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

Pablo Negri
CONICET, Av. Rivadavia 1917, Buenos Aires, Argentine
Instituto de Tecnologia, UADE, Lima 717, Buenos Aires, Argentine
pnegri@uade.edu.ar

Abstract

This paper 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 using level lines and generates a new feature space called Movement Feature Space (MFS). The MFS generates primitives as segments and corners to match a vehicle model generating hypotheses. The MFS is also grouped in a histogram configuration called Histograms of Oriented Level Lines (HO2L). This work uses HO2L features to validate vehicle hypotheses comparing the performance of different classifiers: linear SVM, non-linear SVM, neural networks and boosting. On average, successful detection rate is of 86 percent with 10^{-1} FPPI 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. A 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 neighborhood, measuring and operating with *intelligent traffic lights*. Its main objective is to adapt the computation of adequate green times to variations in traffic load. In order 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 state is estimated by measuring at specific intersections, and the load of other intersections is provided by simulation. With this information, the system defines all the green times to maximize 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, 5, 6, 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 using three basic methodologies: time differences computed between consecutive frames at times t and $t+\alpha$, background subtraction using an image of the scene without vehicles, and edge detection based on variation in brightness.

Fathy [4] combines 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 neighborhood criterion in regions or blobs creating a binary map, and identifying moving objects. The presence of vehicles is confirmed by edge detection.

Zanin [2] uses 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 [10] uses the motion silhouettes and a 3D model to detect and classify vehicles. In the work of Pang [8] the boundaries of the motion blobs are analyzed 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 [7] also tackles vehicle occlusions proposing a windshield-based vehicle detection algorithm. They generate hypotheses 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.

(a) Original Image

(b) Motion Detection

(c) Hypotheses generation

(d) Hypotheses validation

(e) Final results

Figure 1: Overall sequence of the vehicle detection algorithm.

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 the blowing wind or traffic vibrations. In addition, the desired response time for an on-line

application should be between 1-5 fps.

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

In the first stage, the motion detection uses a level line based approach [11, 12], illustrated in figure 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, e.g. illumination changes, weather conditions, etc. In such situations, the appearance of vehicles on the MFS does not change significantly compared to 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 shows fig. 1(c). The algorithm employs the information from the MFS and a vehicle model based on segments and corners.

The information inside each RoI is encoded using a family of Histogram of Oriented Level Lines (HO2L) descriptors calculated on the MFS, and grouped in a configuration based on the R-HOG [13]. The HO2L features obtained from the MFS allow a multi-scale vehicle detection, avoiding the construction of a dense pyramid of sub-sampled versions of the input image that is very costly in terms of calculation time. They can also be computed very quickly using an integral histogram [14].

Third stage of the system performs RoIs validation using a classifier discriminating vehicle and non-vehicle classes. This paper explores four different classifiers evaluating the performance of the HO2L feature space in classification and processing time. Validated RoIs, as shown on fig. 1(d), are finally grouped using Non-Maximal Suppression algorithm. Those RoIs are considered the system outputs (see figure 1(e)).

The structure of the paper is as follows: section 2 details the methodology to obtain the MFS, develops the hypothesis generation algorithm 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.

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 models an image reference 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 and Aubert [15, 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 $\mathbf{p} \in I$, so that their intensity is greater than or equal to λ,

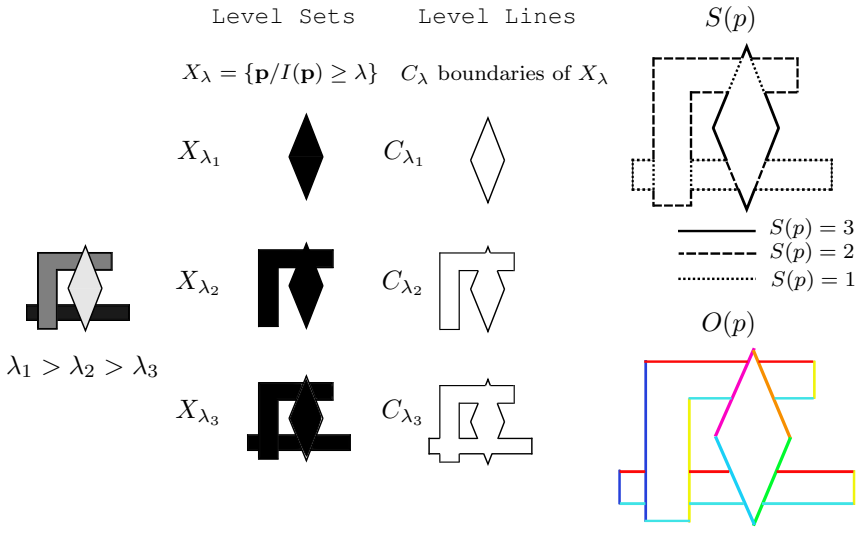
$$X_\lambda = \{\mathbf{p} / I(\mathbf{p}) \geq \lambda\}.$$

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 $\mathcal{C}$ of the image I obtained from a given set of N equally spaced thresholds $\Lambda = \{\lambda_1, ..., \lambda_N\}$. 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 using a derivative filter of 5×5 pixels (orientations are quantized 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.

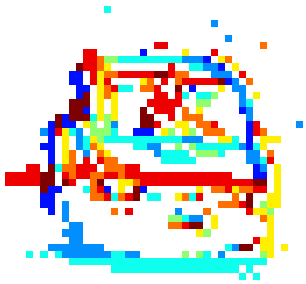
Generally, in the practical implementation, only those pixels for which $S(\mathbf{p})$ is greater than a fixed threshold δ are considered, simplifying the analysis and preserving meaningful contours.



(a) Level lines extraction

(b) Vehicle image

(c) S_t values



(d) O_t values

Figure 2: Level lines calculation of a vehicle image sample.

Figure 2(a) shows the level lines extraction from a simple geometric configuration. It also has two arrays, $S(p)$ with the number of the superimposed level lines, and $O(p)$, for which each color represents a gradient orientation. Figure 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 and Aubert [15, 12] defined an adaptive background reference model, composed of the set of pixel p which are stable over an 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:

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

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

- $p \notin R_t$,
- $p \in R_t \wedge O_t(p) \neq O^R_{t-1}(p)$ (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 due to noise or other perturbations (see [15, 12] for details).

Figure 3 shows two examples of the adaptive reference model. The first row shows the original capture, while the second one illustrates the reference model. Last row presents the detected set D_t with a gray level corresponding to the value of S_t. Note that for fig. 3(a), 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

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

Here, it is considered an *a priori* model of a vehicle, inspired in the configuration proposed by [21], and depicted in fig. 4(a). It is composed by 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 labeled data set.

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 others motion detection algorithms as blobs [8], they do not provide internal information as the MFS, e.g. segment h_2 and corners e_3 and e_4 in the model.

2.2.1 Hypothesis generation using segments

The first configuration analyzed 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 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)).

Another RoIs can be generated using the others 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. First RoI is created considering the segment as h_2 and the RoI is placed taking the segment as the middle position. The second RoI is generated considering the segment as h_3, and the RoI is placed with the segment as the upper limit.

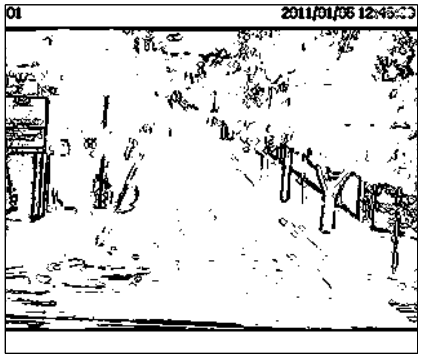
2.2.2 Hypothesis generation using corners

Figure 3(b) shows that occlusions for queued vehicles can be severe when sequences are captured using low angle cameras. Yang [7] proposed a windshield identification procedure to minimize 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 figure 4(a)) to generate additional

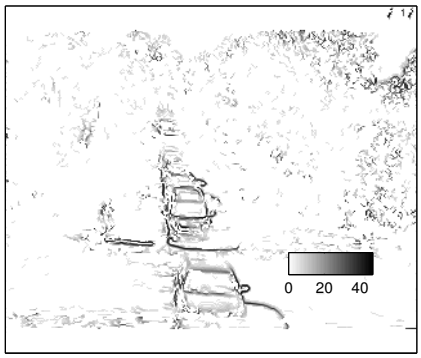
(a) Original

(b) Original

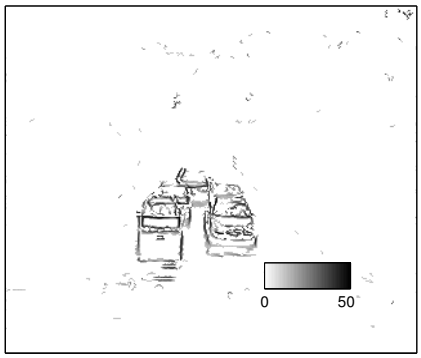
(c) Reference (δ=1)



(d) Reference (δ=3)

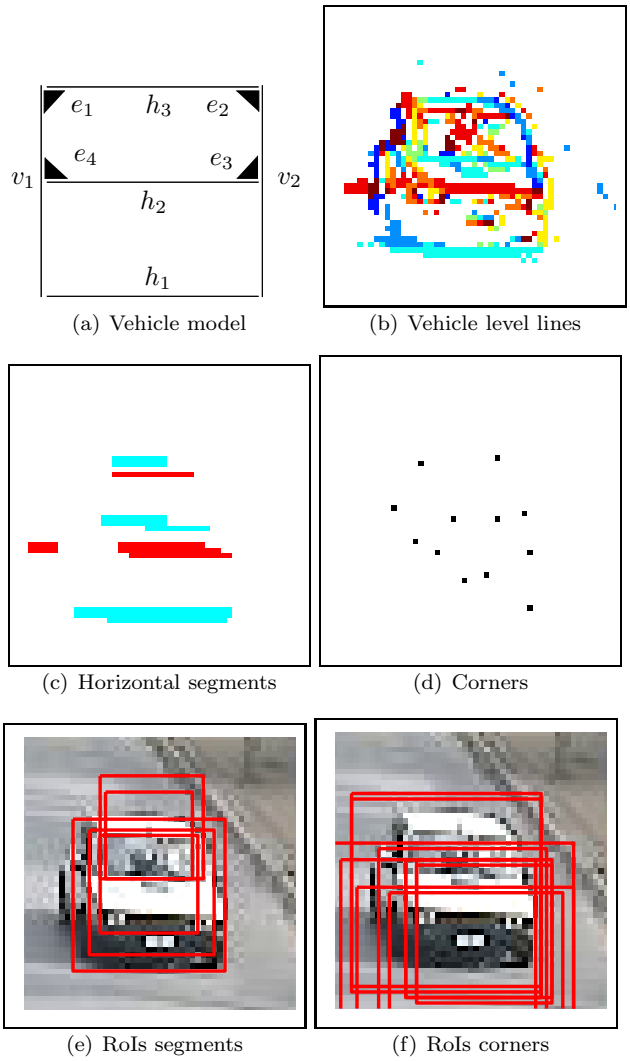


(e) Movement detection (δ=1)



(f) Movement detection (δ=3)

Figure 3: Background model reference and movement detection, with $N=80$, $\eta=8$, and $\delta=1$ for (c)-(d), and $\delta=3$ for (d)-(f).



(a) Vehicle model (b) Vehicle level lines

(c) Horizontal segments (d) Corners

(e) RoIs segments (f) RoIs corners

Figure 4: Vehicle model and RoI generation.

RoIs and increase the probabilities of finding queued vehicles. Corner detection is carried out 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 neighborhood V_p of a corner p, the average of the cross product between $G(p)$ and all 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 neighborhood V_p can be computed as:

$$K = I_x^2 < I_y^2 > + I_y^2 < I_x^2 > -2 I_x I_y < I_x I_y >$$

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

$$I_y(p) = S(p) \sin(O(p)), \quad (2)$$

$$I_x(p) = S(p) \cos(O(p)). \quad (3)$$

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

To generate the RoIs corresponding to 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, a RoI is created by the three corners in the configuration of our vehicle model. Figure 4(f) shows the RoIs generated from these primitives.

2.3 Hypotheses Validation using a classifier

As shown in figure 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 using the MFS. It results in a concatenated set of Histograms of Oriented Level Lines (HO2L), which is computed in a configuration similar to the R-HOG proposed by Dalal [13].

The square RoI is subdivided into two grids of 6x6 and 3x3 non overlapped cells. Within each cell r_i, the MFS^{HO2L} descriptor is the histogram $\mathbf{h}$ having η bins, one for each orientation. For each bin o of $\mathbf{h}$, we add all the $S_t(p)$ values for the p with this orientation, $\mathbf{h}(o) = \{\sum_{p \in r_j} S_t(p)/O(p) = o\}$.

A grid of 2x2 continuous cells generates a block histogram of the four concatenated histograms $\mathbf{h}$, having 4η bins in all. The blocks are then normalized using the *L2-Norm*: $\mathbf{v} \rightarrow \mathbf{v}/\sqrt{\|\mathbf{v}\|_2^2 + \epsilon}$.

Thus, each RoI generates 29 blocks of MFS^{HO2L} concatenated descriptors. The feature vector has 29x4xη 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 HO2L feature space for vehicle detection. It is employed the OpenCV implementation of each classifiers [24], and a two rounds bootstrapping approach [25] is made in place in the learning phase.

Linear 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 maximizes the separation margin separating the training data on the feature space (vehicles versus non-vehicle classes). Let be $\{\mathbf{x}_i, y_i\}$ a training dataset, where $y_i \in \{-1, +1\}$, $\mathbf{x}_i \in \Re^d$. Classification is formulated as:

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

where $\mathbf{w}$ is the normal to the hyperplane. The $\mathbf{x}_i$ at which the equation 4 equals zero are called support vectors and define two parallel planes on both sides of the hyperplane separated by a margin $2/||\mathbf{w}||$. After the SVM training, the $\mathbf{w}$ is calculated from the equation 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 $\mathbf{x}$ and $\mathbf{w}$ establishes the side of the hyperplane where $\mathbf{x}$ 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 analize the input sample on a space of highest dimension using a kernel k. Equation 4 becomes

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

where the sign of $f(\mathbf{x})$ classifies the input sample. In our study, the kernel is the Radial Basis Function (RBF)

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

Non-linear kernels evaluate the input sample against all the support vectors, as shows equation 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 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 be x 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 29x4xη inputs, one output neuron and a number of hidden neurons fixed on the training phase (best results where obtained with 24 hidden neurons with η=8). All the neurons are activated by the symmetrical sigmoid function:

$$f(x) = \beta * \frac{(1 - e^{-\alpha x})}{(1 + e^{-\alpha x})} \tag{8}$$

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

2.3.3 Scale specialized 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 12x12 pixels (the smallest tested RoI) and 36x36 pixels are part of the minimum size base. They will train the first classifier Clf_{12}. Others samples bigger than 36x36 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 Data Sets

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: 320x240 pixels and 640x480 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 later.

Figure 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, specially those which are far away from the corner. Others environmental conditions, as a cloudy view (see fig. 1(a)) are considered to be non-difficult.

Positive samples are picked from training sequences, see table 1. Each classifier is trained using the 75% of positives samples randomly chosen. The remaining 25% compound the validation dataset used to optimize classifiers parameters, e.g. 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

Name	Frames	Circulating vehicles		Lateral Shadows	Rain	Clouds
		Min Size	Max Size			
$SeqTrain320_0$	1193	1104	819	-	-	X
$SeqTrain320_1$	3671	2463	1745	X	-	X
$SeqTrain640_2$	454	57	394	X	-	-
$SeqTrain640_3$	522	63	302	X	-	-
$SeqTrain640_4$	1239	388	840	-	-	X
$SeqTrain640_5$	3067	1165	3462	X	-	X
$SeqTest320_0$	3848	2840	2037	-	X	X
$SeqTest320_1$	2139	1859	795	X	-	-
$SeqTest640_2$	1433	289	1864	X	-	-
$SeqTest640_3$	1432	442	2284	X	-	-
$SeqTest640_4$	1326	320	1544	X	-	X

Table 1: List of recorded sequences with their description.

vehicle real bounding boxes B_{gt} named as ground-true. The overlapping criterion is the same 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 exist 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 be applied thresholds of increasing values on the set B. Validated bounding boxes are filtered by the non maximal suppression algorithm (NMS) [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

Figure 5 depicts the performance of the HG step using different parameters in a *log-log* scale of the FPPI versus the miss rate. The HG step should has the lowest miss rate possible, because vehicles missed in this step are not recovered again.

The parameters evaluated in the experiments are:

- N: the number of equally spaced threshold applied to the input image,
- δ: the threshold applied to $S(\mathbf{p})$ to preserve meaningful level lines. The different values employed in fig. 5 are: δ=*1* ($\circ$), δ=*2* ($\triangle$), and δ=*3* ($*$),
- η: the quantized orientations 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 transitions between the vehicles and the road, but increasing the noise, as shows figure 3. Both 5(a) and 5(b) figures show that decreasing δ for the same N reduces the miss rate while increases false positives.

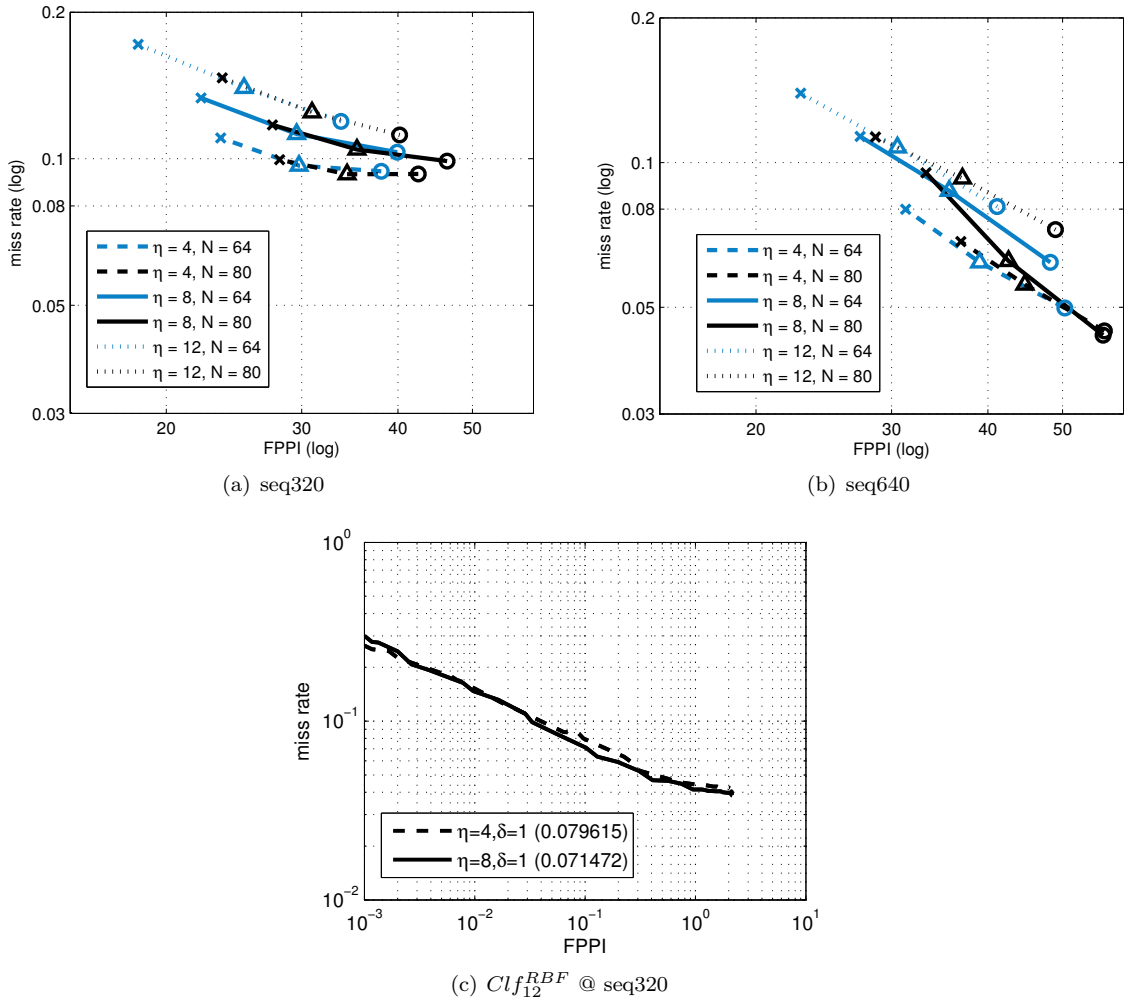
Figure 5(c) shows the rbfSVM classifier Clf_{36}^{RBF} performance on the 320 pixels width dataset for a MFS calculated with N=*80*, δ=*1* and η=$\{$*4,8*$\}$. The miss rate at 10^{-1} FPPI is shown between parentheses. It can be seen that the MFSs calculated with η=*4*, has low miss rate in the HG step (vehicles are rigid and rectangular structures), but has a lower performance in the HV step than η=*8*. Then, a higher number of orientations is a rich source of information for the classifiers and helps to better discriminate vehicle class from non-vehicle samples.

Subsequent classifiers would be trained and tested with a MFS generated with N=*80*, δ=*1* and η=*8*.

3.4 Processing time

The system runs on a Intel Core i5 CPU @ 2.67 Mhz. The program is coded in C++ using the OpenCV version 2.4, but there are some task performed in MATLAB, signed by a (*) in table 2.

Table 2 presents 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



(a) seq320

(b) seq640

(c) Clf_{12}^{RBF} @ seq320

Figure 5: (a)-(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.

obtained in the HG step is related by the number of vehicles, e.g. 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.

Maximum processing time expected to evaluate a 320x240 pixels capture 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 on-line application. Meanwhile, maximum processing time for the 640x480 resolution is 0.6 or 1.2 fps depending the classifier.

4 Results

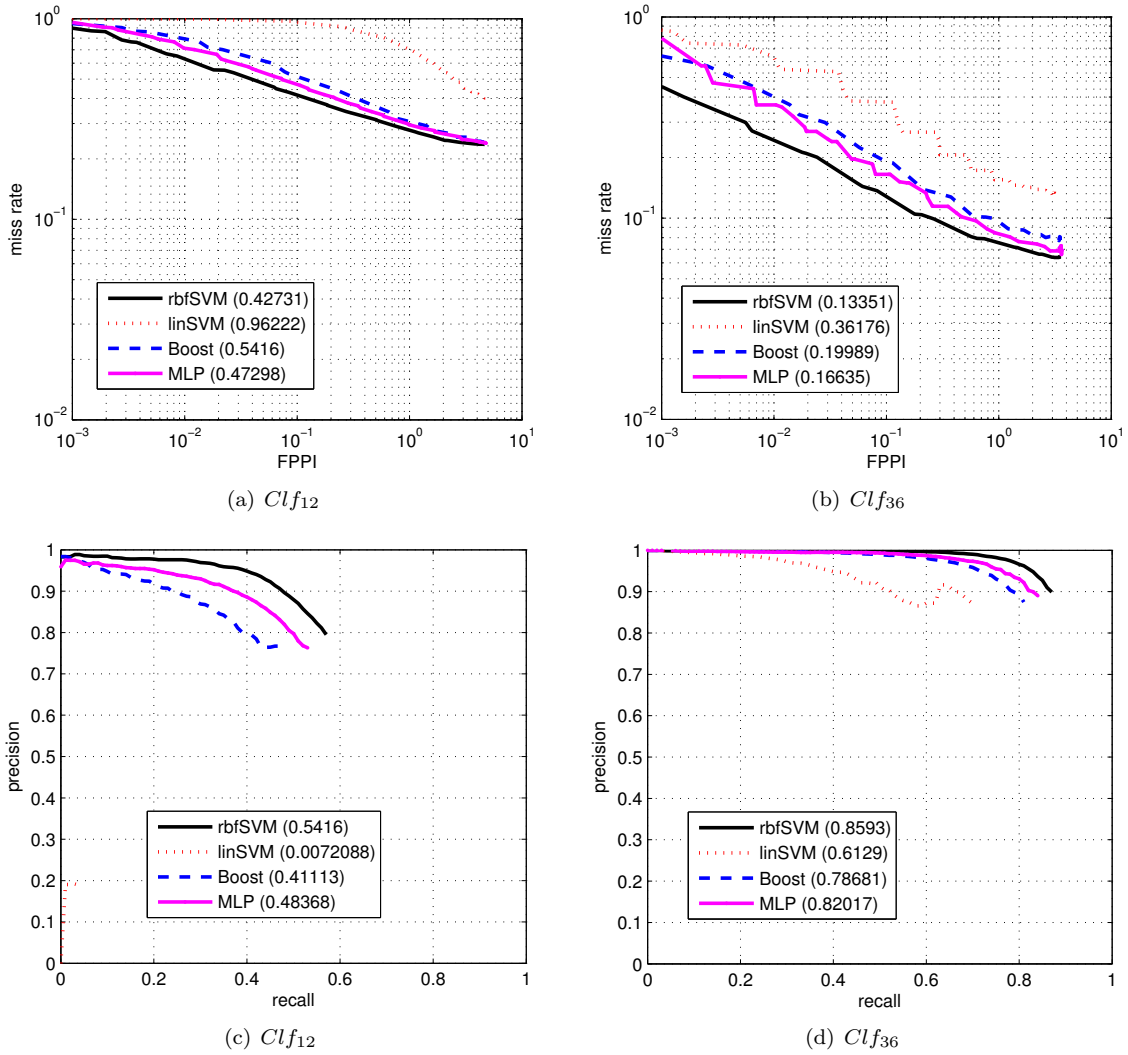
Figure 6 illustrates the average performance of the four classifiers over the set of bounding boxes B generated in the HG step. It plots miss rate versus FPPI in log-log scale (lower curves indicate better performance). Miss rate at 10^{-1} FPPI is a common reference, shown between parentheses. This figure also plots the Precision-Recall curves and the Average Precision value (AP) at 10^{-1} FPPI between 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}. MLP classifier shows a better performance than Boosted classifier if we compare miss rate

Sequence	MFS	Integral histogram	Generated RoIs(*)	
			100	20
320x240	89	3	112	68
640x480	340	12	506	343

Table 2: MFS and HG step 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

Table 3: HV processing time in milliseconds.



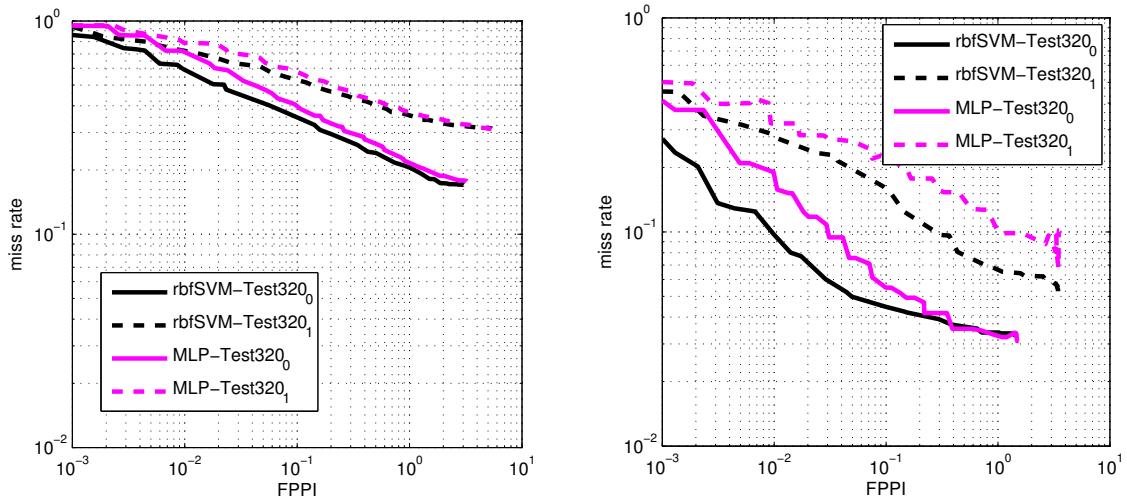
(a) Clf_{12} (b) Clf_{36}

(c) Clf_{12} (d) Clf_{36}

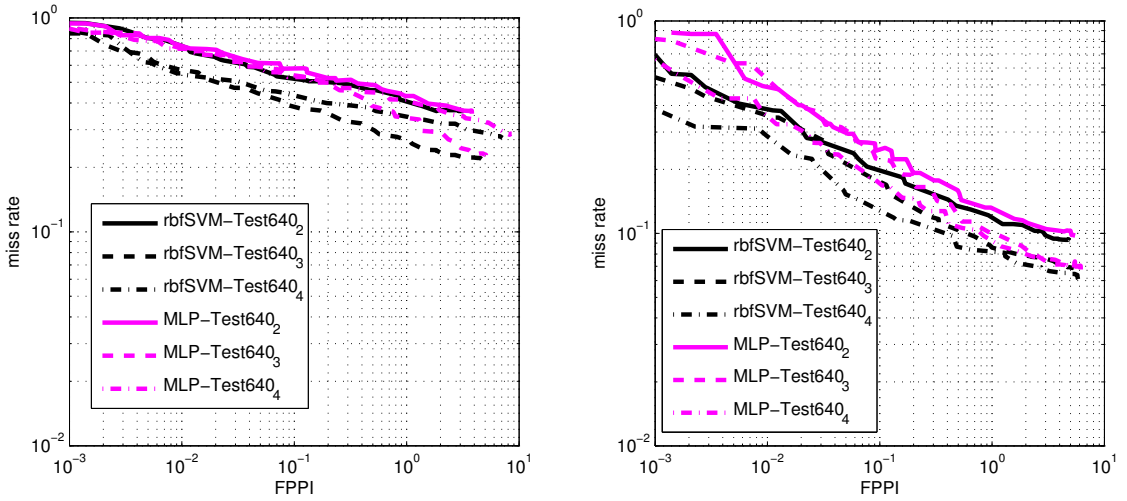
Figure 6: Average performance of the detection system using different classifiers in the HV step.

and APs values. Linear SVM classifier has the worst performance 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. Additionally, strong cast shadows hide these vehicles eliminating intensity transition. In the literature, Butch [10] also addresses this problem that hinders performance.



(a) Clf_{12} (b) Clf_{36}



(c) Clf_{12} (d) Clf_{36}

Figure 7: FPPI performance of rbfSVM and MLP classifiers on each test sequence.

Figure 7 presents the FPPI performance of the rbfSVM and the MLP classifiers on each test sequence. Best performance of the rbfSVM classifiers where obtained in the $SeqTest320_1$ with a miss rate of only 4.4 % at 10^{-1} FPPI. This sequence is considered as non-difficult because was captured during a cloudy day.

As the results show, the rain does not affect the vehicle detection as the cast shadows does. If we analyze figs. 3(a), 3(c) and 3(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 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 happens, 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 color space. Then, the color transition between vehicle and 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 Movement Feature Space (MFS) include an increased in robustness and minimized loss of information.

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 Histograms of Oriented Level Lines (HO2L). Their performance on vehicle detection where evaluated by four different classifiers: linear SVM, non-linear SVM, neural networks and boosting. Non-linear SVM outperforms the others 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: 320x240 and 640x480 pixels size. Increasing the image resolution, which implies more processing time, did not obtained better results. The system performance is in fact, closely related to the illumination and weather conditions of the sequence, e.g. strong cast shadows represent the worst situation for the system hiding vehicles which are not detected on the MFS.

It was proved that the framework can realize on-line vehicle detection at 5 fps for 320x240 image size using the MLP classifier obtaining acceptable performance and should be suitable for embedded implementations on the traffic light.

Further work can be done on the HG step, e.g. 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 considerably the system complexity. 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.

References

[1] Klein, L.A., Mills, M.K., and Gipson, D.R.P.: 'Traffic detector handbook: third edition', Technical Report, 2006, Vol. I & II.

[2] Zanin, M., Messelodi, S., and 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., and Nakibuule, R.: 'Traffic flow monitoring in crowded cities', Spring Symposium on Artificial Intelligence for Development, Stanford, 2010.

[4] Fathy, M., and Siyal, M.Y.: 'Real-time image processing approach to measure traffic queue parameters', IEE Proc. Vision Image and Signal Processing, 1995, 142(5), pp. 297-303.

[5] Higashikubo, M., Hinenoya, T., and Takeuchi, K.: 'Traffic queue length measurement using an image processing sensor', Sumitomo Electric Technical Review, 1997, 43(1), pp. 64-68.

[6] Aubert, D. and Boillot, F.: 'Automatic measurement of traffic variables by image processing application to urban traffic control', Recherche-Transports-Securite, 1999, 62, pp. 7-21.

[7] Yang, J., Wang, Y., Sowmya, A. and 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., and Yung, N.H.: 'A method for vehicle count in the presence of multiple-vehicle occlusions in traffic images', International Transportation Systems, 2007, 8(3), pp. 441-459.

[9] Morris, B., and Trivedi, M.: 'Learning, modeling and classification of vehicle track patterns from live video', IEEE Trans. on Intelligent Transportations Systems, 2008, 9(3), pp. 425-437.

[10] Buch, N., Orwell, J. and Velastin, S.A. 'Urban road user detection and classification using 3D wire frame models', IET Computer Vision, 2010, 4(2), pp. 105-116.

[11] Caselles, V., Coll, B. and Morel, J.M.: 'Topographic maps and local contrast changes in natural images', International Journal on Computer Vision, 1999, 33(1), pp. 5-27.

[12] Aubert, D., Guichard, F. and 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. and 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., and Sur, F.: 'Extracting meaningful curves from images', Journal of Mathematical Imaging and Vision, 2005, 22, pp. 159-181.

[17] Sun, Z., Bebis, G. and Miller, R.: 'On-road vehicle detection using evolutionary Gabor filter optimization', IEEE Trans. on Intelligent Transportation Systems, 2005, 6, (2), pp. 125-137.

[18] Sotelo, M.A., Garcia, M.A., and Flores, R.: 'Vision based intelligent system for autonomous and assisted downtown driving', International Workshop on Computer Aided Systems Theory, 2003, 2809, pp. 326-336.

[19] M. Bertozzi, A. Broggi, and S. Castelluccio: 'A real-time oriented system for vehicle detection', Journal of Systems Architecture, 1997, 43, pp. 317-325.

[20] Srinivasa, N.: 'Vision-based vehicle detection and tracking method for forward collision warning in automobiles', Proc. Intelligent Vehicle Symposium, Versailles, France, June 2002, 2, pp. 626-631.

[21] Collado, J.M., Hilario, C., de la Escalera, A. and Armingol, J.M.: 'Model based vehicle detection for intelligent vehicles', Intelligent Vehicle Symposium, June 2004, pp. 572-577.

[22] Negri, P., Clady, X., Hanif, S.M., and Prevost, L.: 'A cascade of boosted generative and discriminative classifiers for vehicle detection', EURASIP Journal on Advances in Signal Processing, 2008, pp. 1-12.

[23] Achard, C., Bigorgne E., and Devars, J.: 'A sub-pixel and multispectral corner detector', Proc. ICPR, Barcelona, Spain, September 2000, 6, pp. 659-962.

[24] `http://opencv.willowgarage.com/wiki`, version 2.4.2, accessed on March 2013.

[25] Felzenszwalb, P., Girshick, G., McAllester, D. and Ramanan, D.: 'Object detection with discriminatively trained part-based models', IEEE Trans. on Pattern Analysis and Machine Intelligence, 2010, 32, pp. 1627-1645.

[26] Vapnik, V.: 'The nature of Statistical Learning Theory', NY (Ed.), Springer, 1995.

[27] Schapire, R. and Singer, Y.: 'Improved Boosting Algorithms Using Confidence-rated Predictions', Machine Learning, 1999, 37 (3), pp. 297-336.

[28] Everingham, M., Gool, L., Williams, C. K., Winn, J. and Zisserman, A.: 'The PASCAL Visual Object Classes (VOC) Challenge', International Journal on Computer Vision, 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.