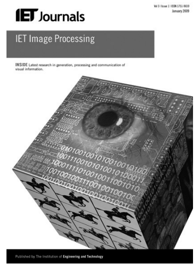
Published in IET Image Processing
Received on 19th January 2012
Accepted on 7th September 2013
doi: 10.1049/iet-ipr.2013.0496



ISSN 1751-9659

Estimating the queue length at street intersections by using a movement feature space approach

Pablo Negri

Instituto de Tecnologia, UADE, CONICET, Av. Rivadavia 1917, Lima 717, Buenos Aires, Argentine
E-mail: pnegri@telecentro.com.ar

Abstract: This study aims to estimate the traffic load at street intersections obtaining the circulating vehicle number through image processing and pattern recognition. The algorithm detects moving objects in a street view by using level lines and generates a new feature space called movement feature space (MFS). The MFS generates primitives as segments and corners to match vehicle model generating hypotheses. The MFS is also grouped in a histogram configuration called histograms of oriented level lines (HO2 L). This work uses HO2 L features to validate vehicle hypotheses comparing the performance of different classifiers: linear support vector machine (SVM), non-linear SVM, neural networks and boosting. On average, successful detection rate is of 86% with 10^{-1} false positives per image for highly occluded images.

1 Introduction

Nowadays, advances in information technology and electronics make it possible to control urban traffic in real time. This control has numerous advantages, such as the reduction of drivers' travel time, fuel consumption and pollution, all contributing to a better and more rational use of a transport network. An Urban Traffic Control System (UTCS) project in development progress at the UADE laboratories (Argentina), seeks the automatic regulation of traffic in a town or a neighbourhood, measuring and operating with 'intelligent traffic lights'. Its main objective is to adapt the computation of adequate green times to variations in traffic load. To do so, full condition of the traffic network should be known at all times. Instead of installing traffic sensors in the entire network, such a state is estimated by measuring at specific intersections, and the load of the other intersections is provided by simulation. With this information, the system defines all the green times to maximise the vehicle flow, thus reducing global congestion.

Historically, inductive loops have been used to measure the traffic load and queue length [1]. In spite of their good performance, modern systems switch to video camera detection obtaining similar or better results. Video cameras are not only cheaper, but simpler to install and maintain. Computer vision is widely applied in transportation systems, such as traffic congestion detection [2, 3], queue length measurement at traffic lights [4–7], lane occupancy estimation [8], vehicle classification [9, 10] and trajectory learning and prediction.

In general, vision-based traffic monitor systems use a camera pointing to a fixed point and the traffic load is estimated by using three basic methodologies: time differences computed between consecutive frames at times t and $t + \alpha$, background subtraction by using an image of the scene without vehicles and edge detection based on variation in brightness.

Fathy and Siyal [4] combine the three methods to measure the queue and delay length. The time difference and background difference methods detect motion in the scene by identifying a deviation in the intensity value of the same pixel in two different captures. Pixels where deviation is significant are grouped by a neighbourhood criterion in regions or blobs creating a binary map, and identifying moving objects. The presence of vehicles is confirmed by edge detection.

Zanin *et al.* [2] use time difference and edge detection. The presence of edges in a road area suggests the presence of a vehicle, and thus the length of the queue can be inferred. Motion detection makes it possible to infer whether vehicles are moving or stationary, indicating traffic congestion.

Other methods [8, 10] set up an adaptive reference model generated by temporal learning based on the static information of the scene. The new input images are compared with the reference by applying a difference function, and the resulting pixels represent the movement. Buch *et al.* [10] used motion silhouettes and a three-dimensional (3D) model to detect and classify vehicles. In the work of Pang *et al.* [8], the boundaries of the motion blobs are analysed to count the vehicles on the road. Its method is prepared to overcome the cases of occlusions generated by vehicle queues when the camera is installed at a low angle.

Yang *et al.* [7] also tackle vehicle occlusions proposing a windshield-based vehicle detection algorithm. They generate hypotheses by using a confidence map which combines the likelihood of a windshield model and a shape and edge matching function. A tracking procedure eliminates false alarms.

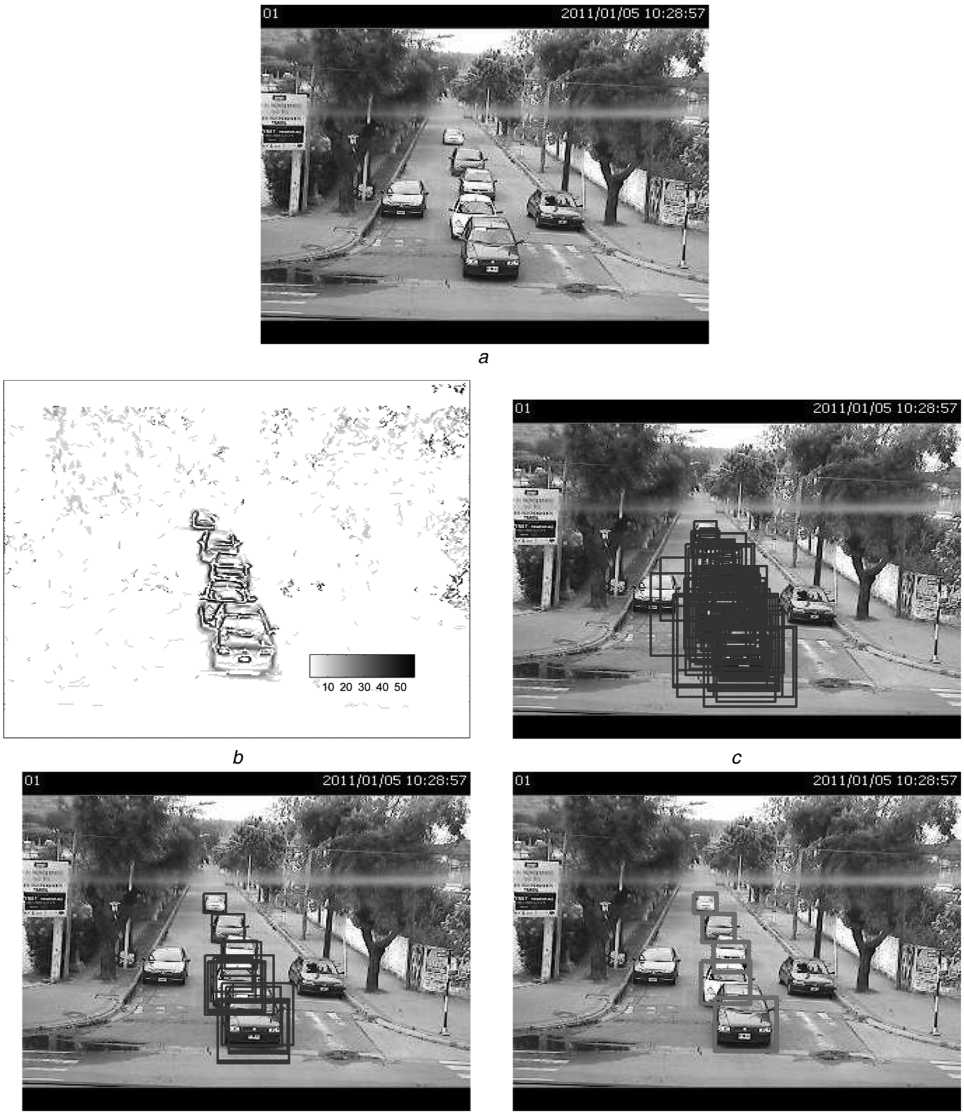
This paper addresses vehicle detection in outdoor sequences captured by a fixed remote camera installed on a traffic light at a low angle. The system should be robust to quick and significant changes to the scene (e.g. shadows, weather conditions), to vehicles which should not be considered (as parked cars), to the presence of many other moving objects (e.g. people etc.) and to the camera movement caused by blowing wind or traffic vibrations. In addition, the desired response time for an online application should be between 1 and 5 fps.

The proposed detection method consists of four stages: motion detection, hypothesis generation, hypothesis validation and final filtering, as shown in Fig. 1.

In the first stage, motion detection uses a level line-based approach [11, 12], illustrated in Fig. 1*b*, generating a movement feature space (MFS). We have developed the MFS based on level lines to obtain an adaptive background model, preserving the orientation of the level lines and a measure similar to the gradient module. Working on the MFS has interesting advantages: it adapts well to slow changes in the scene and is also robust to rapid variations, for example, illumination changes, weather conditions and so on. In such situations, the appearance of vehicles on the MFS does not change significantly compared with normal conditions, and is perfectly well detected by the classifiers.

Hypotheses or regions of interest (RoIs) are generated in the second stage, restricting the search space to some positions within the image, as shown in Fig. 1*c*. The algorithm employs the information from the MFS and a vehicle model based on the segments and the corners.

The information inside each RoI is encoded by using a family of histogram of oriented level lines (HO2 L) descriptors calculated on the MFS, and grouped in a configuration based on the R-HOG [13]. The HO2 L **Q1** features obtained from the MFS allow a multi-scale vehicle detection, avoiding the construction of a dense pyramid of



a b c d e

Fig. 1 *Overall sequence of the vehicle detection algorithm*

a Original image
b Motion detection
c Hypotheses generation
d Hypotheses validation
e Final results

subsampled versions of the input image that is very costly in terms of calculation time. They can also be computed very quickly by using an integral histogram [14].

The third stage of the system performs RoIs validation by using a classifier discriminating between vehicle and non-vehicle classes. This paper explores four different classifiers by evaluating the performance of the HO2 L feature space in classification and processing time. Validated RoIs, as shown in Fig. 1*d*, are finally grouped using non-maximal suppression algorithm. Those RoIs are considered the system outputs (see Fig. 1*e*).

The structure of the paper is as follows: Section 2 details the methodology to obtain the MFS, develops the hypothesis generation algorithm by using the vehicle model and details the validation classifiers. Different experiments are described in Section 3. The system results are discussed in Section 4, while Section 5 concludes the paper.

2 Methodology

2.1 Movement feature space based on level lines

Motion detection in video sequences can be performed by using background subtraction algorithms that model an image reference by capturing the static information of the scene. The presence of a new object in the scene is stated if there exists any difference against the model.

The algorithm used in this paper is based on the work of Bouchafa [15] and Aubert *et al.* [12] using level lines as primitives for the reference model. This methodology has the flexibility to adapt to changes in the scene (e.g. new objects, shadows, modifications etc.).

2.1.1 Definition of level lines: Let I be an image with $h \times w$ pixels, where $I(p)$ is the intensity value at pixel p whose coordinates are (x, y). The (upper) level set X_λ of I for the level λ is the set of pixels $\boldsymbol{p} \in I$, so that their intensity is greater than or equal to λ,

$$X_\lambda = \{\boldsymbol{p}/I(\boldsymbol{p}) \geq \lambda\}$$

For each λ, the associated level line is the boundary of the corresponding level set X_λ, see [11]. Finally, we consider a family of N level lines C of the image I obtained from a given set of N equally spaced thresholds $\Lambda = \{\lambda_1, \ldots, \lambda_N\}$. From these level lines we compute two arrays S and O of order $h \times w$ defined as follows:

- $S(p)$ is the number of level lines C_λ superimposed at p. When considering all the grey levels, this quantity is highly correlated with the gradient module at p.
- $O(p)$ is the gradient orientation at p. In this paper, it is computed in the level set X_λ by using a derivative filter of 5×5 pixels (the orientations are quantised in η values). For each pixel p, we have a set of $S(p)$ orientations values, one for each level line passing over p. The value assigned to $O(p)$ is the most repeated orientation in the set.

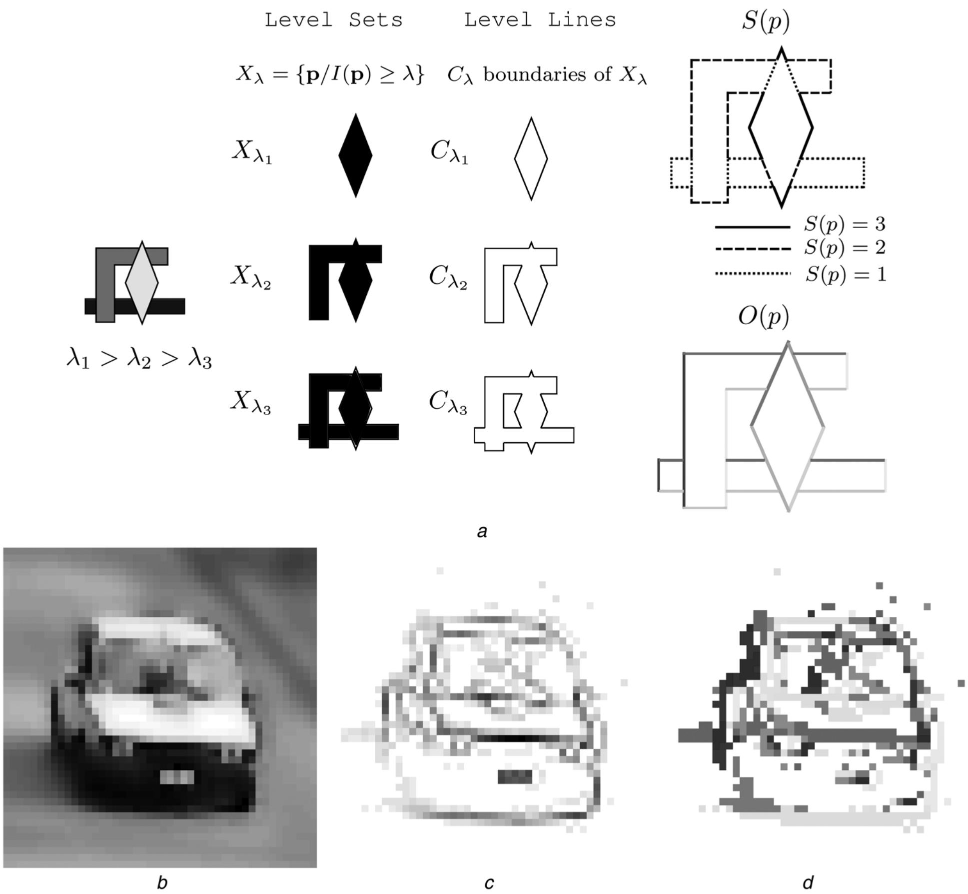


Fig. 2 *Level lines calculation of a vehicle image sample*

a Level lines extraction
b Vehicle image
c S_t values
d O_t values

doi: 10.1049/iet-ipr.2013.0496

Generally, in the practical implementation, only those pixels for which $S(\boldsymbol{p})$ is greater than a fixed threshold δ are considered, simplifying the analysis and preserving meaningful contours.

Fig. 2*a* shows the level lines extraction from a simple geometric configuration. It also has two arrays, $S(p)$ with the number of superimposed level lines, and $O(p)$, for which each colour represents a gradient orientation. Fig. 2 shows S_t and O_t for a vehicle sample.

2.1.2 Movement detection: As described in [16], level lines have many properties: they are Jordan curves, they have a hierarchical representation, they locally coincide with the scene contours and they are invariant to contrast changes.

The last property means that a regular contrast change (monotonic and upper semi-continuous) can either create or remove level lines from a pixel, changing the $S(p)$ quantity, but it could never create a new level line intersecting the original ones [12]. This is crucial because we will use level line intersections to detect movements.

Bouchafa [15] and Aubert *et al.* [12] defined an adaptive background reference model, composed of the set of pixel p which are stable over a horizon of time, together with the corresponding values $O(p)$. More precisely, given a horizon of time T we define R_t as the set:

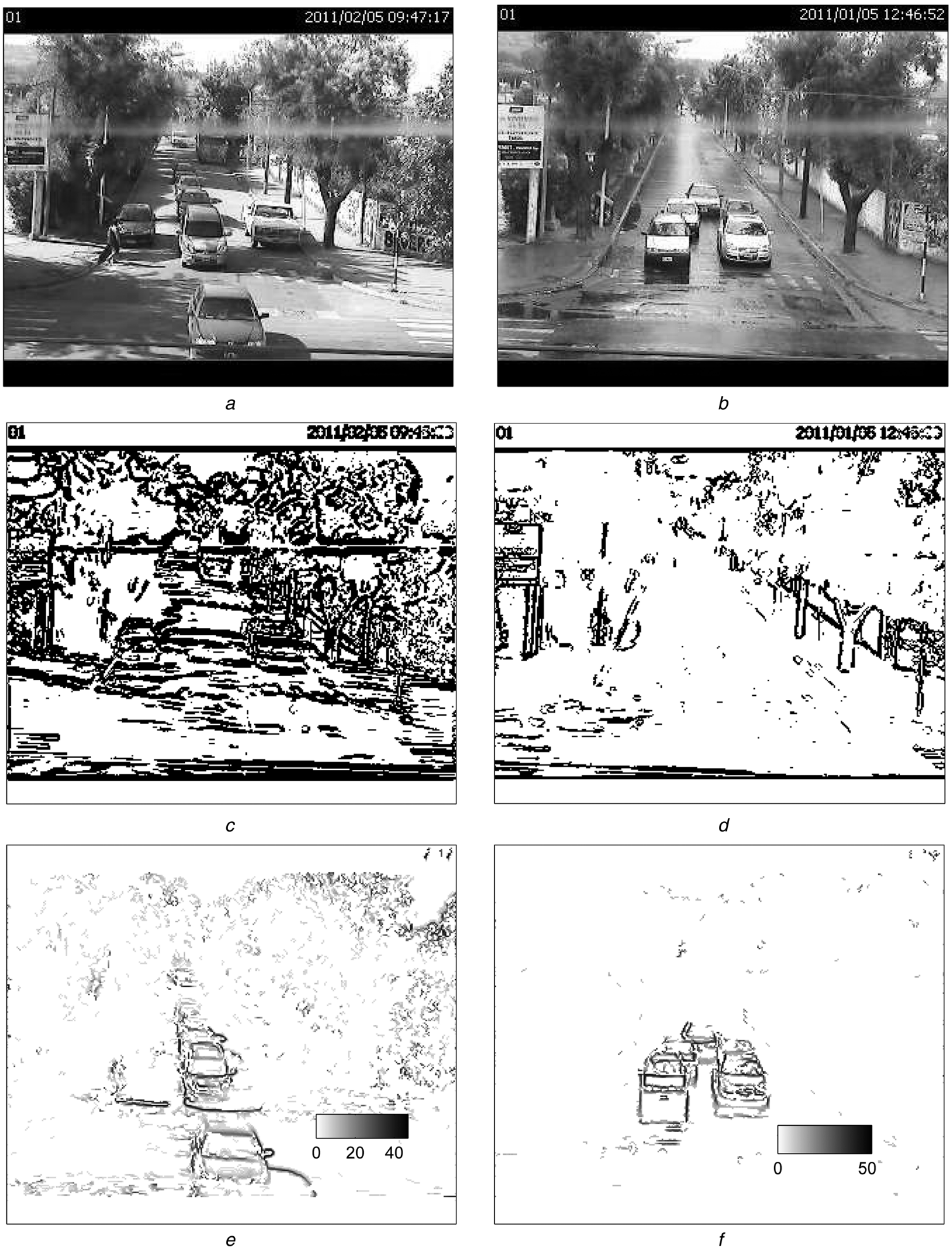


Fig. 3 *Background model reference and movement detection, with N = 80, η = 8 and δ = 1 for (c), (d), and δ = 3 for (d), (f)*

a Original
b Original
c Reference ($\delta = 1$)
d Reference ($\delta = 3$)
e Movement detection ($\delta = 1$)
f Movement detection ($\delta = 3$)

IET Image Process., pp. 1–11
doi: 10.1049/iet-ipr.2013.0496

$$R_t = \{ p \in C : S_{t-1}(p) > \delta, S_{t-2}(p) > \delta, \ldots, S_{t-T}(p) > \delta \wedge O_{t-1}(p) = O_{t-2}(p) = \cdots = O_{t-T}(p) \} \tag{1}$$

Thus, at time t, the input frame generates the pair $S_t(\boldsymbol{p})$ and $O_t(\boldsymbol{p})$ of the meaningful level lines: $\{S_t(p) > \delta, \forall \in \boldsymbol{p}\}$. Pixel $p \in \boldsymbol{p}$ is considered as a moving level line pixel if it is verified that

- $p \notin R_t$,
- $p \in R_t \wedge O_t(p) \neq O^R_{t-1}(p) \neq$ (where $O^R_{t-1}(p)$ is the orientation in R_t at the location of pixel p).

These pixels will make up the binary set D_t. In practice, the equality constraints in the definition of the reference space R_t can be relaxed to allow for small variations of orientation because of noise or other perturbations (see Bouchafa [15] and Aubert *et al.* [12] for details).

Fig. 3 shows two examples of the adaptive reference model. The first row shows the original capture, whereas the second one illustrates the reference model. The last row presents the detected set D_t with a grey level corresponding to the value of S_t. Note that for Fig. 1, parked cars and shadows belong to the reference model and do not appear in D_t.

Below, we will focus the analysis only on pixels in the detected set D_t, and their values of S_t and O_t. This set can be considered as a virtual image with two associated scalar fields, or a kind of feature space referred to as movement feature space, or MFS.

2.2 Hypothesis generation in the MFS

The hypotheses generation procedure (HG) [17] uses primitives of simple calculation as horizontal segments [17], symmetry [18] and corners [19], to define vehicle locations by exploiting the fact that vehicles are rigid bodies principally defined by straight lines. Those primitives can be combined to match simple models: ‘U’ shape [20] or deformable templates [21].

Here, it is considered as an a priori model of a vehicle, inspired in the configuration proposed by Collado *et al.* [21], and depicted in Fig. 4*a*. It is composed of three horizontal segments h_i, two vertical segments v_j and four corners belonging to the windshield e_k. Geometrical relations among those elements, distances and sizes, were statistically estimated from a labelled dataset.

The principal advantage of using the MFS in the HG step, is that parked cars do not generate hypotheses because they belong to the background model. This represents an important advantage over still detection algorithms [13, 22, 7]. Still detection methodologies must have an additional procedure eliminating those cases. For instance, if a car is detected in the same position for a long period of time the system can assume that it is parked. Although in comparison with other motion detection algorithms as blobs [8], they do not provide internal information as the MFS, for example, segment h_2 and corners e_3 and e_4 in the model.

2.2.1 Hypothesis generation using segments: The first configuration analysed is the ‘U’ shape using segments h_1, v_1 and v_2, of our model. The orientation of the horizontal segment h_1 corresponds to the transition from a lit region (road) to a dark one (vehicle shadow). After identifying a segment with this orientation, it becomes the lower side of a square RoI, and the algorithm looks for

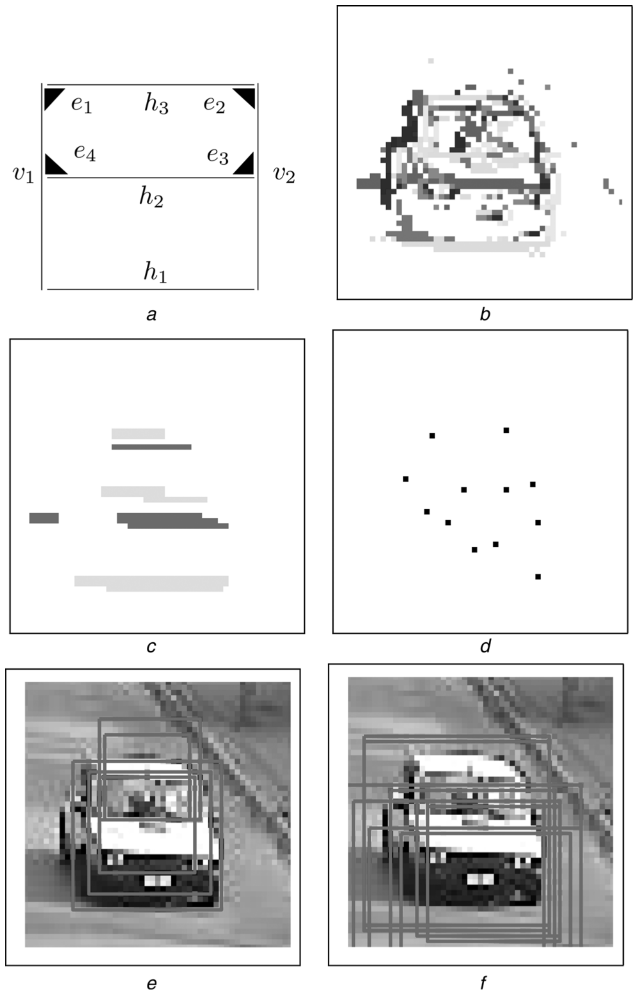


Fig. 4 *Vehicle model and RoI generation*

a Vehicle model
b Vehicle level lines
c Horizontal segments
d Corners
e RoIs segments
f RoIs corners

vertical segments near their boundaries. The presence of vertical segments defines the size of the square RoI. Otherwise, the RoI is not generated.

The drawback of this ‘U’ shape is represented by partially occluded vehicles in the queue, because the bottom of the vehicle is not visible and no shadows are cast (see Fig. 3*b*).

Other RoIs can be generated by using the other horizontal segments h_2 and h_3 which are the lower and the upper limits of the windscreen. This time, each horizontal segment having any orientation, will generate two RoIs. The first RoI is created by considering the segment as h_2 and the RoI is placed by taking the segment as the middle position. The second RoI is generated by considering the segment as h_3, and the RoI is placed with the segment as the upper limit.

2.2.2 Hypothesis generation using corners: Fig. 3*b* shows that occlusions for queued vehicles can be severe when the sequences are captured by using low angle cameras. Yang *et al.* [7] proposed a windshield identification procedure to minimise the occlusion problem. In our sequences, windshields are almost always visible showing at least three corners e_k. We use those primitives on the basis of the vehicle model configuration (see

doi: 10.1049/iet-ipr.2013.0496

Fig. 4*a*) to generate additional RoIs and increase the probabilities of finding queued vehicles. Corner detection is conducted by employing Achard's methodology [23] which is well suited for vectorial operations in our MFS. The application of this algorithm on the MFS is more robust under contrast variations and less time consuming than others methods, like the Harris corner detector.

Here, we consider vectorial field $G(p)$ at pixel p given by the vector of modulus $S(p)$ and direction $O(p)$. Achard's corner detector is based on the assumption that in neighbourhood V_p of a corner p, the average of the cross product between $G(p)$ and all the vectors $G(q)$ where $q \in V_p$, should be higher than the same magnitude around a pixel that is not a corner.

The average cross product in the neighbourhood V_p can be computed as

$$K = I_x^2\langle I_y^2\rangle + I_y^2\langle I_x^2\rangle - 2I_xI_y\langle I_xI_y\rangle$$

where $\langle\rangle$, is the convolution with a 5×5 mask and all the elements are equal to 1, except for a zero in the centre. Assuming that orientation O is given in radians, the values I_x and I_y (in our case) are defined as

$$I_y(p) = S(p)\sin(O(p)) \tag{2}$$

$$I_x(p) = S(p)\cos(O(p)) \tag{3}$$

Thus, in order to find the corners, we look for the local maxima of K.

To generate the RoIs corresponding to the windshield, we start searching two co-linear corners along the horizontal axis. If we find a third corner with the same vertical coordinate as one of the previous ones, an RoI is created by the three corners in the configuration of our vehicle model. Fig. 4*f* shows the RoIs generated from these primitives.

2.3 Hypotheses validation by using a classifier

As shown in Fig. 4, the number of RoIs generated is quite significant. In the hypotheses validation (HV) step, RoIs positions are tested to verify their correctness in order to eliminate false alarms [17].

2.3.1 Histograms of oriented level lines feature space: The feature space encoding the information inside the RoI is calculated by using the MFS. It results in a concatenated set of HO2 L, which is computed in a configuration similar to the R-HOG proposed by Dalal and Triggs [13].

The square RoI is subdivided into two grids of 6×6 and 3×3 non-overlapped cells. Within each cell r_i, the $\text{MFS}^{\text{HO2 L}}$ descriptor is the histogram $\boldsymbol{h}$ having η bins, one for each orientation. For each bin o of $\boldsymbol{h}$, we add all the $S_t(p)$ values for the p with this orientation, $\boldsymbol{h}(o) = \left\{\sum_{p\in r_j} S_t(p)/O(p) = o\right\}$.

A grid of 2×2 continuous cells generates a block histogram of the four concatenated histograms $\boldsymbol{h}$, having 4η bins in all. The blocks are then normalised by using the $L2$-Norm: $\boldsymbol{v} \to \boldsymbol{v}/\sqrt{\|\boldsymbol{v}\|_2^2 + \epsilon}$.

Thus, each RoI generates 29 blocks of $\text{MFS}^{\text{HO2 L}}$ concatenated descriptors. The feature vector has $29 \times 4 \times \eta$ elements in all, and corresponds to the input for the classifiers.

2.3.2 Vehicle classifiers: In this work, four different classifiers evaluate the ability of the HO2 L feature space for vehicle detection. It is employed by the OpenCV implementation of each classifier [24], and a two rounds bootstrapping approach [25] is adopted in place of the learning phase.

Linear support vector machine (SVM): This is a hyperplane-based classifier called support vector machine (SVM) [26]. For linearly separable problems there will exist a unique optimal hyperplane that maximises the separation margin separating the training data on the feature space (vehicles against non-vehicle classes). Let $\{\boldsymbol{x}_i, y_i\}$ be a training dataset, where $y_i \in \{-1, +1\}$, $\boldsymbol{x}_i \in \Re^d$. Classification is formulated as

$$y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) - 1 \geq 0 \tag{4}$$

where $\boldsymbol{w}$ is the normal to the hyperplane. $\boldsymbol{x}_i$ at which (3) equals zero are called support vectors and define two parallel planes on both sides of the hyperplane separated by a margin $2/||\boldsymbol{w}||$. After the SVM training, $\boldsymbol{w}$ is calculated from (4), and stored. This vector will always have the same dimension d, no matter the number of support vectors that define it. The dot product between an input sample $\boldsymbol{x}$ and $\boldsymbol{w}$ establishes the side of the hyperplane where $\boldsymbol{x}$ is placed. An important advantage of the linear SVM is that the classifier can be evaluated very efficiently at test time.

Non-linear SVM: Non-linear SVM analyses the input sample on a space of highest dimension by using a kernel k. Equation (4) becomes

$$f(\boldsymbol{x}) = \sum_{i=1}^{N_{\text{SV}}} y_i\alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b \tag{5}$$

where the sign of $f(\boldsymbol{x})$ classifies the input sample. In our study, the kernel is the radial basis function (RBF)

$$k(\boldsymbol{x}_i, \boldsymbol{x}) = \mathrm{e}^{\gamma||x - x_i||^2}, \ \gamma > 0 \tag{6}$$

Non-linear kernels evaluate the input sample against all the support vectors, as shown in (5), improving the performance but increasing the computation time. In our experiments, the total number of support vectors, on average, is 4260.

Boosting classifier: Boosted classifiers are trained by using Real Adaboost algorithm [27]. They are called strong classifiers because they are the lineal combination of T simple classification function $g \in \Re$ known as 'weak' functions. OpenCV uses 'stumps' for classification functions $g(x)$. Let x be an input sample, the strong classifier $G(x)$ is defined as

$$G(x) = \sum_{t=1}^{T} g_t(x) \tag{7}$$

Input x is evaluated by considering the sign of $G(x)$. The optimal value for T founded in training was 778.

Neural network classifier: We choose a multi-layer perceptron (MLP) architecture of three layers, with $29 \times 4 \times \eta$ inputs, one output neuron and a number of hidden neurons fixed on the training phase (the best results where obtained with 24 hidden neurons with $\eta = 8$). All the neurons are activated by the symmetrical sigmoid function

doi: 10.1049/iet-ipr.2013.0496

$$f(x) = \beta * \frac{\left(1 - \mathrm{e}^{-\alpha x}\right)}{\left(1 + \mathrm{e}^{-\alpha x}\right)} \quad (8)$$

with $\beta = 1$ and $\alpha = 1$.

2.3.3 Scale specialised classifiers: The appearance of vehicles changes drastically when they are far away from the camera. Therefore we split the dataset to train two different classifiers. Those samples for which the RoI has a size between 12×12 pixels (the smallest tested RoI) and 36×36 pixels are part of the minimum size base. They will train the first classifier Clf_{12}. Other samples bigger than 36×36 pixel size train the other classifier Clf_{36}. Once the RoIs are generated in the HG step, they are assessed by either the Clf_{12} or the Clf_{36} depending on their sizes.

3 Experiments

3.1 Datasets

Video sequences were recorded by a Vivotek IP SD7151 camera, filming an intersection in Tandil (Argentina). The recording format is MJPEG, and we recorded two resolutions: 320×240 pixels and 640×480 pixels, with the minimum JPEG compression. These choices reduce the capturing process to 1–3 fps for the former, and 0.3 fps for the latter.

Fig. 3 shows captures having strong lateral shadows and rain. These are difficult images because of their drastic changes in the scene and thus vehicles appearance. Lateral shadows hide the vehicles, especially those which are far away from the corner. Other environmental conditions such as a cloudy view (see Fig. 1*a*) are considered to be non-difficult.

Positive samples are picked from the training sequences, see Table 1. Each classifier is trained by using 75% of positive samples randomly chosen. The remaining 25% compound the validation dataset used to optimise the classifiers parameters, for example, the number of hidden neurons for the MLP and the number of weak classifiers for the Boosted classifier.

Negative samples (images without vehicles) are picked from two sources: the training base and the VOC-2012 dataset composed of 10 046 images. For the first round of the bootstrap approach, one negative sample is randomly picked from each capture or image. In the second round, one false alarm obtained with the first trained classifier is added to the negative training dataset.

3.2 Evaluation

The HG output is a set of bounding boxes $B = \{B_d(1), B_d(2), ..., B_d(i)\}$. In addition, classifiers in the HV step obtain a score s_i, for each $B_d(i)$. To evaluate the performance, this set is compared against the vehicle real bounding boxes B_{gt} named as ground-true. The overlapping criterion is the same that is proposed in Challenge Pascal [28]. If a bounding box B_d exceeds the overlap factor over a B_{gt}, it is considered as a correct detection, or a false positive otherwise. If there exists more than one bounding box overlapping the same B_{gt}, only those B_d with the highest overlapping criterion remain, and the others are considered as false positives.

To compare the performance of different classifiers we will use the false positive per image (FPPI) rate. To draw the FPPI curve, I will applied thresholds of increasing values on the set B. Validated bounding boxes are filtered by the non-maximal suppression (NMS) algorithm [25]. Then, the overall miss rate and false positive rate of the test sequences are obtained. Each threshold value thus generates a point in the FPPI curve. The FPPI curves for each classifier are the average obtained by the 3-fold training.

3.3 Parameters selection

Fig. 5 depicts the performance of the HG step using different parameters in a log–log scale of the FPPI against the miss rate. The HG step should have the lowest miss rate possible, because the vehicles missed in this step are not recovered again.

The parameters evaluated in the experiments are

- N is the number of equally spaced thresholds applied to the input image.
- δ is the threshold applied to $S(\boldsymbol{p})$ to preserve meaningful level lines. The different values employed in Fig. 1 are: $\delta = 1$ (°), $\delta = 2$ (Δ) and $\delta = 3$ (*).
- η is the quantised orientation of the level lines.

All those parameters are closely related. Higher N and lower δ generate a great number of level lines capturing a smooth intensity transition between the vehicles and the road, but increasing the noise, as shown in Fig. 3. Both Figs. 5*a* and *b* show that decreasing δ for the same N reduces the miss rate whereas it increases the false positives.

Fig. 5*c* shows the rbfSVM classifier $\mathrm{Clf}_{36}^{\mathrm{RBF}}$'s performance on the 320 pixels width dataset for an MFS calculated with $N = 80$, $\delta = 1$ and $\eta = \{4, 8\}$. The miss rate at 10^{-1} FPPI is

Table 1 List of recorded sequences with their description

Name	Frames	Circulating vehicles		Lateral shadows	Rain	Clouds
		Min size	Max size			
$\mathrm{SeqTrain320}_0$	1193	1104	819	—	—	X
$\mathrm{SeqTrain320}_1$	3671	2463	1745	X	—	X
$\mathrm{SeqTrain640}_2$	454	57	394	X	—	—
$\mathrm{SeqTrain640}_3$	522	63	302	X	—	—
$\mathrm{SeqTrain640}_4$	1239	388	840	—	—	X
$\mathrm{SeqTrain640}_5$	3067	1165	3462	X	—	X
$\mathrm{SeqTest320}_0$	3848	2840	2037	—	X	X
$\mathrm{SeqTest320}_1$	2139	1859	795	X	—	—
$\mathrm{SeqTest640}_2$	1433	289	1864	X	—	—
$\mathrm{SeqTest640}_3$	1432	442	2284	X	—	—
$\mathrm{SeqTest640}_4$	1326	320	1544	X	—	X

doi: 10.1049/iet-ipr.2013.0496

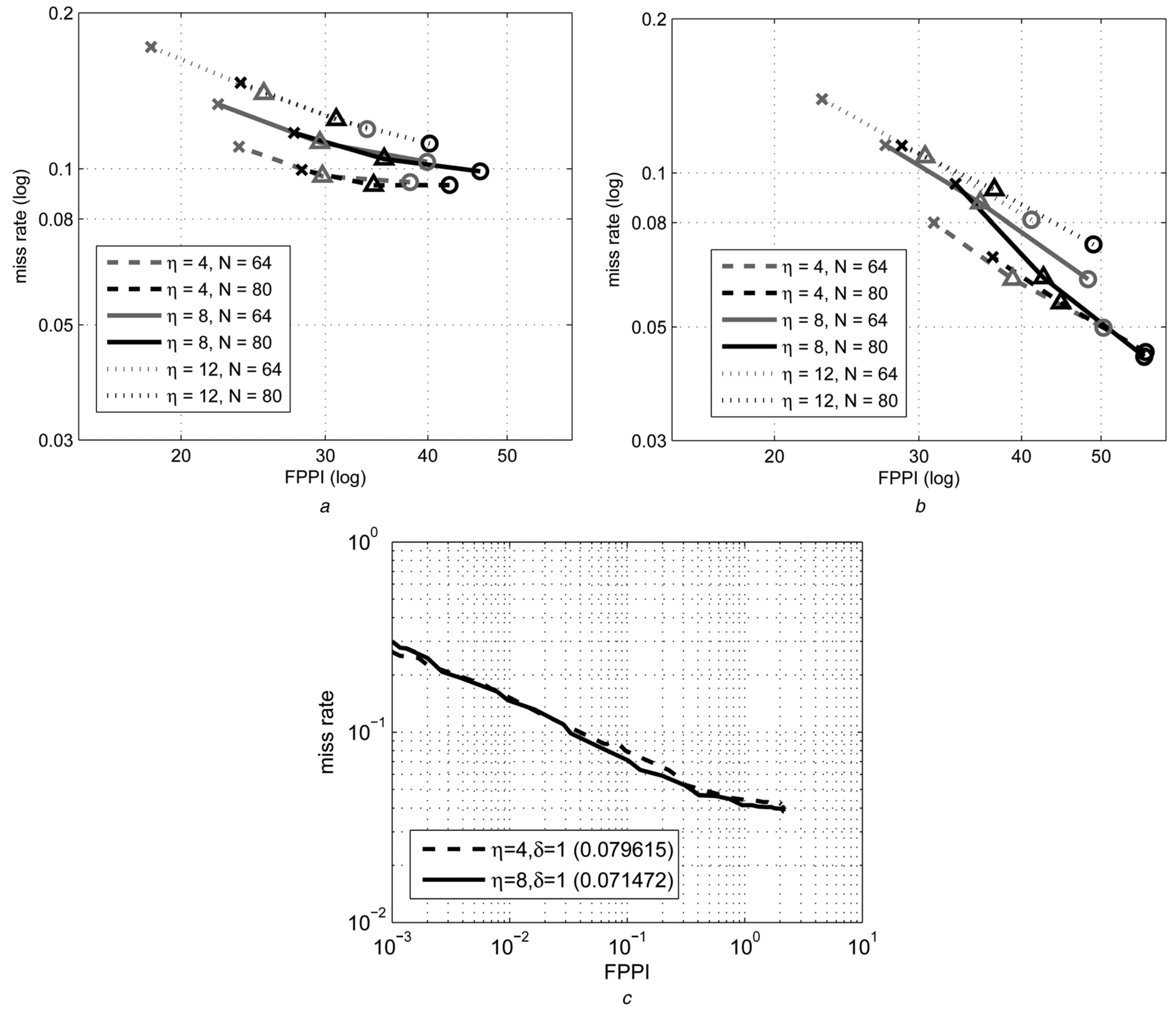


Fig. 5 *(a) and (b) show the average performance of the HG step using different parameters in the MFS generation. (c) HV average performance using rbfSVM classifiers with N = 80 and varying the number of orientations η*

a seq320
b seq640
c $\mathrm{Clf}_{12}^{\mathrm{RBF}}$ @ seq320

shown between the parentheses. It can be seen that the MFSs calculated with $\eta = 4$, have low miss rates in the HG step (vehicles are rigid and rectangular structures), but has a lower performance in the HV step than $\eta = 8$. Then, a higher number of orientations is a rich source of information for the classifiers and helps to discriminate in a better manner vehicle class from non-vehicle samples.

Subsequent classifiers would be trained and tested with an MFS generated with $N = 80$, $\delta = 1$ and $\eta = 8$.

3.4 Processing time

The system runs on an Intel Core i5 CPU @ 2.67 MHz. The program is coded in C++ by using OpenCV version 2.4, but there are some tasks performed in MATLAB, depicted by a (*) in Table 2.

Table 2 presents the processing times in the calculation of the MFS and the HG step. MFS processing time is fixed by the resolution and does not depend on the scene contents. However, the number of RoIs obtained in the HG step is related by the number of vehicles, for example, 20 RoIs can be generated by two or three vehicles, and 100 RoIs by a great number of them (more than eight).

Table 3 shows the processing time employed by the classifiers evaluating different number of RoIs. Clearly, linSVM is the fastest classifier (by several orders) as it is shown in the table.

Table 2 MFS and HG step processing time in milliseconds

Q2

Sequence	MFS	Integral histogram	Generated RoIs(*)	
			100	20
320 × 240	89	3	112	68
640 × 480	340	12	506	343

Table 3 HV processing time in milliseconds

RoIs	Features calculation	rbfSVM	linSVM	MLP	Boost
100	1.17	794	0.13	9.14	3.69
20	0.25	159	0.03	1.95	0.69

IET Image Process., pp. 1–11
doi: 10.1049/iet-ipr.2013.0496

Maximum processing time expected to evaluate a 320×240 pixels capture by using the rbfSVM classifiers is 1 fps. If the classifier is the MLP, the sequence can be evaluated at 5 fps, which is more adapted to an online application. Meanwhile, maximum processing time for the 640×480 resolution is 0.6 or 1.2 fps depending on the classifier.

4 Results

Fig. 6 illustrates the average performance of the four classifiers over the set of bounding boxes B generated in the HG step. It plots miss rate against FPPI in log–log scale (lower curves indicate better performance). Miss rate at 10^{-1} FPPI is a common reference, shown between the parentheses. This figure also plots the precision–recall curves and the average precision value (AP) at 10^{-1} FPPI between the parentheses, which are widely used to compare detectors performance [28]. Classifier rbfSVM outperforms all the other classifiers by 3% of miss rate at 10^{-1} FPPI, having on average a miss rate of 13.3% for the Clf_{36}. The MLP classifier shows a better performance than the Boosted classifier if we compare the miss rate and the APs values. The linear SVM classifier has the worst performance in classifying low scale samples (see Fig. 6*a*).

As expected, the detection rate of the system drops drastically with minimum-size vehicles that are partially occluded. The first reason for this is that they have less resolution and thus, fewer details. Second, many of those vehicles are partially occluded in the queue. In addition, strong cast shadows hide these vehicles eliminating intensity transition. In the literature, Buch *et al.* [10] also address this problem that hinders performance.

Fig. 7 presents the FPPI performance of the rbfSVM and the MLP classifiers on each test sequence. The best performances of the rbfSVM classifiers were obtained in $\mathrm{SeqTest320}_1$ with a miss rate of only 4.4% at 10^{-1} FPPI. This sequence is considered as non-difficult because it was captured during a cloudy day.

As the results show, the rain does not affect vehicle detection as the cast shadow does. If we analyse Figs. 3*a*, *c* and *e*, farthest vehicles in the queue do not generate any intensity transition.

Besides, cast shadows are part of the background reference as horizontal segments. There exists the possibility that

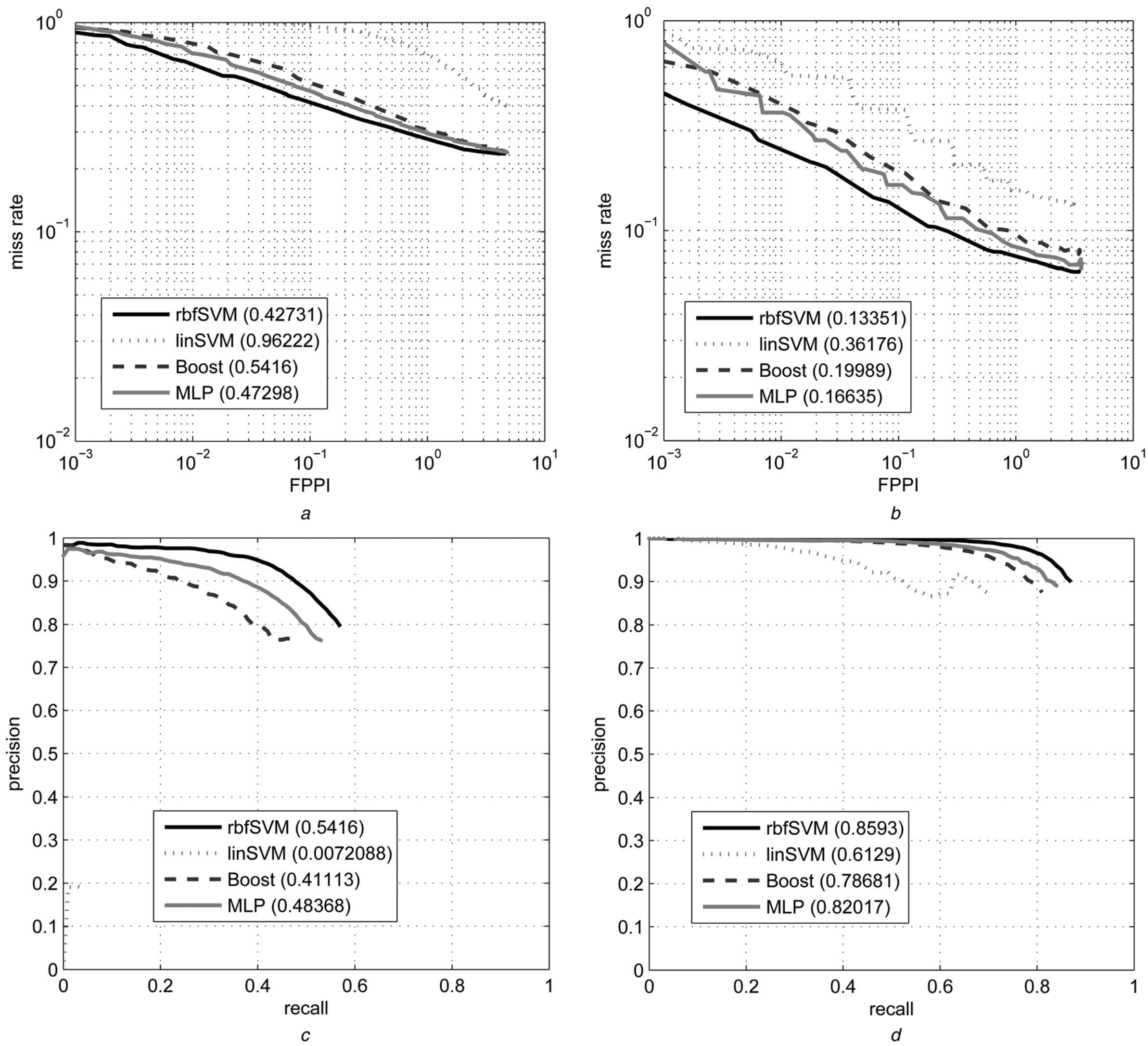


Fig. 6 *Average performance of the detection system using different classifiers in the HV step*
a Clf_{12}
b Clf_{36}
c Clf_{12}
d Clf_{36}

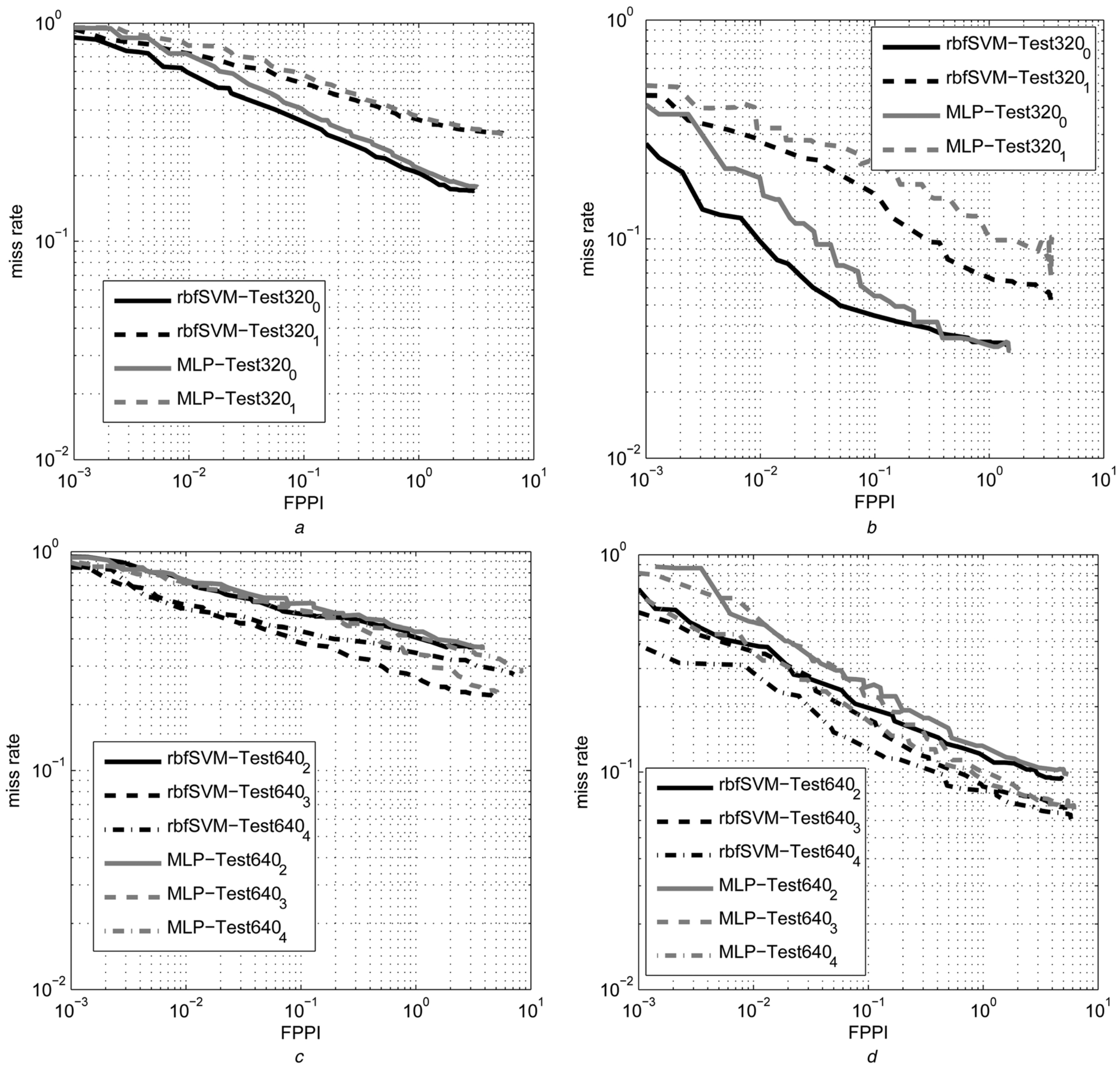


Fig. 7 *FPPI performance of rbfSVM and MLP classifiers on each test sequence*
a Clf_{12}
b Clf_{36}
c Clf_{12}
d Clf_{36}

horizontal vehicle level lines coincide with those reference segments. In that case, these vehicle level lines are not part of the MFS. This situation can rarely happen, but when we work on sequences of 320 pixels width the probability is greater. A solution to overcome this drawback is to generate the MFS on a colour space. Then, the colour transition between the vehicle and the road should be different to the transition of the cast shadow.

5 Conclusions

This paper presents a pattern recognition framework that estimates the number of vehicles passing through an intersection. The main advantages of this system working with the MFS include an increase in robustness and minimised loss of information.

The simple vehicle model uses horizontal segments and corners obtained from the MFS. It not only overcomes the occlusion problem by searching for a windshield configuration, but also generates fast vehicle hypotheses.

Furthermore, computation time efficiency is obtained by grouping the MFS information in HO2 L. Their performance on vehicle detection were evaluated by four different classifiers: linear SVM, non-linear SVM, neural networks and boosting. Non-linear SVM outperforms the other classifiers, followed by the neural network classifier. The proposed system obtains excellent results in highly occluded sequences with queued vehicles, reaching on average, a miss rate of 13% at 10^{-1} FPPI.

Two sequences resolutions were evaluated: 320×240 and 640×480 pixels size. Increasing the image resolution, which implies more processing time, did not provide better results. The system performance is in fact, closely related to the illumination and weather conditions of the sequence, for example, strong cast shadows represent the worst situation for system hiding vehicles which are not detected on the MFS.

It was proved that the framework can realise online vehicle detection at 5 fps for 320×240 image size by using the MLP classifier obtaining acceptable performance and should be suitable for embedded implementations on the traffic light.

IET Image Process., pp. 1–11
doi: 10.1049/iet-ipr.2013.0496

Further work can be conducted on the HG step, for example, incorporating a cascade of boosted classifiers employing the MFS [29]. The cascade can be prepared to eliminate a greater number of false alarms than the HG model-based methodology. However, the implementation increases the system complexity considerably. In pedestrian detection it is justified because of the nature of the person class, and the elaboration of an a priori model is very difficult.

6 Acknowledgments

This work was supported by PICT Bicentenario-2283 (ANPCyT), ACyT R12T03 (UADE) and CONICET.

7 References

1 Klein, L.A., Mills, M.K., Gipson, D.R.P.: 'Traffic detector handbook: third edition', Technical Report, 2006, Vol. I & II
2 Zanin, M., Messelodi, S., Modena, C.M.: 'An efficient vehicle queue detection system based on image processing'. Proc. Int. Conf. of Image Analysis and Applications, 2003, pp. 232–237
3 Quinn, J.A., Nakibuule, R.: 'Traffic flow monitoring in crowded cities'. Spring Symp. on Artificial Intelligence for Development, Stanford, 2010
4 Fathy, M., Siyal, M.Y.: 'Real-time image processing approach to measure traffic queue parameters', *IEE Proc. Vis. Image Signal Process.*, 1995, **142**, (5), pp. 297–303
5 Higashikubo, M., Hinenoya, T., Takeuchi, K.: 'Traffic queue length measurement using an image processing sensor', *Sumitomo Electr. Techn. Rev.*, 1997, **43**, (1), pp. 64–68
6 Aubert, D., Boillot, F.: 'Automatic measurement of traffic variables by image processing application to urban traffic control', *Rech. Transp. Securite*, 1999, **62**, pp. 7–21
7 Yang, J., Wang, Y., Sowmya, A., Li, Z.: 'Vehicle detection and tracking with low-angle cameras'. Proc. ICIP, Hong Kong, September 2010, pp. 685–688
8 Pang, C.C., Lam, W.W., Yung, N.H.: 'A method for vehicle count in the presence of multiple-vehicle occlusions in traffic images', *Int. Transp. Syst.*, 2007, **8**, (3), pp. 441–459
9 Morris, B., Trivedi, M.: 'Learning, modeling and classification of vehicle track patterns from live video', *IEEE Trans. Intell. Transp. Syst.*, 2008, **9**, (3), pp. 425–437
10 Buch, N., Orwell, J., Velastin, S.A.: 'Urban road user detection and classification using 3D wire frame models', *IET Comput. Vis.*, 2010, **4**, (2), pp. 105–116
11 Caselles, V., Coll, B., Morel, J.M.: 'Topographic maps and local contrast changes in natural images', *Int. J. Comput. Vis.*, 1999, **33**, (1), pp. 5–27
12 Aubert, D., Guichard, F., Bouchafa, S.: 'Time-scale change detection applied to real-time abnormal stationarity monitoring', *Real-Time Imaging*, 2004, **10**, (1), pp. 9–22
13 Dalal, N., Triggs, B.: 'Histograms of oriented gradients for human detection'. Proc. CVPR, California, USA, June 2005, pp. 886–893
14 Porikli, F.: 'Integral histogram: a fast way to extract histograms in cartesian spaces'. Proc. CVPR, California, USA, June 2005, pp. 829–836
15 Bouchafa, S.: 'Motion detection invariant to contrast changes. Application to detection abnormal motion in subway corridors', PhD thesis, UPMC Paris VI, 1998
16 Cao, F., Musse, P., Sur, F.: 'Extracting meaningful curves from images', *J. Math. Imaging Vis.*, 2005, **22**, pp. 159–181
17 Sun, Z., Bebis, G., Miller, R.: 'On-road vehicle detection using evolutionary Gabor filter optimization', *IEEE Trans. Intell. Transp. Syst.*, 2005, **6**, (2), pp. 125–137
18 Sotelo, M.A., Garcia, M.A., Flores, R.: 'Vision based intelligent system for autonomous and assisted downtown driving', *Int. Workshop on Comput. Aided Syst. Theory*, 2003, **2809**, pp. 326–336
19 Bertozzi, M., Broggi, A., Castelluccio, S.: 'A real-time oriented system for vehicle detection', *J. Syst. Archit.*, 1997, **43**, pp. 317–325
20 Srinivasa, N.: 'Vision-based vehicle detection and tracking method for forward collision warning in automobiles', Proc. Intelligent Vehicle Symp., Versailles, France, June 2002, vol. 2, pp. 626–631
21 Collado, J.M., Hilario, C., de la Escalera, A., Armingol, J.M.: 'Model based vehicle detection for intelligent vehicles'. Intelligent Vehicle Symp., June 2004, pp. 572–577
22 Negri, P., Clady, X., Hanif, S.M., Prevost, L.: 'A cascade of boosted generative and discriminative classifiers for vehicle detection', *EURASIP J. Adv. Signal Process.*, 2008, pp. 1–12 **Q3**
23 Achard, C., Bigorgne, E., Devars, J.: 'A sub-pixel and multispectral corner detector'. Proc. ICPR, Barcelona, Spain, September 2000, vol. 6, pp. 659–962
24 http://www.opencv.willowgarage.com/wiki, version 2.4.2, accessed on March 2013
25 Felzenszwalb, P., Girshick, G., McAllester, D., Ramanan, D.: 'Object detection with discriminatively trained part-based models', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2010, **32**, pp. 1627–1645
26 Vapnik, V.: 'The nature of statistical learning theory' (Springer, NY, 1995)
27 Schapire, R., Singer, Y.: 'Improved boosting algorithms using confidence-rated predictions', *Mach. Learn.*, 1999, **37**, (3), pp. 297–336
28 Everingham, M., Gool, L., Williams, C.K., Winn, J., Zisserman, A.: 'The PASCAL visual object classes (VOC) challenge', *Int. J. Comput. Vis.*, 2010, **8**, (2), pp. 303–338
29 Negri, P., Lotito, P.: 'Pedestrian detection using a feature space based on colored level lines'. Proc. CIARP, Buenos Aires, Argentine, September 2012, pp. 885–892

IPR20130496

Author Queries

Pablo Negri

Q1 Please expand R-HOG.

Q2 Please provide significance of '*' in Table 2.

Q3 Please provide page number for reference [22]