchmark dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014, pp. 387–394.