

# ROBUST SEGMENTATION FOR OUTDOOR TRAFFIC SURVEILLANCE

Gonçalo Monteiro, João Marcos, Miguel Ribeiro, and Jorge Batista

Institute for System and Robotics  
Dep. of Electrical Engineering and Computers  
University of Coimbra - Portugal

## ABSTRACT

In this paper it is presented a robust segmentation process for detecting incidents on highways. This segmentation process is based on background subtraction and uses an efficient background model initialization and update to work 24/7. A cross-correlation based shadow detection is also used for minimizing ghosts. It is also proposed a stopped vehicle detection system based on the pixel history cache. This methodology has proved to be quite robust in terms of different weather conditions, lighting and image quality. Some experiments carried out on some highway scenarios demonstrate the robustness of the proposed solution.

## 1. INTRODUCTION

In order to ensure a safe and efficient driving, it is important to classify vehicles' behaviors and to understand their interactions in typical traffic scenarios. Not long ago, this burdensome task was performed by human operators at traffic control centers. However, increasing number of available cameras dictated the need for automatic traffic surveillance systems.

The work presented in this paper is part of an automatic traffic surveillance system [1] [2]. The primary goal of the system is to detect and track potentially anomalous traffic events on highways. The core of an incident detection system has to be an accurate and robust segmentation process. In this paper, it is presented a robust segmentation process for outdoor scenarios. This system is based on background subtraction, and uses two background models for contemplating the background variations. In order to avoid noise in the model, it is also used a third background model obtained by a median of  $n$  frames. Two distinct thresholds are used: a per-pixel threshold for a robust adaptation to the scene variations, twenty four hours a day, and a global threshold, that is the minimum value, to eliminate camera noise scenario features like shaking trees and camera pole vibrations. This threshold is estimated in the system initialization. It is also used a shadow/highlight detection algorithm based on cross-correlation to discard the blobs generated by lighting variation (moving clouds, artificial light changes, etc.). This segmentation process was used as input in a stopped vehicles detection system. This stopped vehicles detection system has three main phases. Firstly, there is the segmentation of all vehicles in the scene. Secondly, it is verified if there are any static pixels segmented during a certain period of time. Those static pixels are then grouped

into blobs. The static pixels identification is based on a pixel history cache analysis. Finally, a blob temporal validation is applied to discard false positives in traffic jam situations. If the validation succeeds, an alarm is triggered in the traffic telematic system.

## 2. SEGMENTATION PROCESS

Background/foreground detection is an essential component of most video surveillance systems involving object detection and tracking. Such systems require both robustness to lighting variation and computer feasibility. Most of the existing solutions either make strict assumptions about the scene or simply fail when handling abrupt lighting variations resulting from moving clouds or camera automated gain control. The proposed segmentation system will be described below. This methodology is based on five main parts: 1) Background modeling; 2) Foreground pixels validation criteria; 3) Shadows/Highlight removal; 4) Blob validation; and 5) Dynamic thresholding.

### 2.1. Background Modeling

In order to achieve a good segmentation in an outdoor environment it is necessary to have a background model that considers all scenes' variations not classified as foreground. Three different background images are used to model the background, namely the *primary* background ( $B_P$ ), the *secondary* background ( $B_S$ ) and the *median* background ( $B_M$ ). The  $B_P$  background has to have, at all times, the background model that is closest to the current frame. The  $B_S$  background is used to model objects classified as static but that can undergo small variations in position and/or in shape. Lastly, the  $B_M$  background is the pixel value median of the last  $n$  frames with a frame interval of  $\Delta$  frames. This background is the cleanest one of the three backgrounds due to its median properties.

#### 2.1.1. Initial Background Model Estimation

In [1] was used a weighted average to estimate the initial background model, but this solution is not robust to scenarios with high traffic density. Due to the statistical properties of the average, all samples contribute to the final estimation. In order to achieve an optimum initial background model, the system calculates the median of a stack of frames based on  $I$ , where  $\phi$  is the pixel index,  $I_t$  is the current frame,  $n$  is the buffer size and  $\Delta$  is the interval between samples acquisitions.

---

This work was supported by BRISA, Auto-estradas de Portugal, S.A.



**Fig. 1.** Estimation of the initial background model. a) a frame of the scene where background model was computed. b) background model estimation with an weighted average of 500 frames; c) background model estimation with the median of 50 frames. As one can see, the weighted average estimation is darker in the regions where there is a higher number of vehicles passing (top of the road).

From the undertaken experiments, the median background estimation proved to be more accurate than the weighted average estimation. Initially,  $B_P$  and  $B_S$  is equaled to this initial background estimation (see Fig. 1).

$$B_M(\phi) = \text{med}_\phi \left( I_\phi^t, I_\phi^{t-\Delta}, I_\phi^{t-2\Delta}, \dots, I_\phi^{t-(n-1)\Delta} \right) \quad (1)$$

## 2.2. Online Median Background Estimation

The online median background estimation is computed almost in the same way as the initial background model estimation but now this buffer always contains the last  $n$  samples. The only difference has to do with the samples' buffer size and the update frequency. Buffer sorting is a computationally heavy task, making it therefore necessary to minimize this task. In order to reduce the computational time, both  $n$  and the update frequency were reduced (namely  $n = 5$  and  $\Delta = 30$ ). In order to achieve good results with a small buffer, the samples' buffer is only updated if the related pixel was not classified as foreground in the current frame.

## 2.3. Online Primary and Secondary Background Update

The background update is one of the most important tasks in a 24/7 surveillance system working on outdoor scenarios. The update of the primary background ( $B_P$ ) and of the secondary background ( $B_S$ ) is carried out after the segmentation process. During the segmentation process, a mask matrix with the pixels classified as foreground ( $M_F$ ) is created and the background model ( $B_P$ ,  $B_S$  or  $B_M$ ) closest to the current classified background is identified for each pixel. In a first step, and in order to guarantee that  $B_P$  is the background model closest to the current classified background,  $B_P$  is swapped with  $B_S$  if  $B_S$  is more similar to the current background than  $B_P$ . In order to avoid  $B_P$  and  $B_S$  degradation, it is also verified if  $B_M$  is the model closest to the current background. Since  $B_M$  is a filtered background model, having it as the model closest to the current background means that  $B_P$  has some disturbance.  $B_P$  is equaled to  $B_M$  so that disturbances are eliminated. After this swap process,  $B_P$  pixels are updated with a weighted average using Eq. 2, where  $\eta_s$  is the integration factor of pixels classified as foreground and  $\eta_f$  is the integration factor of the pixels classified as background. The integration factor  $\eta_f$  is used for updating the

background pixel and is tuned to a fast adaptation to background variations. Nonetheless,  $\eta_s$  is much smaller than  $\eta_f$  to avoid the integration of passing objects in the object (namely  $\eta_f = 0.05$  and  $\eta_s = 0.0005$ ).

$$B_P^{t+1}(\phi) = \begin{cases} \eta_s I^t(\phi) + (1 - \eta_s) B_P^t & \text{if } \phi \in M_F \\ \eta_f I^t(\phi) + (1 - \eta_f) B_P^t & \text{if } \phi \in \overline{M_F} \end{cases} \quad (2)$$

The purpose of  $B_S$  is to store information about the pixels classified as background, but that have more than one color on its model. After the learning phase, this background is equal to  $B_P$  but in the detection phase and with the swap operation it is modeled with pixel variations if this exists. But in some situations it is possible that  $B_S$  can be wrongly modeled and in this case if it is not corrected it can originate mis-detections on the segmentation process. To maintain  $B_S$  clean from wrong estimation, it is analysed at all iterations the last time that the model was the closest model to the current background. If the  $B_S$  model is not used during more than  $k$  iterations, it is equaled to  $B_P$ . This way, the model is maintained only if it is frequently used.

## 2.4. Foreground Pixels Validation Criteria

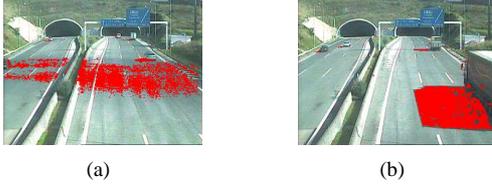
Most systems compute the difference between the current frame and the background image and consider as targets the pixels above a certain threshold. Then, neighborhood pixels are clustered to form possible foreground regions. This process usually leaves gaps that might lead to erroneous foreground detection. Morphology can be used to fill in these gaps. However, threshold-with-hysteresis (TWH) is preferred since it is a more accurate algorithm that only fills meaningful gaps. The thresholded difference image ( $D_{th}(\phi)$ ) is defined by Eq. 3 where  $i \in \{P, S, M\}$  and  $ac$  represents the pixel activity.

$$D_{th}(\phi) = \begin{cases} \min_i |I^t(\phi) - B_i^t(\phi)| & \text{if } \phi \in ac \\ 0 & \text{if } \phi \in \overline{ac} \end{cases} \quad (3)$$

The *Quasi-Connected Components* (QCC) [3] combines TWH with gap filling and connected component labeling. One of the techniques to keep a fast QCC process consists of a reduction in resolution that also provides gap filling. Based on  $D_{th}(\phi)$ , a low resolution image  $P$  (parent image) is created. The thresholded difference image is divided into blocks of  $2 \times 2$  pixels. Each pixel value of  $P$  stores the number of pixels above  $T_l$  and  $T_h$  of the corresponding block in the thresholded difference image  $D_{th}(\phi)$ , where  $T_l$  is a lower threshold and  $T_h$  a higher threshold (explained in detail in Sec. 2.7). Two criteria were used to define the target blocks by analyzing the parent image: a) a block has to have at least one pixel above  $T_h$ ; b) a block has to have at least two pixels above  $T_l$  and has to have a neighbor block with at least one pixel above  $T_h$ .

## 2.5. Shadow/Highlight Removal

In order to detect shadow/highlight on the segmented regions it is computed the color normalized cross-correlation (CNCC). This solution measures the similarity between  $B_P^t$  and  $I^t$ . The color of the pixel is represented in the biconic HSL space in order to split the color information from the brightness values [4]. To measure similarity it is calculated



**Fig. 2.** a) Identification of a shadow induced by a cloud passing in the direction of the image bottom (red pixels). b) Identification of shadows induced by passing vehicles.

the HSL space, not in polar coordinates, but through the projection of the  $(R, G, B)$  vector onto the chromatic  $(H, S)$ -plane to compute the Euclidean values of hue and saturation. It is denoted the representation in Euclidean coordinates with  $(h, s)$ . The projected  $h, s$  part is scaled, so that its length equals the saturation of the HSL color space. It is also denoted that  $c^F = (h^F, s^F, L^F)$  is the pixel components for a foreground pixel and  $c^B$  the same for the background image. It is defined on Eq. 4 the CNCC over a window with size  $M \times N$  for the two color pixels  $c_{xy}^F, c_{xy}^B$  at an image position  $(x, y)$ .

$$CNCC_{x,y} = \frac{\sum_{i,j} (c_{xy}^F \bullet c_{xy}^B) - MN \overline{L^F L^B}}{\sqrt{VAR^F VAR^B}} \quad (4)$$

with

$$VAR^k = \left( \sum_{i,j} (c_{xy}^k \bullet c_{xy}^k) - MN \overline{L^k}^2 \right) \quad (5)$$

where  $i$  ranges from  $x - \frac{M-1}{2}$  to  $x + \frac{M-1}{2}$  and  $j$  from  $y - \frac{N-1}{2}$  to  $y + \frac{N-1}{2}$ ,  $\overline{L^F}$  is the average intensity in the image  $F$  over the  $M \times N$  window,  $k \in \{F, B\}$  and

$$c_{xy}^F \bullet c_{xy}^B = (h_{ij}^F, s_{ij}^F) \circ (h_{ij}^B, s_{ij}^B) + L_{ij}^F L_{ij}^B \quad (6)$$

The operator  $\circ$  denotes the scalar product, with negative values set to zero. This cross-correlation algorithm has a low performance in image areas with low color information. An improvement was introduced to handle this problem, namely it is verified if the analysed color has significant color information. If  $S < \tau \times V$  the pixel color is classified as gray level,  $s$  is assigned to zero. This way, the cross-correlation process just analyses the texture information of the pixel neighborhood. For this analysis it is used the saturation ( $S$ ) and value ( $V$ ) of the HSV color space representation. The value  $\tau$  was empirically set to 0.1. Fig. 2b and 2c show the outcome of the shadow detection algorithm.

## 2.6. Blob Validation

After the foreground pixels identification and shadow/highlight removal, a grouping/labeling algorithm is applied to identify the possible objects in the scene. In order to minimize the false positive detections, each of the possible blobs is validated before labeling it as object/vehicle. Firstly, blobs are filtered by area. Blobs with an area below  $A_{min}$  are discarded ( $A_{min}$  depends on the scene). Secondly, the estimation of the optical flow in the blob area occurs [5] to determine if the blob is a moving object. If no motion is detected in the blob, it is verified if this is a stopped blob. A stopped blob can only

be validated if it is detected more than two times in the same position.

## 2.7. Dynamic Thresholding

Four different thresholds are used in the segmentation process: the per-pixel threshold ( $T_{pp}$ ), the low threshold ( $T_l$ ), the high threshold ( $T_h$ ) and the global threshold ( $T_g$ ). The  $T_{pp}$ , attempts to account the scene noise at a pixel through time, e.g., when the camera shakes, the edge pixels will have a significant intensity change; in these cases, we will increase  $T_{pp}$ . The  $T_g$  is the minimum value to eliminate camera noise and depends on the scenario. The low threshold  $T_l$  is the sum of  $T_g$  and the per-pixel threshold  $T_{pp}$  that dynamically adapts its value along the process (Eq. 7). And  $T_h$  is obtained by Eq. 8 ( $\Psi = 1.25$ ).

$$T_l(\phi) = T_g(\phi) + T_{pp}(\phi) \quad (7)$$

$$T_h(\phi) = T_l(\phi) \times \Psi - T_l(\phi) \quad (8)$$

### 2.7.1. Thresholds Initialization

The threshold  $T_l$  is empirically initialized with 10. In Section 2.1.1 a buffer of  $n$  frames is used to estimate the initial background model. The same sorted buffer ( $B_p(i, j)$ ) is used to estimate  $T_g$  [6]. Where  $p$  is position inside the sorted buffer  $B$  of pixel  $(i, j)$  and, consequently,  $B_{\frac{n+1}{2}+1}$  of the data. The threshold  $T_g$  is computed by Eq. 9. Where  $\lambda$  is a fixed multiplier, while  $g$  is a fixed scalar (namely  $\lambda = 2$  and  $g = 5$ ).

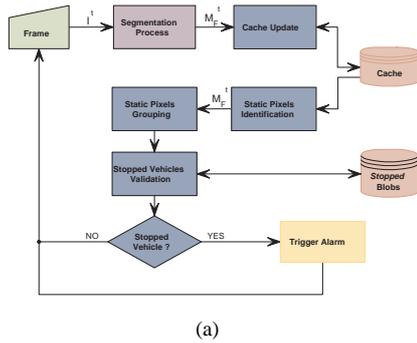
$$T_g(\phi) = \lambda \left( B_{\frac{n+1}{2}+g}(\phi) - B_{\frac{n+1}{2}-g}(\phi) \right) \quad (9)$$

### 2.7.2. Thresholds Update

Just like background images, threshold levels are updated on the last step of the frame processing, i.e., after determining which pixels belong to the foreground and to the background. If a pixel value is bigger than  $T_l(\phi)$  and is not classified as foreground, then it is labeled as a noisy pixel and  $T_{pp}$  is increased by  $T_{inc}$ . By increasing  $T_{pp}$  of noisy pixels, the system sensibility is reduced. When pixels are classified as background more than  $\mu$  frames their  $T_{pp}$  is decreased by  $T_{dec}$ , therefore increasing their sensibility. To avoid the system instability  $T_{pp}$  should be much bigger than  $T_{dec}$ . Those values were set to  $T_{inc} = 8, T_{dec} = 1$  and  $\mu = 5$ .

## 3. STOPPED VEHICLES DETECTION

The main assumption for the stopped vehicles detection method is that some foreground pixels with the same color that appears during a large period, are probably part of a stopped vehicle. After a blob has been formed by these pixels grouping, it will only be considered as stopped if it holds approximately the same position and dimensions for a pre-defined validation time. The studied method has two main phases: stopped pixels recognition and a subsequent stopped vehicle validation process, that can culminate with a real stopped vehicle detection (see Fig. 3).



**Fig. 3.** Overview of the stopped vehicles detection process.

### 3.1. Stopped Pixels Recognition

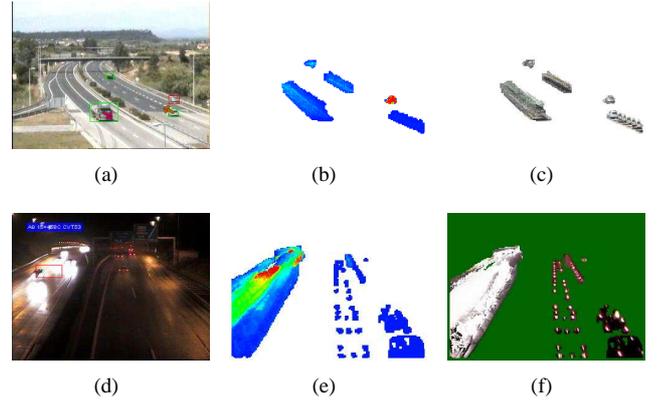
The stopped pixels are identified by the analysis of the segmented pixels. A pixel is classified as *static* if it is labelled as foreground with the same color with a certain frequency. A pixel history cache [7] is used to verify the pixel color frequency. For each pixel it is kept a set of colors that appears at least one time in the last  $T_h$  frames of the video signal. Cache is an array of the same size of the image. Each entry corresponds to a pixel and it has a list of Codewords (Codebook). A Codeword saves the RGB color components and a validation buffer that keeps occurency history of a color in the pixel in the last  $\alpha$  frames. Occlusions, vibrations or lighting changes that can temporarily hide or change a vehicle's pixel color are taken into account with the pixel history cache.

For each foreground pixel it is checked if the actual color matches any Codeword present in the Codebook. A match event exists if the euclidean distance between two colors, in RGB space, is below  $\epsilon = 30$ . If there is a match, with a Codeword in Cache, RGB components are updated by a weighted average, and the validation buffer is updated with '1'. For all the other Codewords present in the Codebook the buffer is updated with '0'. If there is no Codeword matching, a new Codeword is inserted in the pixel Codebook. The validation buffer of the Codewords of non segmented pixels are also updated with '0'. Finally, all the Cache Codewords that not appear at least one time in the last  $T_h$  frames are deleted ( $T_h = 25$ ).

A pixel is validated as static if exists a Codeword on its Codebook with  $\beta$  occurrences in the last  $\alpha$  frames, where  $\alpha$  is the validation buffer size (namely  $\beta = 40$  and  $\alpha = 64$ ). This strategy handles the occlusion problem mentioned above. However, when a stopped vehicle has its four emergency lights activated, the vehicle lights region has intermittently two or more colors. To deal with this particular, but frequent, situation, it is also allowed that a pixel is validated as stopped whenever two Codewords have more then  $\beta/2$  occurrences in  $\alpha$  frames. Fig. 4 b) shows chromatically the number of occurrences of the most frequent Codeword.

### 3.2. Stopped Vehicle Validation

Once identified the static pixels, they are grouped to make a blob. The validation process of a possible stopped vehicle starts if the blob has a significant area. Not all the blobs that result from static pixels grouping are really stopped vehicles. They can be vehicles moving slowly in the image or wrong segmented regions. The main idea of the validation process

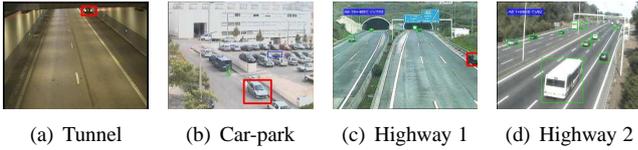


**Fig. 4.** Stopped vehicles detection system experiments. a) and d) stopped vehicle detection; b) and e) chromatic representation of the number of occurrences of the most frequent Codeword; c) and f) color of the most frequent Codeword. The right example represents a false positive due to the high frequency of the color white in pixels (vehicles lighting reflected in the wet road). In f) the absence of Codewords associated with a pixel is represented by the green color.

is to “track” the blob in validation and verify if it stays in the same position with the same dimensions during a certain validation period. In this situation, the blob is considered as a stopped vehicle, and an alarm is triggered. Validation period should be inversely proportional to the average velocity on the region where the stopped blob is. This average velocity is estimated during the learning phase. This approach avoids false positives at some image regions where vehicles hold more time due to a image perspective effect in the vanishing point.

## 4. EXPERIMENTAL RESULTS

The system was tested with a real set of image sequences from highways traffic surveillance cameras with different weather conditions, lighting, image quality and fields of view. The system was also tested in real time in some Portuguese high traffic density highway scenarios and in a car-park at Coimbra University. Table 4 shows the experimental results obtained in some different scenarios. The tunnel situation was acquired from the VSSN06 dataset (8 minutes) [8]. The car-park scenario and the highway scenarios were analysed during 24 hours (see Fig. 5). In Fig. 4, it is illustrated the vehicles detection (green box) and the stopped vehicles identification (red box). In the experiments performed to the system it was verified that the mis-detection of stopped vehicles was due to the lack of image contrast. Most of the false positives are related to a wrong segmentation result. Another type of false positives is represented in Fig. 4d, this is frequent in rainy situations due to the saturation of the image with the vehicles lighting reflected on the road. The segmentation process is able to detect vehicles under lighting variation in a robust way, except in situations of abrupt lighting variation (eg. artificial lighting turned on or off). The system can segment vehicles in outdoor scenarios and detect stopped vehicles over a  $320 \times 240$  pixel image at 9 fps on a 3.2 GHz *P4 Intel Processor* under Linux OS.



**Fig. 5.** Scenarios where the tests were performed. Bounding box legend: Green - moving objects; Blue - stopped vehicles (in the last 5 minutes); Red - vehicles validated as stopped in the present frame.



**Fig. 6.** Outcome of the segmentation process.

## 5. CONCLUSIONS

In this paper, it is proposed a methodology to segment vehicles in outdoor scenarios based in background subtraction. This system uses an efficient background model initialization and update to work 24/7. A stopped vehicles detection system based on a pixel history cache is also presented.

The experiments conducted on a large number of scenes demonstrate that this system is able to segment vehicles robustly under different weather conditions, lighting, image quality and image compression variation. This segmentation process proved to be a good basis for an incident detection system. The proposed stopped vehicles detection system achieved as well a good performance in the tested conditions described above.

## 6. REFERENCES

- [1] J. Batista, P. Peixoto, C. Fernandes, and M. Ribeiro, "A dual-stage robust vehicle detection and tracking for real-time traffic monitoring," in *IEEE Int. Conference on Intelligent Transportation Systems*, 2006.
- [2] G. Monteiro, M. Ribeiro, J. Marcos, and J. Batista, "Wrong way drivers detection based on optical flow," in *IEEE ICIP 07*, 2007.
- [3] T.E. Boulton, R. Micheals, X. Gao, P. Lewis, C. Power, W. Yin, and A. Erkan, "Frame-rate omnidirectional surveillance and tracking of camouflaged and occluded targets," in *IEEE Workshop on VS'99*, 1999.
- [4] Daniel Grest, Jan-Michael Frahm, and Reinhard Koch, "A color similarity measure for robust shadow removal in real-time," in *VMV 2003*, 2003.
- [5] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *DARPA Image Understanding Workshop*, 1981, pp. 121–130.
- [6] Simone Calderara, Andrea Prati, and Rita Cucchiara, "Reliable background suppression for complex scenes," in *VSSN 06*, 2006.

**Table 1.** Experiments conducted on the global system.

	Total	Stopped	Hits	FP
Tunnel	56	4	4	1
Car-Park	1012	85	82	18
Highway 1	-	0	0	7
Highway 2	-	2	2	4

- [7] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis, "Real-time foreground-background segmentation using codebook model," in *Real-time Imaging*, 2005, pp. 167–256.
- [8] "Call for algorithm competition in foreground/background segmentation," [http://mmc36.informatik.uni-augsburg.de/VSSN06\\_OSAC/](http://mmc36.informatik.uni-augsburg.de/VSSN06_OSAC/), January 2008.