Detecting Pedestrians on a Movement Feature Space

Pablo Negri[a,b], Norberto Goussies[a,c], Pablo Lotito[a,d]

[a]CONICET, Av. Rivadavia 1917, Buenos Aires, Argentina
[b]Instituto de Tecnologia, UADE, Lima 717, Buenos Aires, Argentina
[c]DC-UBA, Buenos Aires, Argentina
[d]PLADEMA-UNCPBA, Campus Universitario, Tandil, Argentina

Abstract

This work aims at detecting pedestrians in surveillance video sequences. A pre-processing step detects motion regions on the image using a scene background model based on level lines, which generates a Movement Feature Space, and a family of oriented histogram descriptors. A cascade of boosted classifiers generates pedestrian hypotheses using this feature space. Then, a linear Support Vector Machine validates the hypotheses that are likeliest to contain a person. The combination of the three detection phases reduces false positives, preserving the majority of pedestrians. The system tests conducted in our dataset, which contains low-resolution pedestrians, achieved a maximum performance of 25.5 % miss rate with a rate of of 10^{-1} false positives per image. This value is comparable to the best detection values for this kind of images. In addition, the processing time is between 2 and 6 fps on 640x480 pixel captures. This is therefore a fast and reliable pedestrian detector.

Keywords: pedestrian detection, level lines, movement detection, Adaboost cascade, linear SVM

1. Introduction

Pedestrian detection, as well as character and face recognition, vehicle detection, etc., is considered a key chapter of pattern recognition in computer vision. The various applications associated with this subject, such as surveillance, safety systems, robotics, and content-based indexing, attracts the interest of researchers and manufacturers, as well as the military and security agents.

This variety results in innumerable algorithms and methodologies that have been recently developed [1, 2, 3]. However, a recent and thorough analysis of the state of the art by Dollar et al [4] concludes that, for certain applications or architectures, pedestrian detection is still an open problem, e.g. cases of low-resolution pedestrian images. Such is the case of a person's 3D motion model [5], which may have between 14 and 22 degrees of freedom, showing the countless

Email address: pnegri@uade.edu.ar (Pablo Negri)

combinations of potential positions, let alone the changes in appearance due to clothes, points of views, lighting conditions, etc. Owing to these factors, pedestrian detection through computer vision remains a constant challenge in the field of pattern recognition.

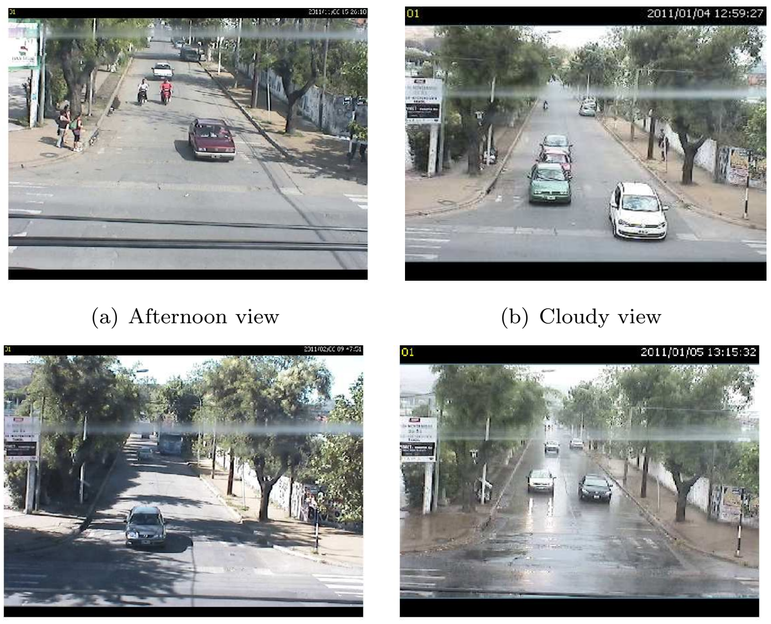
The various pedestrian detection applications can be classified by the type of architecture used in the image acquisition, namely: static cameras or views (still detection), fixed-camera video sequences, or mounted camera sequences (e.g. robots, vehicles). Even though the nature of the images used may be different, the methods are all consistent in finding the best descriptors adapted to the image dataset and a classifier making it possible to find pedestrians based on those descriptors.

In still detection, people are identified in photographic images [6, 7, 8, 9, 10, 11]. In general, a person's position or size scale on the image is unknown. The system thus needs to evaluate the entire image in different scales. There are multi scale descriptors, such as the Haar-like filters [6], or the Histograms of Orientations (HOG) [8, 9], which are then used by SVM-type classifiers in the first case, or by cascades of boosted classifiers in the second. It is also possible to obtain a pyramid of scales from the original image, calculating the HOG in a fixed scale, and using a linear SVM classifier [7] or a latent SVM classifier [10].

In the case of a sequence of images taken from a fixed camera position, temporal information is generally used. This information may be used creating descriptors based on the video flow [12, 13, 14], which require a high frame rate of the sequence. A background model or reference image is also used to identify the pixels that do not belong to this model, and it is inferred that they are part of moving objects. The primitives used may be the level lines [15, 16, 17], Mixture of Gaussians [18, 19], or a combination of color, texture and HOG channels [20]. Identifying the motion in the scene, assuming that pedestrians are actually moving, makes it possible to reduce the search space for the system, to create new descriptors, or to calculate confidence filters or maps.

Finally, the application involving on-board cameras is used in robotics for human computer interaction, or in intelligent vehicles with pedestrian avoidance systems to minimize accidents [21, 22, 23]. In this case, the complexity lies in the fact that the camera is in motion, processing the dynamic flow in the sequence which must be solved to identify pedestrians. Gravila [21] uses a disparity map from a stereo system to get a silhouette that is subsequently evaluated on a hierarchical tree of template shapes, using the Chamfer distance. Dollar [11] prefers to perform a frame-by-frame analysis, without using the motion information, and to meet the real-time requirements using a fast cascade detector and a multi-scale calculation of gradient histograms.

The system discussed in this paper seeks to detect pedestrians in outdoor sequences captured by a fixed remote camera. To get images with an adequate resolution for detection, the sequence that is generally obtained is reduced to a frame rate of 1 or 2 fps, given the limited bandwidth available. The applications involved mainly consist of remote surveillance systems, where it is important to detect intruders in restricted or dangerous areas, such as railroads or highways. The detection of people allows a surveillance system to report an improper


(a) Afternoon view (b) Cloudy view

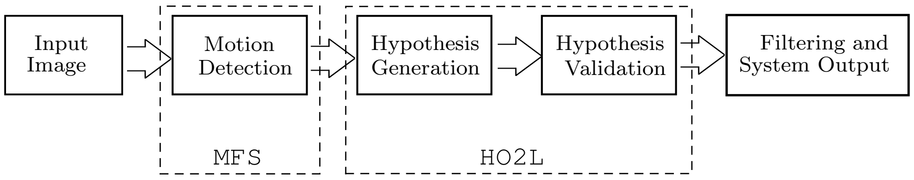
(c) Lateral Shadows (d) Rainy view

Figure 1: Capture samples of the recorded sequences.

presence in these dangerous areas so as to carry out appropriate actions. The system should be robust to rapid and significant changes in the scene (e.g. shadows, weather conditions), and to the presence of many other moving objects (e.g. vehicles, trains, etc.), as shown on figure 1. These images, which make up the dataset of this paper, correspond to different captures of a camera mounted in a traffic light. Moreover, the camera view is not exactly fixed because of the movement caused by the blowing wind or traffic vibrations.

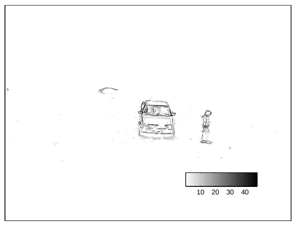
The proposed detection system consists of four stages: motion detection, hypothesis generation, hypothesis validation and final filtering, as shown on figure 2(a).

In the first stage, the motion detection uses a level line based approach, illustrated in figure 2(c), generating a Movement Feature Space (MFS). The MFS information is grouped in order to obtain a descriptor family that we call Histograms of Oriented Level Lines (HO2L). Then, they are used by a cascade of boosted classifiers to generate hypotheses of the presence of a person, see fig. 2(d), restricting the search space to some positions within the image. The hypotheses or regions of interest (RoIs) are then validated by a linear SVM classifier using the HO2L descriptors grouped in a configuration similar to the R-HOG [7]. Validated RoIs, as shown on fig. 2(e), are then grouped using a Mean Shift Clustering method [24, 25] or the Non-Maxima Suppression algorithm. Returned bounding boxes (RoI positions) are considered the system output (see

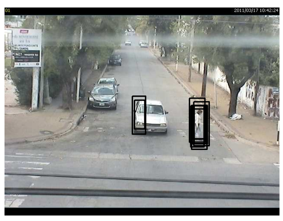


(a) Detection Schema

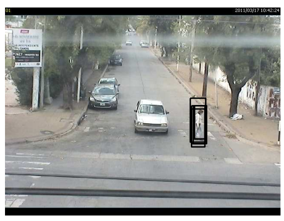
(b) Original Image



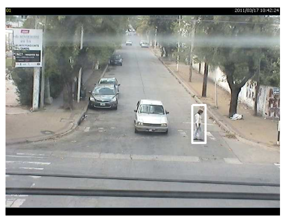
(c) Motion Detection



(d) Hypothesis generation



(e) Hypothesis validated



(f) Final RoIs

Figure 2: Overall sequence of the detection algorithm.

figure 2(f)).

The main advantages of the system proposed in this paper include an increased robustness, minimized loss of information, and computation time efficiency.

We have developed 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. The MFS adapts well to slow changes in the scene while it is robust to rapid variations (e.g. illumination changes or weather conditions). In these situations, people's appearance on the MFS does not change significantly compared to normal conditions.

The calculation of level lines in a transformed HSV color space, called Texton Color Space (TCS) [26], which makes it possible to retrieve color transitions that are lost when working on gray levels, is proposed.

The orientation histograms obtained from the MFS allow for a multi-scale pedestrian detection, avoiding the construction of a dense pyramid of subsampled versions of the input image that is very costly in terms of calculation time. They can also be computed quickly using the integral histogram, which offers the possibility of accelerating the detection stage. The combination of generative and discriminative classifiers speeds up the search for pedestrian hypotheses on the image, as proven in [27].

In summary, the system calculates the MFS of the sequence and continues to work on this space without using the information from the original image again, either the gray levels or the color channels. The desired response time is between 2 and 6 fps in sequences of a 640x480 frame size with a miss rate of 25.5 % at 10^{-1} false positives per frame. This result is comparable to the best methods of the state of the art in still detection [4].

The structure of the paper is as follows: section 2 details the methodology to obtain the MFS, develops the hypothesis generation algorithm using the cascade of classifiers, and details the SVM validation classifier. Various experiments and analyses of classifiers are described in section 3. The system results and the comparison with the state of the art are found in 4, while section 5 provides conclusions and perspectives.

2. Algorithms and Methodology

2.1. Movement Feature Space Based on Level Lines

Motion detection in video sequences is carried out, in general, using background subtraction algorithms. They model an adaptive image reference capturing the static information of the scene background. Every frame in the sequence is compared against the model and the difference between them indicates the presence of a new object.

The algorithm used in this paper is based on the work by Bouchafa and Aubert [15, 16]. They propose the use of level lines as primitives for the reference model. This methodology is also adaptive incorporating into the background model the changes in the scene, such as new objects, shadows, modifications, etc. In [17] they use level lines and moving region detection using the a contrario approach to detect new objects in the scene. The processing time of this method depends on the information within the picture, which is less adapted for use in on-line system detection. Another method to construct the reference model in order to detect persons is shown in the work by Mokhber et al [18]. They use a Mixture of Gaussians [28] modeling the colors of each pixel in the reference. However, in outdoor sequences tests, this algorithm was not robust to rapid changes in the scene, such as strong cast shadows, passage of a cloud, or weather variations. In these cases, the algorithm generates big binary detection regions, where it is not possible to distinguish a moving object.

2.1.1. Definition of Level Lines

Let I be a monochromatic 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 level λ is the set of pixels $\mathbf{p} \in I$, so that their intensity is greater than or equal to λ,

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

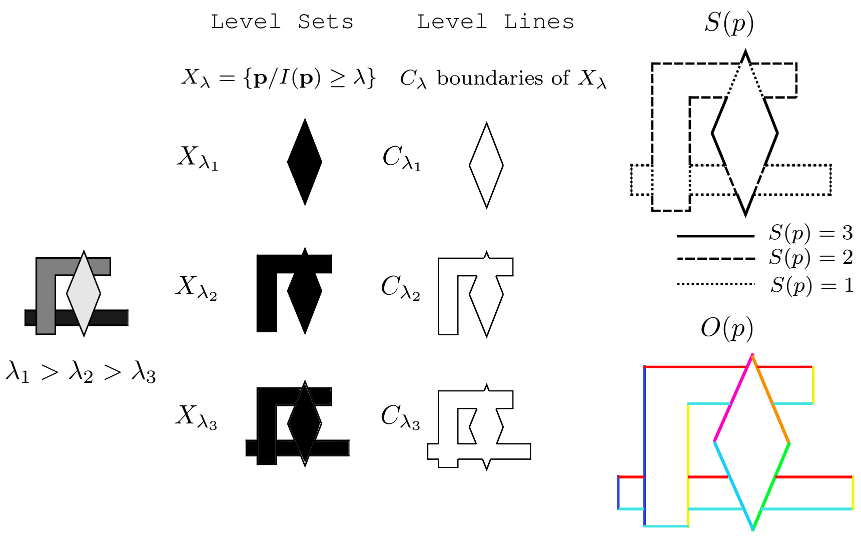


Figure 3: Level lines extraction toy example.

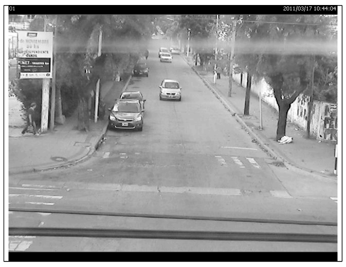
For each λ, the associated level line C_λ is the boundary of the corresponding level set X_λ [29]. Let $\Lambda = \{\lambda_1, ..., \lambda_N\}$, be a set of N equally spaced thresholds calculated from both the minimum and maximum values of I. The set Λ generates a family of N level lines $\mathcal{C} = \{C_{\lambda_1}, ..., C_{\lambda_N}\}$ that we use to 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. Figure 3 shows the value of $S(p)$ as the addition of $C_{\lambda_1} + C_{\lambda_2} + C_{\lambda_3}$. When considering all the gray levels ($N = 255$), this quantity is highly correlated with the gradient module.

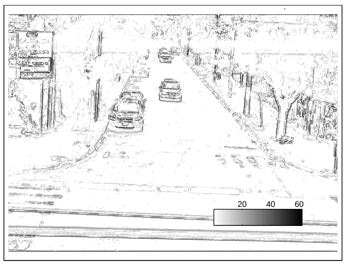
- $O(p)$ is the gradient orientation at p, computed in the level set X_λ using a derivative filter of 5×5 pixels [30]. After calculating the orientation of all p at the boundaries of X_λ, the degree values are quantized to η integer values. As the orientations are calculated for each X_λ of the set, the same pixel p can have more than one orientation value. The value assigned to $O(p)$ is the most repeated orientation in the set.

Generally, in a 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.

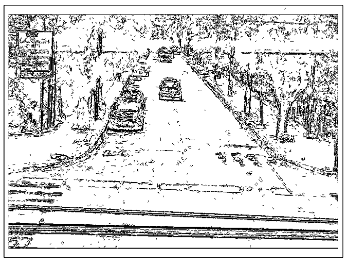
In Fig. 4 two different sets of equally spaced thresholds $\Lambda_1 = \{\lambda_1, ..., \lambda_{N_1}\}$ and $\Lambda_2 = \{\lambda_1, ..., \lambda_{N_2}\}$, were used with $N_1 = 64$ and $N_2 = 128$ in the same capture, and $\delta = 1$. With this value of δ, the preserved level lines of $N = 128$ are those for which the grayscale transition has half the values than those of the level lines of $N = 64$. The great number of details (and noise) obtained in 4(e) can be clearly noticed.



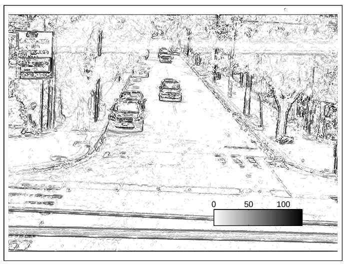
(a) Image Original



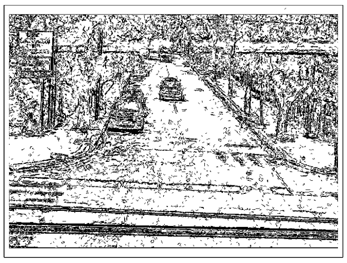
(b) S_t for $N = 64$, $\delta = 1$



(c) $S_t > 0$ for $N = 64$, $\delta = 1$

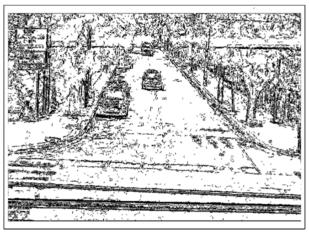


(d) S_t for $N = 128$, $\delta = 1$

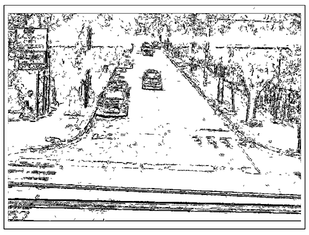


(e) $S_t > 0$ for $N = 128$, $\delta = 1$

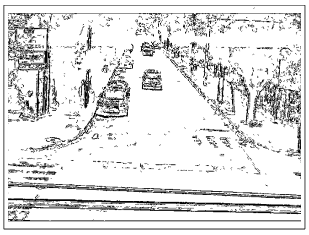
Figure 4: This figure shows two choices for parameter N applied on a sequence frame. $N = 64$ captures fewer details than $N = 128$.



(a) $S_t > 0, \delta = 1$, $N = 96$



(b) $S_t > 0, \delta = 2$, $N = 96$



(c) $S_t > 0, \delta = 3$, $N = 96$

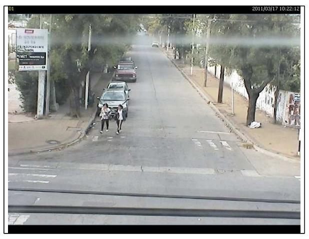
Figure 5: Figure showing the effect of increasing parameter δ for the same value of N.

In the case of Fig. 5, three values of threshold δ were applied to the same $S(p)$ with $N = 96$. Increasing δ helps to eliminate the noise in the image, thus preserving stronger transitions. For example, for the pair $(\delta = 1, N = 96)$, the minimal transition to create a level line having 256 gray values, is $256/N = 2.6$, while for the pair of parameters $(\delta = 3, N = 96)$, the minimal transition is $3 * 256/N = 8$.

2.1.2. Motion Detection

As described in [31], level lines have many properties, being invariant to contrast changes. This means that a regular contrast change (monotonic and upper semicontinuous) 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 [16]. This is crucial because we will use level line intersections to detect movements. This last assertion means that our method will be robust to regular contrast variations.

Now, let two consecutive images, I_{t-1} and I_t, be obtained at times $t-1$ and t. When looking for scene changes at pixel p, the variation of $S_t(p)$ in comparison with S_{t-1} could correspond to a movement, but also to a change in contrast. A more reliable indicator is a variation on $O(p)$, i.e., $O_{t-1}(p) \neq O_t(p)$, which means that there is a new level line 'crossing' the old one. However, the number of points verifying that condition between two consecutive images could be very few. Bouchafa and Aubert [15, 16] define an adaptive background reference model, composed of the set of pixels $\mathbf{p}$ which are stable over a period of time, together with the corresponding values $O^R(\mathbf{p})$.


(a) Frame at t

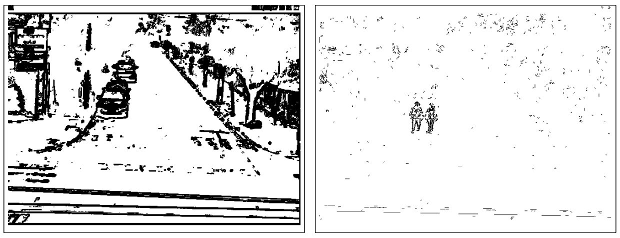

(b) Reference R_t (c) Motion detection D_t

Figure 6: Motion detection at time t, showing the background model reference R_t and the detected movement D_t.

More precisely, given a horizon of time T, we define array R_t as the reference model determined by:

$$R_t = \{p \in I_t : O_{t-1}(p) = O_{t-2}(p) = \cdots = O_{t-T}(p)\} \tag{1}$$

All the pixels p satisfying this condition are preserved in array R_t which is made up of η layers, one for each orientation value.

In practice, the equality constraints in the definition of reference space R_t are relaxed to allow for small orientation variations due to noise or other perturbations, calculating a frequency of occurrence array F_t (see [15, 16] for details).

Thus, at time t, the set of pixels p, which are not in the reference or which have an orientation other than the reference: $O_t(p) \neq O_t^R(p)$, are inferred to belong to moving objects. These pixels will make up the detected set D_t. Figure 6 shows an example of the reference model of the video sequence at time t. The detected set D_t is presented in Fig. 6(c). Note that for this frame, parked cars belong to the reference model and do not appear in D_t.

The η parameter indicating the quantization of the level line orientation values is calculated using the 2π module. A significant number of orientations helps to discriminate the motion level lines better than those belonging to the reference. However, in order to be robust to the high variability in human appearance, the colors of clothes, etc., after obtaining matrix D_t, the values of O_t are quantized in η_d values to modulus π so as not to differentiate between a dark-bright and a bright-dark transition($\eta_d = \eta/2$).

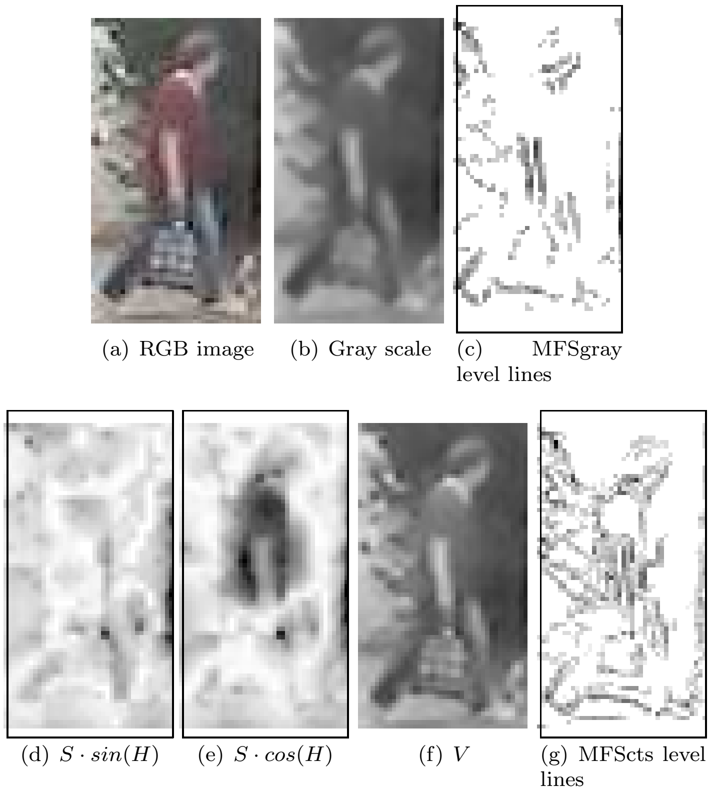
Below, we will focus the analysis only on pixels in the detected set D_t, and their values of S_t and O_t. This set can be considered as a virtual image with two associated scalar fields, or a kind of feature space referred to as *Movement Feature Space* (MFS).

2.1.3. Colored Level Lines

The color of an object in the scene is the result of the body reflection. It depends on two characteristics related to the physical properties of the material: the penetration of light and the scattering of the body's pigments [32]. Clothes are opaque bodies and reflect light in a way that interferes with the detection of color transitions between a person and the background. This difficulty increases when detecting small-sized objects and when the capture is converted into a gray scale.

Fig. 7(a) shows a person wearing a red t-shirt walking in front of a green hedge. If we transform the color image into its gray-scale representation (see fig. 7(b)), the RGB average approaches both colors and it is impossible to find a transition between them without generating any level lines (see Fig. 7(c)).

We thus propose to work with a transformed HSV color space, where it is possible to recover the transition between the body (clothes) and the background finding the level lines shown in Fig. 7(g). In this space, Hue (H) is the color feature, Saturation (S) measures the degree of purity of the Hue, and Intensity (V) is the average gray level. Carron [33] proposes this transformation scheme


(a) RGB image (b) Gray scale (c) MFSgray level lines

(d) $S \cdot sin(H)$ (e) $S \cdot cos(H)$ (f) V (g) MFScts level lines

Figure 7: Level lines generation using the MFSgray and MFScts spaces for the same pedestrian example. Note that this image is the pedestrian of figure 4.

because color features are less sensitive to non-linear effects, being less correlated than the RGB color space.

If Saturation has high values, the Hue is very pertinent. In contrast, when Saturation has small values, Hue is noisy or unstable, and, consequently, it may be irrelevant [33]. The latter means that Hue is ill-defined in unsaturated cases, and this channel can generate irrelevant level lines [32].

To overcome this, Alvarez et al. [26] introduce a simplification of Carron's method referred to as Color Texton Space (CTS). In this space, two new channels are generated: $S \cdot cos(H)$, and $S \cdot sin(H)$. Intensity V remains unchanged. Thus, in a pixel where H is not relevant because of the low value of S, those channels do not have an important value.

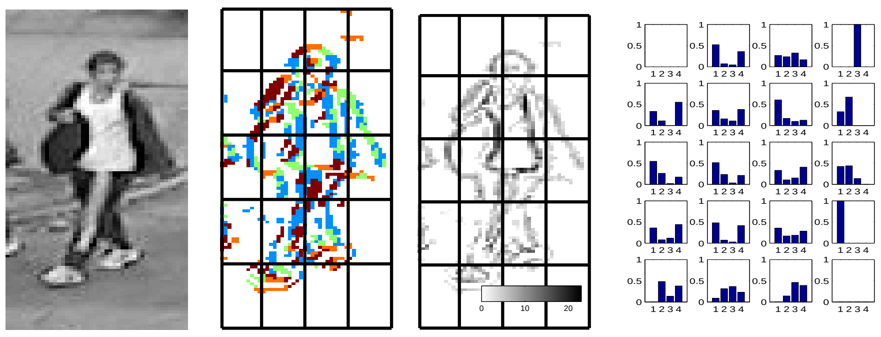
First, we calculate $S_t^x(p)$ and $O_t^x(p)$ for each channel of the Texton Space, where $x = \{SsinH, ScosH, V\}$. Then, in order to obtain $S_t^{CTS}(p)$ and $O_t^{CTS}(p)$, for each pixel p, the orientation and the modulus of the greatest $S_t^x(p)$ are chosen. Finally, we obtain D_t from $S_t^{CTS}(p)$ and $O_t^{CTS}(p)$, as explained above.

Hereinafter, we will refer to the system obtaining the MFS from the grayscale image as MFSgray, and to the system calculating the MFS using the Color Texton Space as MFScts.

2.2. Generating Hypotheses on the MFS

The generation of hypotheses seeks to restrict the search space within the image, regardless of moving objects other that people. At runtime, the MFS generates motion information produced by pedestrians, other moving objects (vehicles in our dataset), or different sources of noise, such as lighting changes or camera movement. Noisy or motionless regions should be discarded quickly in order to focus the processing on pedestrian hypotheses. The cascade of boosted classifiers proposed by Viola & Jones [34] is perfectly suited to perform this task. This architecture has the particularity to achieve real-time detection with an excellent performance. Futhermore, the behavior of the cascade can be improved by combining generative and discriminant classifiers [35, 27, 10]. Generative classification involves an a priori knowledge about the target object, by constructing a model of the class, and finding a similarity score with this model. Discriminant classification, on the other hand, draws a hyperplane in the feature space and any test sample is evaluated depending on the side of the plane where it was placed. [35, 10] generative classifiers are applied to the search for hypotheses, which are then validated by a more complex or a discriminant classifier. In [27], the unsupervised learning of the Adaboost cascade selects generative classification functions in the first stages and discriminant functions for the subsequent ones. In this paper, we suggest combining these two types of classification functions in the cascade.

From then onwards, pedestrian hypotheses, bounding boxes, and region of interest (RoI), refer to a rectangle within the image with a size proportional to 12x30 pixels.



(a) Person (b) O_t grid (c) S_t grid (d) MFS^{HO2L} features

Figure 8: MFS^{HOG} calculation on defined grids. In (b), each color corresponds to one of the four directions of the level lines. In (c), darker pixels have higher S_t values.

2.2.1. Descriptors for the Cascade

To encode the information inside each hypothesis we will employ two kinds of descriptors using the matrix $S_t(p)$ and $O_t(p)$ calculated by the MFS algorithm at time t. The first one are histograms of oriented level lines (HO2L), and the second one accumulates the values of S_t inside a patch. As in [34], where the set of descriptors of a pattern of 32x32 pixels was made up of thousands of Haar-like filters, we suggest to calculating our descriptors on a dense grid of overlapped patches in a 12x30 pattern size ($\Delta w_p = 12$,$\Delta h_p = 30$).

A patch is a rectangle defined by $\{x, y, \Delta w, \Delta h\}$, where x and y are the horizontal and vertical positions within the pattern, and Δw and Δh are their size. We define three types of patches: square $r_s = \{x, y, l, l\}$, vertical rectangles $r_v = \{x, y, l, 2l\}$, and horizontal rectangles $r_h = \{x, 2l, l\}$, where l is the rectangle side taking the values $\{2, 4, 6, 8\}$. Positions (x, y) are increased by 1 pixel, always ensuring that the patches are within the pattern. This methodology generates a set of 1 985 patches: 852 squares, 421 horizontal rectangles, and 712 vertical rectangles.

The MFS^{HO2L} descriptor j is histogram $\mathbf{h}$ with η_d bins, one for each orientation η, which is computed as follows:

- Let $r_j = \{x_j, y_j, \Delta w_j, \Delta h_j\}$ be the patch position of descriptor j in the pattern,
- For each bin o of $\mathbf{h}$, we add all the $S_t(p)$ values for p with this orientation, $\mathbf{h}(o) = \{\sum_{p \in r_j} S_t(p)/O(p) = o\}$,
- The histogram values are normalized using $L1 - norm$, $\mathbf{h} \rightarrow \mathbf{h}/(\|\mathbf{h}\| + \epsilon)$.

The histograms can be evaluated easily using the histogram integral by $4\eta_d$ access to this array. Moreover, these descriptors are multi-scale: when a bounding box $\{x_{BB}, y_{BB}, \Delta w_{BB}, \Delta h_{BB}\}$ with $\Delta w_{BB} = \alpha \cdot \Delta w_p$ and $\Delta h_{BB} = \alpha \cdot \Delta h_p$

is evaluated, factor α is used to compute the position and size of r_j inside the bounding box.

The example of Figure 8 shows a pedestrian bounding box of size $\{\Delta w_{BB} = 46, \Delta h_{BB} = 115\}$, with $\alpha = 3.83$. The non-overlapped rectangular patches illustrated were generated considering $l = 3$, which is converted to $l_{BB} = \alpha \cdot 3 = 11.5$. Then the scaled size of the patches r_j calculated for the bounding box are $\{\Delta w_j = 12, \Delta h_j = 30\}$. For the sake of simplicity, we consider four directions for the descriptor MFS^{HO2L} calculated within each patch, where histogram bin 1 corresponds to the vertical direction, bin 3 corresponds to the horizontal direction, and the other two are the diagonal directions.

The second set of features, called MFS^{MAG} computes the sum of the S_t values inside each patch. This feature contributes to the fact that HO2L descriptors cannot differentiate between a strong edge in the patch, generating a number one in the corresponding bin, and only one pixel of noise, which creates the same histogram after normalization.

The integral histogram is used to calculate MFS^{MAG}, but, instead of getting the histogram, the bins values are added before normalization.

2.2.2. Cascade Detector

The generation of hypotheses is carried out by a cascade of boosted classifiers [34] discriminating pedestrian and non-pedestrian bounding boxes. Each boosted classifier is trained using the Real Adaboost algorithm [36]. They are called strong classifiers because they are the lineal combination of T simple classification function $g \in \Re$ known as weak functions. Let x be an input sample, the strong classifier $G(x)$ is defined as follows:

$$G(x) = \sum_{t=1}^{T} g_t(x) \tag{2}$$

Input x is evaluated considering the sign of $G(x)$, or compared to a threshold.

The cascade architecture evaluates an input x by a series of strong classifiers G_k of increasing complexity, where complexity is associated whit the number T_k of weak classifiers g that compose each one. If x is not validated in one of the stages, it does not continue to the next stage and is eliminated. The first G_k stages are made up of very few g, maybe only one, eliminating a negative sample x quickly, while negative hypotheses similar to pedestrians (in the feature space), are evaluated by complex classifiers with a stronger discriminant power.

This classifier has a good performance, it evaluates about 10 000 sliding windows in some milliseconds and delivers to the next detection step only the most probable pedestrian hypotheses. In addition, because of the use of the MFS, those frames of the sequence without motion do not generate any hypotheses: all the tested bounding boxes are eliminated in the early stages of the cascade.

As equation 2 shows, the strong classifier $G(x)$ is the combination of classification functions $g(x)$ associated with descriptors calculated within the patches. At each learning iteration of Adaboost, the different $g(x)$ are calculated using the learning dataset and a distribution of weights W [36]. The sign of $g(x)$, which has real values, provides the label to input x (+ 1 or - 1), while module $|g(x)|$ gives a value of 'confidence' in the prediction. Each function $g(x)$ makes a classification error in the learning dataset, so the one with the minimum error is preserved in the linear combination of $G(x)$. We propose here that these $g(x)$ can be of different nature: discriminant or generative.

The first classification functions presented are of a discriminant nature, computed from the oriented histogram descriptors:

- $g_{j,o}^{MFS_{disc}^{HO2L}}(x)$: the input is the value of bin o in histogram MFS^{HO2L} of descriptor j.

Other works have used the HOG descriptor associated with a discriminant function. In [9], when the Adaboost algorithm selects one bin of the histograms in the feature space, the others histogram bins are also incorporated into the classifier. Zhu [8], however uses the vector made up of the histograms within a block, one for each cell, to train an SVM classifier at each Adaboost iteration. Here, each bin of the histogram is treated individually and is incorporated into the strong classifier without considering the other bins of the histogram. In pedestrian detection, this classification function identifies the patches where there is a predominant orientation in the training dataset (usually the vertical orientation).

Unlike before, we will propose a methodology that considers the histogram information as a whole and does not require any learning. To compute this generative classification function we create a model representing a priori knowledge of the pedestrian class, using the HO2L descriptors.

For each feature j, a histogram model $\mathbf{m}_j^{HO2L}$ is associated using the information of positive samples in the training dataset. A method to obtain the model implies computing the median value of each bin of the descriptors j from the positive dataset [27].

In this paper, the histogram model is obtained minimizing an error criterion. The idea is to seek histogram $\mathbf{h}_{j,i}$ of the positive sample x_i that best represents feature j in the positive dataset. The methodology involves calculating, for each sample x_i, a vector $\mathbf{v}_i$ composed of the distance of their $\mathbf{h}_{j,i}$ to the features j of the others samples x_k in the dataset: $\mathbf{v}_i = [d(\mathbf{h}_{j,i}, \mathbf{h}_{j,k})/k = 1, ..., P \cap k \neq i]$. The distance between two HO2L histograms is calculated using the Bhattacharyya distance [37]:

$$d(\mathbf{h}_{j,i}, \mathbf{h}_{j,k}) = \sqrt{1 - \mathbf{h}_{j,i} \cdot \mathbf{h}_{j,k}} \tag{3}$$

and $[\cdot]$ is the scalar product. This distance, with values in the range of [0,1], is widely used to measure the similarity among histograms, and it has a good compromise between performance and complexity. The descriptor j minimizing the $L2$ norm of the distance vector is asigned to the histogram model $\mathbf{m}_j^{HO2L}$.

The generative classification function is defined as follows:

- $g_j^{MFS_{gen}^{HO2L}}(x)$: where the input is the Battacharyya distance between the descriptor MFS^{HO2L} j of the x sample, and the model MFS^{HO2L} associated with j (eq. 3).

The descriptor MFS^{MAG} is associated with a discriminant classification function:

- $g_j^{MFS^{MAG}}(x)$: the input is the sum of the $S_t(x)$ values of the pixels inside patch r_j.

Function $g_j^{MFS^{MAG}}(x)$ can eliminate those bounding boxes where no major transitions, or motion noise were generated, e.g. camera vibrations, slight movements of objects (trees branches blown by the wind).

2.3. Hypothesis Validation

In order to increase the precision of the system we evaluate the hypotheses that are classified as positive by the cascade of boosted classifiers using a linear Support Vector Machine (SVM). These hypotheses can contain true pedestrian bounding boxes and false positives. The SVM classifier has to discard as many non-pedestrian bounding boxes as possible while keeping the true positives regions. This is a difficult problem since the hypotheses have already been classified as positive regions by a previous non linear classifier. However, the SVM has the advantage that classification is made on a higher feature space. Different types of kernel machines can be considered: linear kernel [7, 22], polynomial, RBF, or latent SVM [10], etc. An important advantage of the linear SVM is that the classifier can be evaluated very efficiently during test time. For this reason it is the right choice for on-line detectors.

2.3.1. Descriptors for the SVM Classifier

The input features for the SVM classifier are the HO2L descriptors grouped in a configuration similar to that of the R-HOG descriptors proposed by Dalal [7], which we call R-HO2L. They are computed over a dense grid of superimposed blocks at a single scale of $\varsigma \times \varsigma$ cells of size $\rho \times \rho$ discretizing the gradient directions in η bins. The blocks are normalized using the *L2-Norm*: $\mathbf{v} \to \mathbf{v}/\sqrt{\|\mathbf{v}\|_2^2 + \epsilon}$. The pattern for the SVM descriptors have 24x60 pixels size, which is the minimum size of the pedestrians in our dataset.

2.3.2. SVM Classifier

Hypotheses are validated by a hyperplane learning algorithm referred to as Support Vector Machine (SVM) [38]. For linearly separable problems, among all the hyperplanes separating the training data on the feature space (person versus non-person classes), there will be a single optimal hyperplane maximizing the separation margin, as shown on Fig. 9 [39]. Let $\{\mathbf{x}_i, y_i\}$ be a training dataset, where $y_i \in \{-1, +1\}$, $\mathbf{x}_i \in \Re^d$. Thus, the classification of sample $\{\mathbf{x}_i$ 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 points at which equation 4 holds are called support vectors and define two parallel planes (the dotted lines in Figure 9) on both sides of the hyperplane separated by margin $2/||\mathbf{w}||$. The

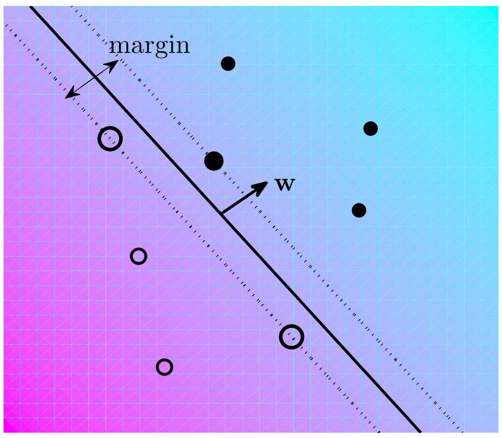


Figure 9: Hyperplane for a linearly separable problem. The dotted lines represent the margin defined by the three support vectors: two O's and one filled point.

proposed optimization problem seeks to find the pair of hyperplanes that maximize the margin, while minimizing $||\mathbf{w}||$, subject to equation 4 $\forall i$.

The Lagrangian formulation introduces the Lagrange multipliers α_i, one for each learning sample. There is a single solution which has $\alpha_i \neq 0$ only for the support vectors, and which defines the hyperplane below:

$$\mathbf{w} = \sum_{i=1}^{N_s} \alpha_i y_i \mathbf{x}_i \tag{5}$$

When the problem is not linearly separable (our problem), the restrictions change to allow for the possibility of mis-classified samples. To solve this new optimization problem, slack variables and new constrains are introduced in order to find a soft margin classifier. In this case, the solution of equation 5 holds, with the difference that it adds a new constraint giving α_i an upper bound C.

After the SVM training, $\mathbf{w}$ is calculated from equation 5, and stored. This vector will always have the same dimension d, regardless of the number of support vectors that define it. The dot product of input vector sample $\mathbf{x}$ and $\mathbf{w}$ is sufficient to evaluate it. That is why the linear SVM is chosen for on-line detectors, over other nonlinear SVM classifiers. The latter, using polynomial kernels, RBF, etc., have to evaluate the input sample with each support vector, which improves the performance, but increases the computation time.

The linear SVM training was carried out by the OpenCV implementation [40], using a two round bootstrapping approach [10].

3. Experiments

3.1. Datasets

Video sequences were recorded using a Vivotek SD7151 camera, filming an intersection in the city of Tandil (Argentina). The recording format is MJPEG with a 640x480 pixels size. We have chosen the minimum JPEG compression to reduce blocking artifacts in captures. In addition, a bilinear interpolation is applied to input frames: on each HSV channel and on the gray scale image. With this JPEG resolution, the limited network bandwidth reduces the recording process to one capture every three seconds, on average.

The dataset is made up of seven video sequences. Five of them are used to train the classifiers, having 1 110 labeled pedestrian in more than 5 000 frames. In the test sequence, there are 1 357 labeled pedestrians in 4 200 frames.

Pedestrians circulate at a distance between 30 and 60 meters from the camera. Their average height in the image is 65 pixels, representing a dataset with a very low resolution. Moreover, 45 % of the pedestrians are partially occluded by other pedestrians, and out of them, 8.6 % are highly occluded by an overlapping factor of 0.4.

In the cascade of boosted classifiers learning phase, each classifier is trained using a dataset consisting of 4 500 negative samples randomly picked from the sequences, and 6 600 positive samples. The positive dataset is obtained using the sliding window approach on the training sequences, similarly to the approach used for testing, to retrieve 3 300 positive bounding boxes which have a high overlapping ratio (over 0.85) on the 1 110 ground true training samples. Taking their left-right reflections, we double the positive number.

The SVM classifier training phase uses 2 220 positive samples (ground true bounding boxes and their reflections) and more than 12k negative samples randomly picked from training sequences and negative images from the INRIA Person dataset, in the first round of the bootstrapping learning. In the second round of the bootstrapping, the negative set is made up of the mis-classified samples from the first negative set and incremented by the false positives found by the first classifier on the training sequences and the INRIA negative images.

3.2. Evaluation of Results

The system response is a set of bounding boxes $B = \{B_d(1), B_d(2), ..., B_d(i)\}$, and their associated classification score s_i. To evaluate the performance, this set is compared against the pedestrian real bounding boxes B_{gt}, referred as ground-true. The overlapping criterion is the same as that proposed in Challenge Pascal [41]. If a bounding box B_d exceeds the overlap factor over B_{gt}, it is considered to be as a correct detection, or otherwise a false positive. If there is more than one bounding box overlapping the same B_{gt}, only B_d remains with the highest overlapping criterion, and the others are considered to be false positives.

Score s_i associated to $B_d(i)$ is computed using equation 4, as the scalar product between the feature vector $\mathbf{x}_i$ of $B_d(i)$ and the linear SVM $\mathbf{w}_{linSVM}$, plus factor b:

$$s_i = \mathbf{x}_i \cdot \mathbf{w}_{linSVM} + b \qquad (6)$$

To compare the performance of different versions of the classifier, we will use the False Positive Per Image (FPPI) rate, which is better adapted to the sliding windows testing approach than the Detection Error Tradeoff (DET) [4].

The set of bounding boxes B consists of those $B_d(i)$ with a score $s_i > 0$, on the positive hyperplane side. To draw the FPPI curve, thresholds of increasing values will be applied to set B. Each threshold value generates a point in the FPPI curve.

Validated bounding boxes are filtered by the non-maxima suppresion algorithm (NMS) [10]. The overall miss rate and false positive rate of the test sequences are obtained from the resulting bounding boxes.

The choice value to compare the miss rate of the classifiers correspond to a FPPI of 10^{-1}, instead of the *log-average miss rate*. From [4], the *log-average* is computed as the average of nine evenly spaced FPPI rates in the range of 10^{-2} to 10^{0}. In our system, almost all curves end before this maximum value. However, as claimed by Dollar, the miss rate at 10^{-1} FPPI and the *log-average miss rate* yield similar results.

3.3. Parameter Selection

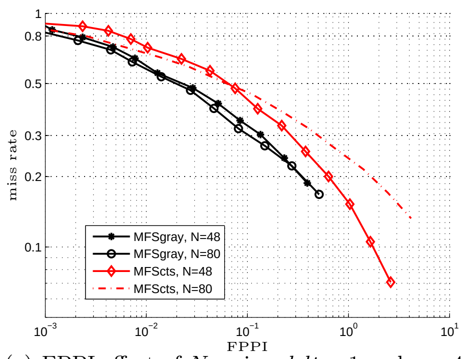
In this section different parameters will be evaluated, especially those affecting the hypothesis validation stage (linear SVM classifier). For our system, the latter is critical and must have the best performance possible: a lower miss rate with fewer false alarms. The influence of these parameters on motion detection and hypothesis generation stages does not affect the system performance significantly. The curve results are the average of 5 fold independent trained classifiers.

3.3.1. Parameters of Level Lines

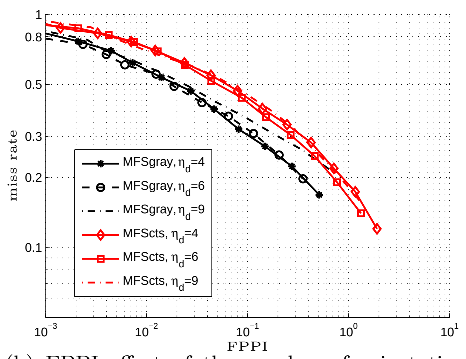
Figure 10 evaluates the performance of classifiers varying parameter N of the level lines, which corresponds to the number of thresholds in the set $\Lambda = \{\lambda_1, ..., \lambda_N\}$. Figure 10(a) shows the results. For the MFSgray space, the miss rate of $N = 80$ is 30.7 % at 10^{-1} FPPI. Decreasing the number of thresholds to $N = 48$ increases the miss rate in 2 %. For the space MFScts, where there are more available information, the effect is the opposite, changing from a miss rate of 46.1 % with $N = 80$ to 43.5 % with $N = 48$.

Figure 10(b) shows the FPPI effect of the quantization parameter η_d of level line orientation. The best performance is obtained using a value of $\eta_d = 4$ bins for the MFSgray space. When $\eta_d = 6$, performance drops by 1 %, thus showing that there is no significant difference between performances. However, when choosing a higher number of orientations, $\eta_d = 9$, performance decreases by 9 %. For MFScts, a performance with $\eta_d = 6$ is optimum.

Another important parameter in level line encoding is the value of δ, which was the threshold applied to obtain relevant level lines. Figure 11 shows the

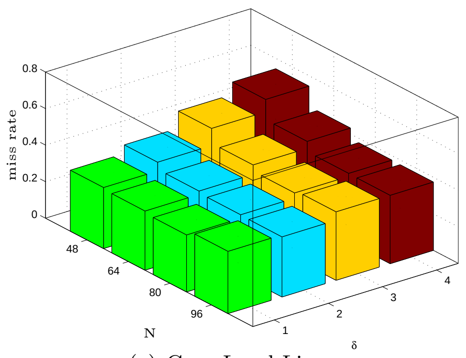


(a) FPPI effect of N, using $delta$=1 and η_d=4

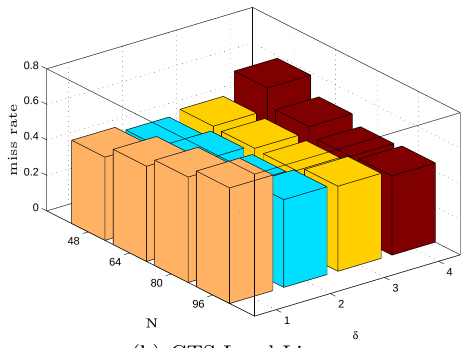


(b) FPPI effect of the number of orientation bins η_d, using $delta$=1 and N=80 for MFSgray, and $delta$=1 and N=48 for MFScts.

Figure 10: FPPI performance of the SVM classifiers with different parameters.



(a) Gray Level Lines



(b) CTS Level Lines

Figure 11: FPPI effect of N and δ.

miss rate values at FPPI 10^{-1} for different combinations of N and δ, since these two parameters are closely related.

The optimal value for MFSgray is $N = 80$, with $\delta = 1$ and a miss rate of 30.7 %. For the MFScts space, the value of $N = 48$ with $\delta = 2$ minimizes the error to 42.3 %.

3.3.2. R-HO2L Encoding Parameters

In this work, the block has $\varsigma \times \varsigma$ cells, with $\varsigma = 2$, as the size of our pattern is smaller than the one used by Dalal [7]. Figure 12 shows the performance of different cell sizes with $\rho \times \rho$ pixels tested. The size $\rho = 3$, where the vector dimension is $d = 2\ 128$ with $\eta_d = 4$, has a better performance in the MFScts space with a miss rate of 42.3 %. By switching to $\rho = 4$, where the feature vector dimension is $d = 1\ 120$, performance drops by 1 %. For the MFSgray space, the performance for $\rho = 3$ is 3 % better than for $\rho = 4$. These results prove that

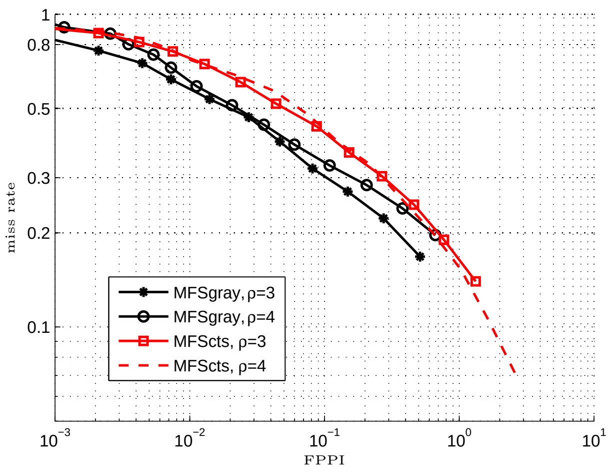


Figure 12: The figure shows the FPPI effect of varying the size of the R-HO2L cells, with values $N = 80$ and $\delta = 1$ for MFSgray, and $N = 48$ and $\delta = 2$ for MFScts.

the smaller size of ρ represents a finer coding of the sample and improves the performance.

3.3.3. Increasing the Training Positive Dataset

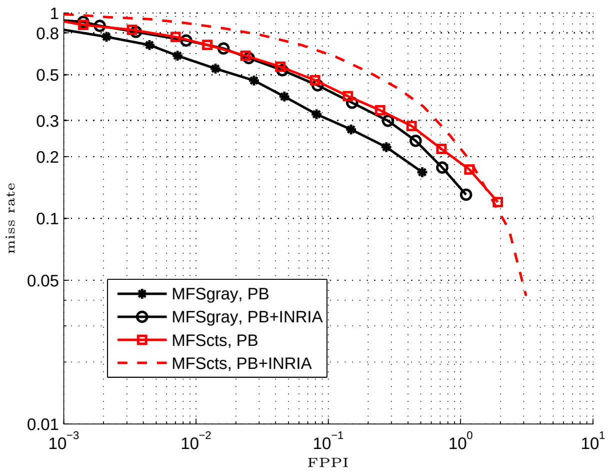


Figure 13: FPPI effect of the number of positives in the training. PB means the original positive sample base from the street sequence, and BP+INRIA means that the INRIA positive samples were added for training.

Figure 13 shows classifiers that were trained by adding 2 417 cropped positive examples of training available from the INRIA Person dataset. Although, in general, the system performance should improve when adding positive samples during the training [2], results are not better due to the fact that the descriptors of these samples are not calculated in a movement space (they are still photos). In addition, the classifier does not appropriately generalize the pedestrian class, and the performance is decreased by 12 % for both spaces.

3.4. Processing Time

This section discusses the processing time of the various steps in the detection execution, using an Intel Core processor i5 @ 2.67 Ghz.

First, we assess the computation time to obtain the MFS in table 1. As developed in section 2.1 level lines are calculated in a loop of N iterations. A threshold is applied at each iteration to get the level sets and, subsequently, their level lines. The greater N is, the more iterations are needed, thus increasing the processing time. Appendix B proposes a fast approximate calculation of the level lines using a series of filters, being independent of N.

The processing time also depends on the number of orientations of level lines η. As we stated in section 2.1.2, the MFS reference R_t is an array having η layers. Increasing η implies that more layers make up array R_t, then more processing time is required to compare the novelties (motion).

η	MFSgray		MFScts	
	Iterative MFS	Fast MFS	Iterative MFS	Fast MFS
4	489	133	708	247
6	523	173	753	295

Table 1: MFS Processing time in milliseconds.

After calculating the MFS, O_t and D_t arrays are used to obtain the histogram integral. This processing also depends on η: 10 msecs @ $\eta_d = 4$ and 15 msecs @ $\eta_d = 6$.

Table 2 details the average processing times for two cases: frames with a single pedestrian, and a frame with 11 pedestrians in the view, representing the longest processing time in detection. In the case of frames with very few pedestrians there is no difference in the processing time of the cascade operation. However, when the number of pedestrians is significant, the number of validated hypotheses is very high and they are all evaluated between stages 20 and 30 of the cascade, thus considerably increasing the processing time.

The validation stage, yields an important difference between both cases, and different values of η_d. An SVM linear classifier is calculated faster with fewer orientations.

Detection task	1 pedestrian				11 pedestrians			
	$\eta = 4$		$\eta = 6$		$\eta = 4$		$\eta = 6$	
	20 stg	30 stg	20 stg	30 stg	20 stg	30 stg	20 stg	30 stg
HYp. generation	16	20	31	41	125	170	187	320
HYp. validation	14		14		50		85	

Table 2: Executing time of the detection in milliseconds.

Table 3 shows the accumulated processing times for both examples. When there is no motion in the scene, the pre-processing speed is fixed in the MFS calculation, and the histogram integral. In that case, the cascade evaluates the image in less than 10 milliseconds, meaning that the total time is about 154 milliseconds (Fast MFSgray and $\eta_d = 4$). The global processing system IS between 2 and 6 fps for sequences of 640 x 480 pixels, depending on the content of the frame, the number of orientations and the number of stages in the cascade.

Space	1 pedestrian		11 pedestrians	
	$\eta = 4$	$\eta = 6$	$\eta = 4$	$\eta = 6$
MFSgray	175	218	315	433
MFScts	289	340	429	560

Table 3: Overall processing time for each case in milliseconds.

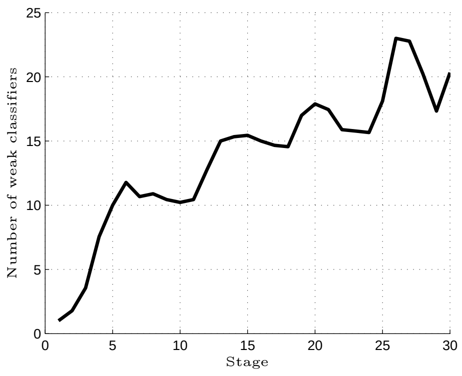
4. System Results

4.1. Cascade Analysis

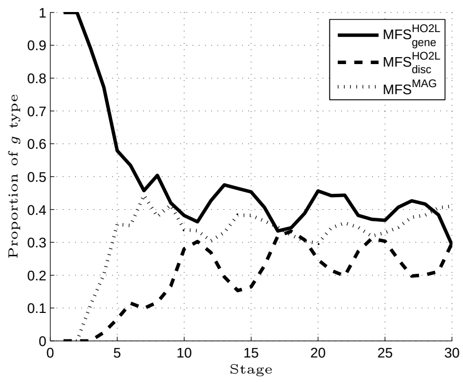
Figure 14 shows the average values of cascade components for three independent trained cascades using the MFSgray space. Fig. 14(a) describes the number of weak functions g at each stage of the cascade. The early stages consists of very few classification functions, and the last stages have a bigger number. The type of features at each stage of the cascade is shown on Fig. 14(b). Until stage four, almost all the classification functions are generative. Discriminant functions increase in number to reach 40-60 % of the total g functions from stage 10. They were chosen by the Adaboost algorithm to better discriminate pedestrian classes with harder negative samples.

Figure 15 analyzes the behavior of the cascade, discriminating frames without motion, with moving objects other than pedestrians, and frames with pedestrians. In the sliding windows approach, the total number of evaluated bounding boxes for each capture in the test sequences is around 10 000. The curves in Fig. 15 show the ratio of the bounding boxes validated by the stage of the cascade on the MFSgray space. When there are no moving objects, the validated bounding boxes with information caused by either noise or camera movements, are totally eliminated at stage 3. In the case of frames without pedestrians, but with other moving objects (vehicles), the ratio of validated bounding boxes is one order below that of the images with pedestrians. These fewer bounding boxes will be eliminated by the HV step. Note that there is no difference in the validated bounding boxes between the 20th and the 30th stages.

Figure 16 shows the first two stages of the boosted cascade. Both stages have only one generative classification function MFS_{gen}^{HO2L}. Both histograms obtained within the patch associated with the feature and histogram model for calculating the classification function can be seen in the figure. As an example, based on input x corresponding to the pedestrian's case, we calculate the



(a) Number of weak classifiers per stage



(b) Proportion of the classification function type on each stage

Figure 14: Description of each stage in the cascade.

Bhattacharyya distances, allowing us to obtain the value of the classification function that is compared to the thresholds of the stage:

1. Stg $G_1(x)$
   - compute the histogram within the path of feature 1946, $\mathbf{h}_{1946}$,
   - compute the Bhattacharyya distance with the model of feature 1946: $d(\mathbf{h}_{1946}(x), \mathbf{m}_{1946}) = 0.079679$,
   - get the strong classifier output: $G_1(x) = g_1(d) = 0.65611$,
   - compare G_1 with the stage threshold TH_1: $G_1(x) > TH_1 = -0.0459$, and validate the sample to be evaluated for the second stage of the cascade.
2. Stg $G_2(x)$,
   - compute the histogram within the path of feature 1351, $\mathbf{h}_{1351}$,

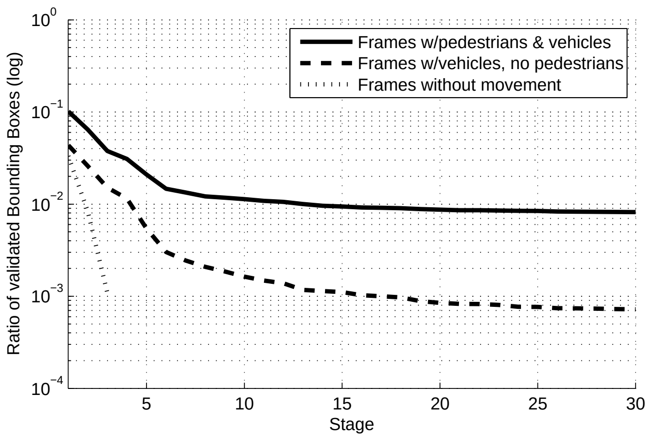
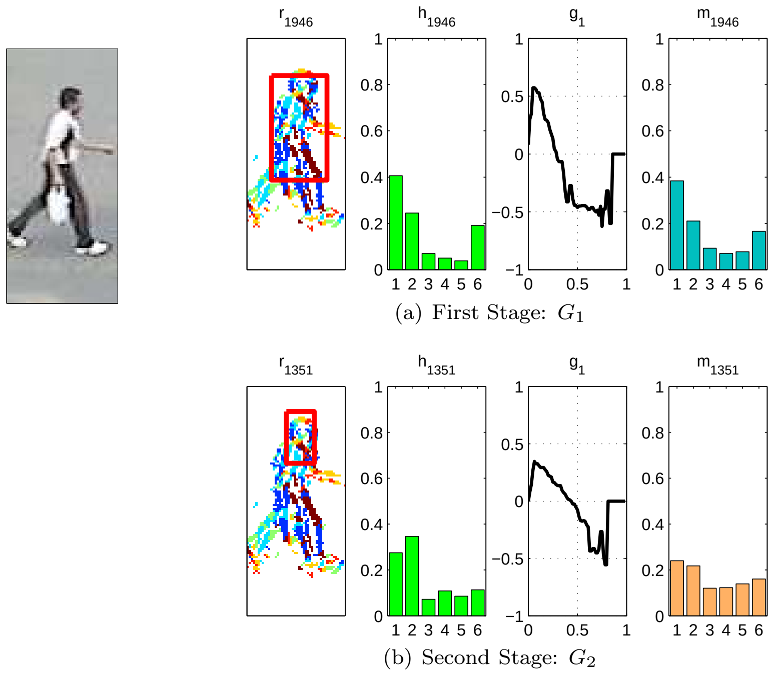


Figure 15: Bounding box validation at each stage of the cascade.



(a) First Stage: G_1

(b) Second Stage: G_2

Figure 16: First two stages of the cascade of classifiers and the histogram features calculated in a pedestrian example.

- compute the Bhattacharyya distance with the model of feature 1351: $d(\mathbf{h}_{1351}(x), \mathbf{m}_{1351}) = 0.12889$,
- get the strong classifier output:

$G_2(x) = g_1(d) = 0.2879$,

- compare G_2 with the stage threshold TH_2:
  $G_2(x) > TH_2 = -0.19159$, and validate the sample to be evaluated for the third stage of the cascade.

In the example, both strong classifier outputs $G_k(x)$ were larger than the corresponding threshold TH_k. Then, the bounding box is delivered to the subsequent stages in the cascade.

4.2. Cascade Results

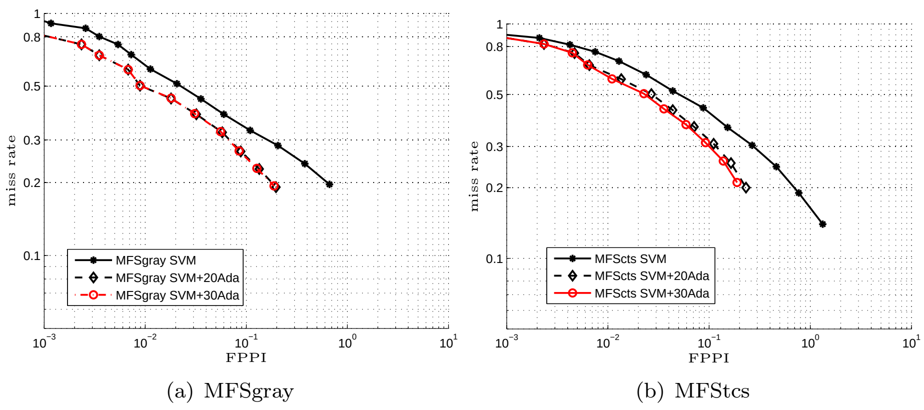


Figure 17: FPPI performance of the overall system.

Figure 17 shows the FPPI system performance comparing detection without HG step (only applying the SVM classifier), and the performance of the system using two Adaboost cascades with different numbers of stages. The combination of the cascade and the SVM classifier (HG+HV) improves the behavior of the system, eliminating a greater number of non pedestrian bounding boxes. This can be seen in a translation of the curve to the left, minimizing false positives. A less desired effect of the hypothesis generation occurs when pedestrians are not detected by the cascade. In the case of MFSgray space, no pedestrians are missed at this point. On the other hand, for MFScts space, the HG misses 6 % of positives.

For MFSgray, the performance of the HG+HV system has a miss rate of 25.5 % at 10^{-1} FPPI for the Adaboost with 30 stages, and a miss rate of 25.8 % for the cascade with 20 stages. Using the HG as a previous step in the detection system improves the performance by 8 %. That means that the Adaboost cascade, as it is a non-linear classifier, provides an important number of hypotheses, while eliminating some negatives that the linear SVM classifier would validate. As shown on Fig. 1, the negative hypotheses are handled by the linear SVM in a higher classification space, thus increasing the probability to eliminate them.

In the case of MFScts space, the lower miss rate of 30.2 % is obtained by the HG+HV combination, using a cascade of 30 stages. The improvement of the proposed system accounts for 12 % in the miss rate, compared to the performance of MFScts without the HG step.

The results obtained are comparable with the best performances achieved by Dollar [4], using datasets with higher resolution. This paper presents the performances using the *log-average*, but as the authors claimed in their work, those values are comparable with the miss rate at 10^{-1} FPPI.

4.3. Overall Results and Benchmarking

The results of the system were compared with two methods of the literature working on still images. First, a linear SVM classifier was trained using the Dalal R-HOG features space. The second methodology is the one proposed by Felzenszwalb [10]. These experiments were conducted using the implementation of OpenCV library [40] for the former, and [42] using the same training dataset as our system for the latter.

The implementation results of the latent SVM are expressed in a Precision-Recall (PR) curve calculated from the set of resulting bounding boxes. The number of correct detections is 1 071 with 4 545 false alarms after the NMS filtering. This value accounts for a false positive per image of 1.06. Since it is not possible to plot the FPPI curve, we take this FPPI point and find the corresponding threshold from the curves of Fig. 17 that allows us to generate the PR curve. If the FPPI curve does not reach a false positive value of 1.06, the threshold generated by the rightmost point of the curve is used. After calculating the PR curve, we can also obtain the average precision value (AP), widely used to compare the performance of detectors.

Figure 18 displays the results in the PR curve of the literature methodologies, and the different classifiers and architectures of our system. The curves with the ‘ADA’ prefix show the results of the system with the hypothesis generation and the hypothesis validation steps.

The latent SVM obtains an AP of 62.2 %, showing a better peformance than Dalals classifiers, which have an AP of 60.8 %. The PR curves of our system have AP values higher than 60 % in most cases. Better performance results are achieved by the ADAMFSgray space (AP=70.4%), and the ADAMFScts spce (AP=68.6%), the former in precision and the latter in recall. As can be seen in the ADAMFScts vs MFScts curves, HG+HV improves the AP values by 10 % and precision up to 20 % for recall values higher than 0.5.

Finally, the systems using the fast calculation of MFS, with the ‘Fast’ prefix, yields results comparable with the normal calculation of level lines, while reaching detection rates between 2 and 6 fps.

5. Conclusions and future work

This paper proposes a pedestrian detection system using video sequences taken from a video surveillance-type fixed camera filming outdoors. Such images

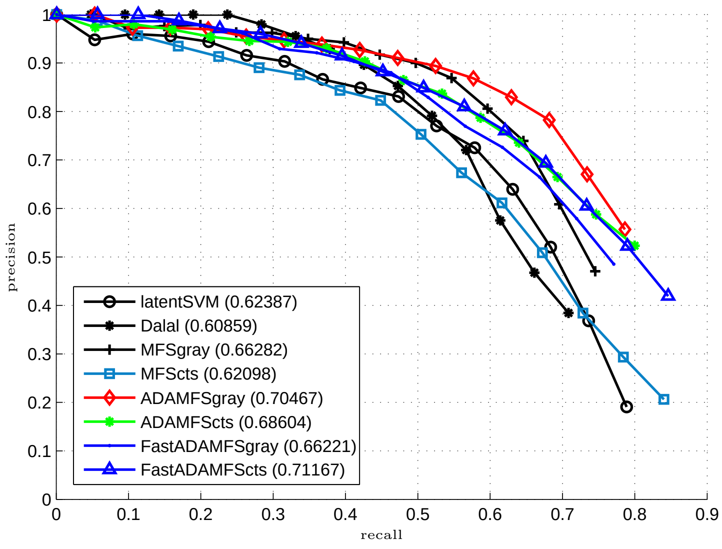


Figure 18: Overall results and benchmarking.

are considered to be significantly complex, as they suffer severe changes in the scene appearance due to weather conditions, rapid changes in lighting, etc., added to the presence of many other moving objects. The system architecture presented consists of a series of stages that first detect movement in the scene, and then check whether the moving object is a pedestrian.

The movement is extracted through a background model methodology based on level lines, which are robust to severe lighting changes, such as those occurring in outdoor images. In addition to the calculation of the level lines in the monochrome image, the use of a color space referred to as Color Texton Space was suggested. This makes it possible to retrieve the color transitions that disappear when the image is converted from the color space to a gray scale. This phase generates the Movement Feature Space (MFS) that will be used by the other stages of the system.

The descriptors generated by the MFS are histograms of oriented level lines (HO2L), encoding level lines orientations and a value similar to the gradient module. This kind of descriptors can be calculated at multiple scales and very quickly through an integral histogram. By means of this pre-preprocessing, and using a cascade of boosted classifiers, the regions with motion in the scene are quickly extracted. The cascade can combine generative classification functions with discriminant functions, accelerating this process. Those image regions with motion and an appearance similar to pedestrians are validated by a linear SVM classifier, and discriminated from other moving objects in the scene, such as cars. SVM classifiers use HO2L descriptors with a configuration similar to the R-HOG.

Using the best combination of the coding parameters of the level lines and the oriented histograms, the system achieved a maximum performance of 25.9 % miss rate with a rate of 10^{-1} false positives per image. The system can operate between 2 and 6 fps, making it a robust and reliable choice for use in real video surveillance applications seeking, for example, the improper presence of pedestrians in forbidden places.

In this paper we have compared the MFS based system with two state of the art methodologies: the linear SVM proposed by Dalal et al [7], and the latent SVM of Felzenszwalb et al [10], trained with our base. Our system yielded similar or better results than these algorithms when applied to our test sequences.

As a perspectives, an analysis of tracking methodologies is being conducted on sequences with sufficient frame rate using the MFS, as well as a combination of pedestrian generative models and tracking algorithms. The combination of these methods would provide a more robust response to the detection and identification of people in the scene, in order to analyze their behavior along the sequence.

6. Acknowledgments

This work was supported by PICT-2283 of ANPCyT, ACyT R11020 of UADE and CONICET (Argentina).

References

[1] T. Moeslund, A. Hilton, V. Krüger, A survey of advances in vision-based human motion capture and analysis, Journal on Computer Vision and Image Understanding 104 (2) (2006) 90–126.

[2] M. Enzweiler, D. M. Gavrila, Monocular pedestrian detection: Survey and experiments, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (12) (2009) 2179–2195.

[3] D. Geronimo, A. L. Lopez, A. D. Sappa, T. Graf, Survey of pedestrian detection for advanced driver assistance systems, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (7) (2010) 1239–1258.

[4] P. Dollr, C. Wojek, B. Schiele, P. Perona, Pedestrian detection: An evaluation of the state of the art, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (4) (2012) 743–761.

[5] J. K. Aggarwal, Q. Cai, Human motion analysis: A review, Computer Vision and Image Understanding 73 (3) (1999) 428–440.

[6] C. Papageorgiou, T. Poggio, A trainable system for object detection, International Journal on Computer Vision 38 (1) (2000) 15–33.

[7] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: IEEE Computer Vision and Pattern Recognition, Vol. 1, 2005, pp. 886–893.

[8] Q. Zhu, S. Avidan, M. Yeh, K. Cheng, Fast human detection using a cascade of histograms of oriented gradients, in: IEEE Conference on Computer Vision and Pattern Recognition, Vol. 2, 2006, pp. 1491–1498.

[9] J. Begard, N. Allezard, P. Sayd, Real-time humans detection in urban scenes, in: Proceedings of the British Machine Vision Conference, 2007, pp. 21.1–21.10.

[10] P. Felzenszwalb, G. Girshick, D. McAllester, D. Ramanan, Object detection with discriminatively trained part-based models, IEEE Pattern Analysis and Machine Intelligence 32 (9) (2010) 1627–1645.

[11] P. Dollr, S. Belongie, P. Perona, The fastest pedestrian detector in the west, in: British Machine Vision Conference, 2010, pp. 1–11.

[12] P. Viola, M. Jones, D. Snow, Detecting pedestrians using patterns of motion and appearance, in: IEEE International Conference on Computer Vision, Vol. 2, 2003, pp. 734–741.

[13] N. Dalal, B. Triggs, S. Schmid, Human detection using oriented histograms of flow and appearance, in: European Conference on Computer Vision, Vol. Part II, 2006, pp. 428–441.

[14] S. Walk, N. Majer, K. Schindler, S. B., New features and insights for pedestrian detection, in: IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 1030–1037.

[15] S. Bouchafa, Motion detection invariant to contrast changes. application to detection abnormal motion in subway corridors, Ph.D. thesis, UPMC Paris VI (1998).

[16] D. Aubert, F. Guichard, S. Bouchafa, Time-scale change detection applied to real-time abnormal stationarity monitoring, Real-Time Imaging 10 (2004) 9–22.

[17] T. Veit, F. Cao, P. Bouthemy, An a contrario decision framework for region-based motion detection, International Journal on Computer Vision 68 (2) (2006) 163–178.

[18] A. Mokhber, C. Achard, M. Milgram, Recognition of human behavior by space-time silhouette characterization, Pattern Recognition Letters 29 (1) (2008) 81–89.

[19] S. Stalder, H. Grabner, L. Gool, Cascaded confidence filtering for improved tracking-by-detection, in: European Conference on Computer Vision, 2010, pp. 369–382.

[20] A. Descamps, C. Carincotte, B. Gosselin, Person detection for indoor video-surveillance using spatio-temporal integral features, in: Interactive Human Behavior Analysis in Open or Public Spaces Workshop, 2011, pp. 110–118.

[21] D. Gravila, S. Munder, Multi-cue pedestrian detection and tracking from a moving vehicle, International Journal on Computer Vision 73 (1) (2007) 41–59.

[22] M. Bertozzi, A. Broggi, M. D. Rose, M. Felisa, A. Rakotomamonjy, F. Suard, A pedestrian detector using histograms of oriented gradients and a support vector machine classifier, in: Intelligent Transportation Systems Conference, 2007, pp. 143–148.

[23] A. Broggi, P. Cerri, S. Ghidoni, P. Grisleri, H. G. Jung, A new approach to urban pedestrian detection for automatic braking, IEEE Intelligent Transportation Systems Conference 10 (4) (2009) 594–605.

[24] D. Comaniciu, Mean shift: A robust approach toward feature space analysis, IEEE Transactions on pattern analysis and machine intelligence 24 (5) (2002) 603–619.

[25] B. Finkston, accessed on march 2012. [link].
URL `http://www.mathworks.com/matlabcentral/fileexchange/10161-mean-shift-clustering`

[26] S. Alvarez, A. Salvatella, M. Vanrell, X. Otazu, 3d texton spaces for color-texture retrieval, in: Image Analysis and Recognition, 2010, pp. 354–363.

[27] P. Negri, X. Clady, S. Hanif, L. Prevost, A cascade of boosted generative and discriminative classifiers for vehicle detection, EURASIP JASP 2008 (2008) 1–12.

[28] C. Stauffer, W. Grimson, Adaptive background mixture models for real-time tracking, in: IEEE Conference on Computer Vision and Pattern Recognition, Vol. 2, 1999.

[29] V. Caselles, l. B. Col, J. Morel, Topographic maps and local contrast changes in natural images, International Journal on Computer Vision 33 (1999) 5–27.

[30] B. Jahne, H. Haussecker, P. Geissler (Eds.), Handbook of Computer Vision and Applications, Vol. 2, Academic Press, 1999.

[31] F. Cao, P. Musse, F. Sur, Extracting meaningful curves from images, Journal of Mathematical Imaging and Vision 22 (2005) 1519–181.

[32] M. Gouiffes, B. Zavidovique, A color topographic map based on the dichromatic reflectance model, EURASIP JIVP 2008 (2008) 1–14.

[33] T. Carron, P. Lambert, Color edge detector using jointly hue, saturation, and intensity, in: IEEE International Conference onf Image Processing, Vol. 3, 1994, pp. 977–981.

[34] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: IEEE Converference on Computer Vision and Pattern Recognition, Vol. 1, 2001, pp. 511–518.

[35] M. Enzweiler, D. M. Gavrila, A mixed generative-discriminative framework for pedestrian classification, in: IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.

[36] R. Schapire, Y. Singer, Improved boosting algorithms using confidence-rated predictions, Machine Learning 37 (3) (1999) 297–336.

[37] T. Kailath, The divergence and bhattacharyya distance measures in signal selection, IEEE Transactions on Communications 15 (1) (1967) 52–60.

[38] V. Vapnik, The nature of Statistical Learning Theory, Springer, 1995.

[39] B. Schölkopf, A. Smola, Learning with Kernels. Suppor Vector Machines, Regularization, Optimization, and Beyond, MIT Press, Cambridge, MA, 2002.

[40] Ver. 2.4.2 (available on december 2012). [link].
URL `http://opencv.willowgarage.com/wiki`

[41] M. Everingham, L. Gool, C. K. Williams, J. Winn, A. Zisserman, The pascal visual object classes (voc) challenge, International Journal on Computer Vision 8 (2) (2010) 303–338.

[42] Felzenszwalb, ver. 5 (available on december 2012). [link].
URL `http://people.cs.uchicago.edu/~rbg/latent/`

Appendix A. Cascade learning algorithm

Algorithm 1 shows the strong classifier learning using Real Adaboost algorithm [36]. The algorithm input are the positive dataset PB, the negative dataset NB, and a parameter T indicating the number of weak functions to incorporate to $G(x)$.

Algorithm 1 Adaboost strong classifier training: ADATRAIN(PB,NB,T)

```
Require: Set of labeled samples $(x_1, y_1), ..., (x_N, y_N)$ ,
    with $x_i \in PB \cup NB$ and $y_i \in \{-1, +1\}$,
Ensure: Weight distribution $W_i = 1/N$,
1: for $t = 1, ..., T$ do
2:   for all $n = \{MFS^{HO2L}_{gen}, MFS^{HO2L}_{disc}, MFS^{MAG}\}$ do
3:     Train weak learner $g^n_t$ using $W_t$,
4:     Get minimum error $e^n_t$
5:   $g_t = g^m_t \ / \ m = argmin(e^n_t)$,
6:   Update:
       $$W_{t+1}(i) = \frac{W_t(i) \cdot e^{-y_i g_t(x_i)}}{Z_t}$$
     where $Z_t$ is a normalization factor.
7: return $G(x) = \sum_{t=1}^{T} g_t(x)$
```

At each algorithm iteration, the classification functions are computed from the training dataset and the weight distribution W. Between al the g^n_t availables, it is selected the one who gets the minimun error classifying the training dataset. This g^m_t is added to the linear combination of G. This process is repeated until $G(x)$ has T weak classification functions.

The algorithm 2 shows the methodology to obtain the cascade of boosted classifiers $C_{Ada} = \{C_1; C_2; ...; C_M\}$, where M is the number of stages in the cascade. The inputs of the algorithm are the positive dataset BP, and two parameter that will control the learning process at each stage: d_{min}, which is the minimum percentage of detections on the validation dataset, and f_{max} which is the maximum percentage of false alarms accepted for the stage. The d_{min} is usually chosen very high: 99% or more. In the iterative method, the strong classifier learning algorithm 1 returns a classifier G_k. The starting threshold value to validate the samples is zero[1]. While it is not reached d_{min} on the validation dataset, the threshold is decremented. This operation affects directly the false alarms rate of the stage: decreasing TH, the more false alarms will be validated by $G_k(x)$. Once reached d_{min}, the false alarms ratio is calculated on the negative dataset. If this value exceeds f_{max}, a new strong classifier $G_k(x)$ is trained incrementing the number of weak functions n. When both conditions

[1]It would be the case of classifying an entry by the sign returned by $G(x)$: $G(x) = sgn(\sum_{t=1}^{T} g_t(x))$

are reached, or when the number of g exceeds a maximum number, the classifier C_k is incorporated to the cascade.

For the next iteration, a new negative dataset is collected from images without pedestrians, being the false alarms of the new C_{Ada}.

The criteria to stop the algorithm can be one of the followings:

- It was reached an overall false alarms ratio, e.g. $F_{target} = 10^{-7}$,
- The cascade is composed by a maximum number of stages in the cascade,
- If the number of false alarms founded to train the next stage is less than a minimum number of samples.

Algorithm 2 Cascade of boosted classifiers

```
Require: $f_{max}$, $d_{min}$
Ensure: Set of positive samples PB,
    Set of negative samples NB,
    $F_0 = 1.0$,
    $i = 0$,$C_{Ada} \leftarrow empty$,
 1: while StopCriteria(i)=FALSE do
 2:    $i = i + 1$,
 3:    $n_i = 0$;$F_i = F_{i-1}$,
 4:    while $f > f_{max}$ do
 5:       $n_i = n_i + 1$,
 6:       $C_i$ = ADATRAIN(PB,N,$n_i$)
 7:       Set threshold $TH_i = 0$,
 8:       Evaluate $C_i$ on validation set to get positive rate $d$,
 9:       Decrease $TH_i$ until $C_i$ has detection rate of at least $d_{min}$
10:       Obtain false alarms ratio $f$ in NB,
11:    $N \leftarrow 0$,
12:    $C_{Ada} = \{C_{Ada}; C_i\}$
13:    Use $C_{Ada}$ to get a set of negative samples NB,
14:    $F_i = f \times F_{i-1}$
15: return $C_{Ada}$
```

Appendix B. MFS fast calculation

In order to reduce the processing time, we propose a methodology to calculate an approximation of the oriented level lines. We rely on the local result of the calculation of the level lines, and the values of the neighbors. The level line which passes through a pixel will be generated by the greater transition between him and one of the 4 neighbors. This operation is done by the convolution of the image with four derivative filters.

$$f_{d1} = \begin{bmatrix} -1 & 1 \end{bmatrix} \quad f_{d2} = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \quad f_{d3} = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \quad f_{d2} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

Let be I a one channel input image, four matrix are obtained convolving I with filters f_{di}:

$$C_i(x) = |I(x) * f_{di}|_{i=1,\ldots,4}$$

The approximated value of $S(x)$ at each pixel x is founded choosing the largest value among the convolved matrix $C_i(x)$:

$$S_a(x) = C_m(x)_{m=argmax\{C_i(x),i=1,\ldots,4\}}$$

The values of $O(x)$ arise from the orientations of the gradient calculated in I with a Sobel filtering.

To retrieve the relevant values of S_a, we kept those pixels x with a value exceeding the threshold γ, which is calculated as:

$$\gamma = \frac{\delta(max(I) - min(I))}{N}$$

where $max(I)$ and $min(I)$ are the maximum and minimum values, in gray levels, of the image I. The parameters N and δ are those introduced in section 2.1.