Analysis of Pixel-Level Algorithms for Video Surveillance Applications

Luigi Di Stefano, Giovanni Neri, Enrico Viarani
DEIS - Department of Electronics Computer Science and Systems, University of Bologna
Via Risorgimento, 2 - 40136 Bologna, ITALY
{ldistefano, gneri, eviarani}@deis.unibo.it
http://labvisione.deis.unibo.it

Abstract

We propose a classification for a set of pixel-level algorithms employed in video surveillance applications and define a performance evaluation metric, based on an analysis of experimental data, for comparing the addressed algorithms. The results of such a comparison are presented and discussed. The set of algorithms considered in this work comprises several algorithms widely known in literature.

1. Introduction

Video Surveillance (VS, from now on) is among the most important application fields for Computer Vision and Pattern Recognition.

A remarkable deal of work have been done aimed at developing "special-purpose" VS-systems chasing high-level assessment tasks (a survey [6], traffic data collection and monitoring [4] [5], gesture recognition of people exchanging objects in public areas [9]). General-purpose VS-systems are, instead, more versatile thanks to their independence from scene knowledge. This makes general-purpose VS-systems easier to configure and suitable to meet typical VS needs so far delegated to humans.

The structure of a general-purpose VS-system, shown in Figure 1, can be thought as a hierarchy of three levels: the Pixel-Processing Level, the Frame-Processing Level and the Tracking-Processing Level. The first level mainly distinguishes objects from background, classifying all the image pixels; it typically embodies a set of noise filters for the most common noise sources, which will be referred to as "Typical Problems" in the reminder of the paper. At the Frame-Processing Level the VS-system joins together foreground pixels belonging to the same object and also, based for instance on proximity or speed criteria, may attempt to group blobs belonghing to object parts which have been recovered as non connected regions by the lower processing level. In the Frame-Processing level it is often present a size-filtering stage aimed at cutting small blobs, probably coming from noise. The Tracking Level tracks entire objects managing events like merges, splits, occlusions, sudden motion of a background object or foreground objects becoming motionless; it also determines objects features like size, speed and direction, interactions, gestures (if objects are humans). If the scene is outdoor, the tracking level can keep count of the daytime light cycle in order to manage shadows and optimise parameters following the changes due to different lighting conditions. The two higher levels can also provide feedback information for the underlying ones in order to set properly their parameters.

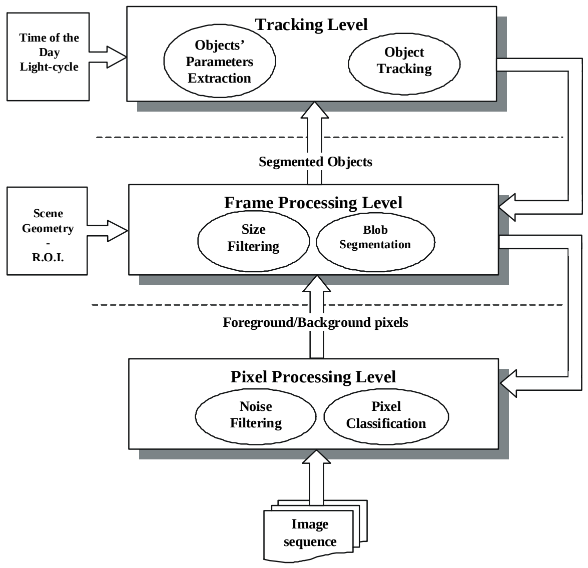

Figure 1. Hierarchical Structure of a VS-system

However, many relatively-simple and general-purpose VS-systems rely on the Pixel-Processing level only, since they perform just motion or change-detection in selected areas, in order to give simple triggers to the human analysis of the incoming video sequence [10] (instead of giving an interpretation of what is happening on the scene). These systems allow to manage multiple on-line events as well as off-line event viewing, thanks to automatic video recording triggered by the change-detection algorithm.

For both, level-structured and simple, general-purpose VS-systems the result of the Pixel-Level processing conditions the performance of the whole system. The higher is the reliability of the pixel classification stage, the better will be the quality, the speed and the versatility of the VS-system. For instance, a noisy pixel classification will decrease the speed of the Frame-Level processing stage which will result saturated due to noise filtering, or will over-trigger the user.

Among the most important "Typical Problems" for the Pixel-Level stage there are: changes in scene illumination (for instance, clouds hiding the sun), background instability due to motion (e.g. waving trees), object camouflage over background (objects with the same colour as the background), foreground objects detection aperture due to homogeneously coloured objects [14]. Whenever unprotected against such a noise sources, VS-systems miss foreground pixels or classify as "foreground" pixels belonging to the background, compromising the quality of the whole system.

The goal of this work is to analyse and compare a selected set of Pixel-Level algorithms chosen among those suitable to general-purpose VS-systems. Some of them are widely known in literature while some others, which will be described in more detail, have been developed by the authors in the context of this research. In order to carry out the comparison we propose a classification of the considered algorithms and define a performance evaluation metric based on the analysis of experimental data. These come from the analysis of a set of benchmark sequences showing most of the Typical Problems for Pixel-Level algorithms.

2. Pixel level algorithms

Pixel-Level algorithms (found in [1], [6], [7], [8], [9], [12], [13], [15]) can be classified into the two major classes shown in Figure 2. Background Algorithms use a background reference to classify pixels, while Frame-Differencing Algorithms extract foreground pixels by comparing the current frame with previous frame(s).

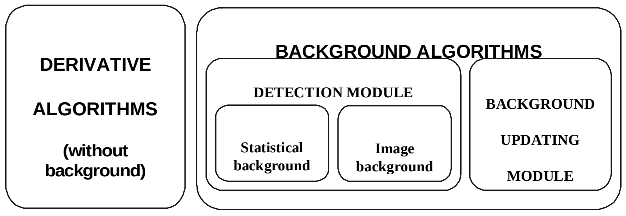

Figure 2. Classification of Pixel-Level algorithms

In the initial phase of our work aimed at evaluating Pixel-Level algorithms we carried-out a "Pre-Selection" step by implementing several algorithms, collecting experimental data obtained in benchmark sequences and comparing the algorithms belonging to the same class on the basis of the performance evaluation metric which will be shown in the next Section. This work, described in [16] and not completely reported here for the sake of brevity, was basically aimed at finding out the "best representative" of each class/module: as shown in Figure 3, we have chosen a set of algorithms per class based on the criteria that the set must include at least two algorithms covering the typical "class-behaviour" under critical conditions. Following the scheme of Figure 3, we describe now the classes and the algorithms.

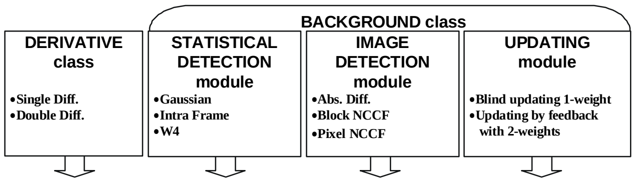

Figure 3. Algorithms considered for the "pre-selection" step

The Derivative Algorithms do not have any background reference to be updated and produce their output through one or more differences between contiguous frames, thus they are robust against light changes but present lacks of sensitivity. The "Double Difference" computes two thresholded differences between three frames: *abs[F(t-2)-F(t-1)]* and *abs[F(t-2)-F(t)]* [3], [17]. The two binary images resulting from the previous operations are AND-ed to obtain the final output.

Background Algorithms typically include a "Detection Module" and a "Background Maintenance Module". The former classifies pixels by comparing the current frame against the Background, the latter keeps the Background constantly updated. The Background can be either an image or a set of statistical parameters. Most algorithms that use a reference need a training period to set up the reference. A training period is made out of a certain number of frames in which no objects are present in the scene and the background changes only as result of various sources of noise. The algorithms with Statistical Detection module are generally thought to recognise and filter out the background instabilities learning the pixels variability during a training period. This yields a sensitivity improvement respect to the derivative class. However, excessive sensitivity can causes saturation or also a diffused blindness that determines a poor detection; these drawbacks are mainly due to the training conditions. We have implemented a generic Gaussian Model based algorithm (derived from [13] and [11]) that assumes a gaussian distribution for background noise and the "W4" algorithm, that is used as Pixel-Level algorithm in the "Hydra" VS-system [7]. W4 is conceived to reduce the blindness, typical of statistical algorithms, when the pixels variation range is wide during the training period. When this happens W4 sets a less selective thresholding based on the hypothesis of a bimodal distribution for noise. It computes three statistical parameters per each pixel during the training period. We found that these parameters characterise very well the typical pixel variability as long as the working conditions are relatively similar to the training conditions. In such a context, "W4" filters-out effectively the background instability, preserving a discrete sensitivity for zones affected by noise types like waving objects moved by the wind, blinking lights or flickering monitors. Finally, as a variation of the previous two, we have implemented an algorithm called “Intra Frame” that computes as noise thresholds the maximum and minimum differences among two frames for a pixel during the training period. It has been ceonceived to patch the excessive sensitivity of Gaussian Model and to get faster processing speed than W4. The algorithms with Detection Module and based on a Background Image are probably the most widely known. Among these, we have chosen the Absolute Difference because of its simplicity and velocity in computation that grants a good point of view for a class features estimation. Taking ideas from the Block Matching Algorithm [2], we have implemented two versions of the correlation between the current image and the background, thus trying to solve the typical Absolute Difference problem, i.e. the excessive sensitivity to light changes. As known, indeed, the cross correlation is quite independent of “global” light changes [5]. Thus we have implemented two correlation-based algorithms: in the algorithm referred to as Block NCCF the image is partitioned in square blocks and the foreground blocks are detected by thresholding the NCCF (Normalised Cross Correlation Function) between the current and the background images. The second algorithm, referred to as Pixel NCCF, allows for a higher resolution detection since the NCC among the current image and the background is computed on a pixel-basis (i.e. each pixel in the current image is the center of a block to be matched with a corresponding background block).

As far as the Background Updating Module is concerned, we considered a simple constant-weight updating referred to as “1-Weight” (or "blind" updating, since it does not take into account the pixel classification performed by the detection module [5]) and an updating strategy, referred to as “2-Weights”, which instead is based on exploiting a feedback from the detection module to the background updating module, as shown in Figure 4.

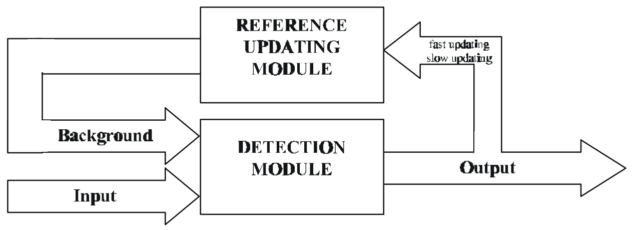

Figure 4. Output feedback in the background updating

Basically, there are two weights that determines how fast the intensity value (or the statistics) of a pixel in the current image will contribute to the updating of the background: if, in the current frame the pixel has been classified as foreground, the slower weight will be chosen while, if it has been classified as background, the fast weight will causes a fast inclusion of the contribution associated with the pixel into the reference. Since misclassifications can happen, the two weights are necessary and cannot be replaced by a trivial "full-update/don't update" policy: the fast-but-not-immediate weight act as low-pass filter, protecting background from noisy pixel-transient updating; the non-zero updating for foreground, allows for the recovering of foreground misclassifications. The pixel value for the actual background (time t) will be $PixelBackground_t(x,y) = PixelImage_{t-1}(x,y) \cdot Weight(x,y) + PixelBackground_{t-1}(x,y) \cdot [1-Weight(x,y)]$, where *Weight(x,y)* will vary as:

$$Weight(x,y) = \begin{cases} FastWeight,\ Pixel(x,y)_t \in background \\ SlowWeight,\ Pixel(x,y)_t \in\ foreground \end{cases}$$

This approach can be applied also to a Statistical Background. In such a case, the two weights will determine how many past frames will be considered for the computation of the statistical parameters: if we are updating a background pixel, so as for the fast weight, we will consider only the few recent entries for the statistic, in order to let the current entry to heavily condition the global parameters of the whole statistic (i.e. fast updating of the statistics). Conversely, if a pixel is classified as foreground, its actual data will smoothly affect the statistical reference parameters by considering a population made out of several old frames (i.e. slow updating of the statistics).

3. Performance evaluation

Since all Pixel-Level algorithms produce a binary image as output we have devised a performance metric which allows to evaluate frame by frame whether a pixel has been classified correctly. To do this, after selecting the frame to be analysed, we generate by-hand a ground-truth mask corresponding to the correct pixel classification (Figure 5).

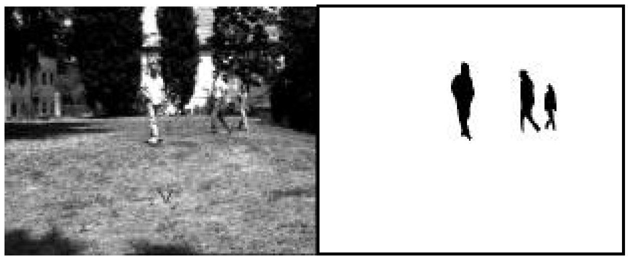

Figure 5. An image and its ground-truth

The software implementing the considered low-level algorithms contains also several functions aimed at counting the pixels which appear as misclassified with respect to a given ground-truth mask. These functions allow for counting separately false positives (i.e. background pixels erroneously classified as foreground) and false negatives (i.e. background pixels erroneously classified as foreground).

As a general principle, we choose a metric for classification errors that accounts for misclassifications relatively to the size of a Region Of Interest:

$$PixelError\% = \frac{MisclassifiedPixels}{ROIsize} \cdot 100$$

Following this principle we can define the errors for False Positives and Negatives:

$$FPerror\% = \frac{FPpixels}{B} \cdot 100 \text{ , } FNerror\% = \frac{FNpixels}{M} \cdot 100$$

where *B* is the background size and *M* (mask) is the foreground size (both in pixels). Extending the ROI to the entire image we define the Absolute Error:

$$AbsoluteError\% = \frac{MisclassifiedPixels}{imageSize} \cdot 100$$

Substituting the *FP* and *FN* errors % we obtain:

$$AbsoluteError\% = FPerror\% \cdot \frac{B}{B+M} + FNerror\% \cdot \frac{M}{B+M}$$

$$= \frac{FPpixels + FNpixels}{imageSize} \cdot 100$$

where the full image size is *imageSize = B+M*.

The *AbsoluteError%* gives an "exact" measure of how good is the algorithm performance, but it accounts rather poorly for the human visual perception of the binary output resulting from a given classification. Investigating for this mismatch, we found that, in some images, foreground objects are very small compared to the image size, while, in some others, foreground objects dominate the image area. In these cases the human visual judgement is hugely biased by the performance of the algorithm into the biggest region. To adopt a metric which follows more closely the human visual perception we have defined the *VisualError%* which gives more relevance to the False Positives if objects are small in the image, or more relevance to False Negatives if objects are big

$$VisualError\% = \frac{(FNpixels + FPpixels)}{2 \cdot \min\{M; B\}} \cdot 100$$

The following relation among the two defined errors shows that they substantially differ by a factor which is always>1, i.e. the *VisualError%* amplifies the *AbsoluteError%* as a function of the difference between object and background sizes:

$$VisualError\% = \frac{Min + Max}{2 \cdot Min} AbsoluteError\%$$

$$Min = smallest\{M; B\}$$

$$Max = greatest\{M; B\}$$

Figure 6 shows two different elaboration for the source image of Figure 5, where objects are small respect to the image size:

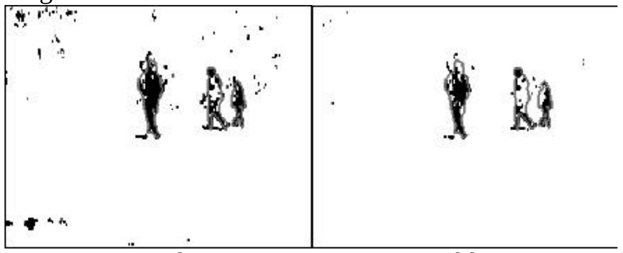

Figure 6. Examples of two outputs for the image of Figure 5

It is clear that Figure 6-(a) is much noisier than Figure 6-(b) but this is not evident looking at the absolute errors in Table 1 that differs only by 0.7%: conversely, a better description of the human visual perception is given by the visual errors, where the difference grows to 13%.

	Figure 6-**(a)**	Figure 6-**(b)**
Absolute Error%	2,2%	1,5%
Visual Error%	41,7%	28,7%

Table 1. Absolute and Visual errors for the previous outputs

We have recorded a set of 15 benchmark sequences reproducing several combinations of the Typical Problems considered in Section 1. To obtain the experimental data reported in this paper we ran our algorithms over entire sequences, selecting significant frames containing specific Typical Problems for the data measurements.

4. Experimental results

As introduced in Section 2, measurements has been carried out in two steps: first we performed a Preliminary Selection comparing algorithms belonging to the same class, then we have compared the "winners" in a Final Comparison. For the first phase we choose sequences containing Typical Problems particularly harmful for the considered class in order to put in evidence the qualities of each algorithm. For the Final Comparison we have used a sequence containing an as wide as possible variety of Typical Problems, in order to find out the better algorithm in general terms.

The Final Comparison scheme between the classes (described in Section 2) is shown in Figure 7.

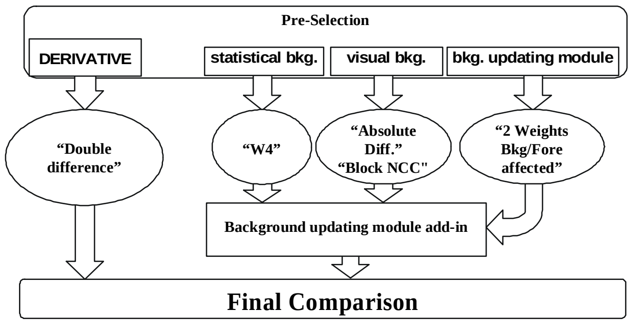

Figure 7. Scheme for the final comparison

Concerning the Pre-Selection, as for the Derivative Algorithms, we found that the best performance on the benchmark sequence were obtained with "Double Difference" algorithm, the best Detection Module in case of Statistical Reference was the algorithm referred to as "W4", the best Detection Modules in case of Visual Reference were the "Absolute Difference" and "Block NCC" algorithms and the best Reference Updating Module was the algorithm referred to as "2 Weights" (more in [16]).

For the Final Comparison we choose a single sequence, "Parco Sc", (length = 500 frames, frame size = 768 x 576 pixel, grey levels = 256, frame-rate=12 Hz, initial training = 119 frames). This sequence has been chosen because of its completeness in containing Typical Problems: light changes when objects are in motion, background instability, object camouflage, and foreground aperture. To obtain measurements we have selected significant frames containing Typical Problems: we choose frame number 135 for camouflage; 170, 215, 271 for background instability and a smooth light change; 290, 310, 323 for background instability and a huge 60 grey-levels light change between 290 and 323; 365 for foreground aperture; 423 for background restoring evaluation after background instability.

We present here the results of the Final Comparison between the Pre-Selection best-representatives resulting from the Pre-Selection. The measurements are based on the performance evaluation metric presented in Section 3 and follow the scheme shown in Figure 7. Each frame has been processed by four algorithms: "Double Difference", "W4" with "2 weights updating", "Absolute Difference" with "2 weights updating" and "Block NCC" with "2 weights updating".

Per each processing we have calculated the errors: False Positive%, False Negative%, Absolute Error% and Visual Error%.

FPerror%	**Frames of "Parco Sc" file sequence**								
	135	**170**	**215**	**271**	**290**	**310**	**323**	**365**	**423**
Double Difference	**0%**	**0,004%**	**0,02%**	**0,06%**	0,083%	**0,34%**	**0,48%**	**1,51%**	**0,039%**
W4	0,091%	0,335%	0,345%	0,812%	1,2%	3,43%	7,1%	22,4%	21%
Absolute Difference	0,004%	0,101%	0,090%	0,068%	**0,073%**	0,383%	0,586%	2,11%	0,795%
Block Based NCCF	0,028%	0,361%	0,270%	0,274%	0,240%	0,664%	1,003%	3,051%	1,113%

Table 2. False Positives errors in source frames per selected algorithms

Table 2 shows how the "Double Difference" output has the lowest False Positive noise quantity but, in Table 3, the same algorithm has the worst score for False Negatives, the best algorithm being in this case "W4" with "2 weights updating". For this reason we reject the "Double Difference".

FNerror%	**Frames of "Parco Sc" file sequence**								
	135	**170**	**215**	**271**	**290**	**310**	**323**	**365**	**423**
Double Difference	93,1%	77,9%	60,5%	60,2%	64,2%	32,5%	30,7%	34,7%	72,2%
W4	**28,4%**	**21,4%**	**11,6%**	**21,5%**	**19,5%**	**14,7%**	**12,5%**	**13,4%**	**9,3%**
Absolute Difference	64,6%	47,9%	32,8%	53,8%	40,1%	36,5%	26,9%	28,4%	40,5%
Block Based NCCF	68,3%	35,4%	11,8%	38,3%	32,7%	16,6%	16,6%	35,2%	27,6%

Table 3. False Negatives errors in source frames per selected algorithms

Absolute Error%	**Frames of "Parco Sc" file sequence**								
	135	**170**	**215**	**271**	**290**	**310**	**323**	**365**	**423**
Double Difference	0,348%	0,227%	0,337%	1,033%	1,52%	1,22%	1,52%	5,4%	1,06%
W4	**0,197%**	**0,409%**	0,41%	1,15%	1,61%	3,74%	7,29%	21,3%	20,8%
Absolute Difference	0,245%	0,269%	**0,28%**	0,938%	0,969%	1,37%	**1,49%**	**5,2%**	**1,35%**
Block Based NCCF	0,283%	0,484%	0,337%	**0,89%**	**0,968%**	**1,1%**	1,54%	6,8%	1,49%

Table 4. Absolute Error in source frames per selected algorithms

In the global error tables (Table 4 and Table 5) "W4" with "2 weights updating" works fine until the light remains constant while the "Block NCC" with "2 weights updating" shows its major robustness to light changes in central frames, when the light discontinuity is greater. By evaluating the Absolute and Visual errors %, the algorithm which holds the best score for more frames is the "Absolute Difference" with "2 weights updating".

Visual Error%	Frames of "Parco Sc" file sequence								
	135	**170**	**215**	**271**	**290**	**310**	**323**	**365**	**423**
Double Difference	46,5%	**32,4%**	29%	31,9%	33.9%	22,3%	22,1%	22,8%	**35,5%**
W4	**26,3%**	58%	35,5%	35,5%	35.9%	68,2%	106%	89,9%	730%
Absolute Difference	32,7%	38,4%	**24,1%**	**29%**	**21.6%**	25%	**21,6%**	**20,1%**	48%
Block Based NCCF	37,8%	69,1%	30%	27,5%	**21.6%**	**20,6%**	22,3%	28,7%	52,9%

Table 5. Visual Error in source frames per selected algorithms

5. Conclusions

This work proposes a classification for Pixel-Level algorithms suitable for general-purpose VS-systems. The proposed scheme is effective in classifying several algorithm known in literature and has suggested us the right input for developing original and robust modules like the "Absolute Difference" with "2 weights updating". After a Pre-Selection step and the definition of a suitable metric, we have compared the performances of selected Pixel-Level algorithms with respect to their ability to classify pixels into background and foreground. The result of such a comparison suggest, as the best combination of detection and updating modules, the "Absolute Difference" with the "2 weights updating". This evaluation accounts for both false positives and false negatives in a manner that is similar to the human perception of classification errors. The "Absolute Difference" is the simplest among the algorithms that uses a reference and its combination with the "2 weights updating" makes the resulting algorithm quite independent by any scene and objects constraints.

Our future work will be aimed at developing an as general-purpose as possible, rule-based, tracking module to be linked to the Pixel-Level processing module. This will involve further analysis of Pixel-Level algorithms in order to asses whether the performance evaluation metric adopted in this paper and based on human visual perception reflects the behaviour of such algorithms when they are not the unique module of a VS-system but instead act as input for higher-level processing modules.

References

[1] A. Amir, M. Lindenbaum, "Ground From Figure Discrimination" *proceedings Conference on Computer Vision and Pattern Recognition*, Computer Society, 1998,Pages: 521-527

[2] H.G. Musmann, P. Pirsch, H.J. Grallert, "Advances in Picture Coding", *Proceedings of the IEEE*, Vol. 73, No. 4, April 1994, pp. 523-546

[3] H. Ching-Kai, C. Tsuhan, "Motion Activated Video Surveillance Using TI DSP", *Proc. of DSPS Fest '99* Houston Texas, August 4-6, 1999

[4] L. Di Stefano, E. Viarani, "Vehicle Detection and Tracking Using the Block Matching Algorithm", *Recent Advances in Signal Processing and Communications* N. Mastorakis Editor, World Scientific and Engineering Society Press, ISBN 960-8052-03-3, 1999.

[5] L. Di Stefano, I. Milani, E. Viarani, "Evaluation of Inductive-Loop Emulation Algorithms for UTC Systems", *Sixth International Conference on Control, Automation, Robotics and Vision*, ICARCV 2000, 5-8 December 2000, Singapore. ISBN 981-04-3445-6.

[6] D.M. Gavrila "The Visual Analysis of Human Movement: a Survey", *Computer Vision and Image Understanding* Vol. 73, Number 1, January 1999, pp 82-98, 1999

[7] I. Haritaoglu, D. Harwood, L.S. Davis, "W4 Who? When? Where? What? A real time system for detecting and tracking people", *Automatic Face and Gesture Recognition, Proceedings. Third IEEE International Conference on*, 1998.

[8] I. Haritaoglu, D. Harwood, L.S. Davis, "Hydra: Multiple People Detection and Tracking Using Silhouettes", *Image Analysis and Processing, 1999. Proceedings.* International Conference on, Sept. 1999

[9] I. Haritaoglu, R Cutler, D. Harwood, L.S. Davis, "Backpack: Detection of People Carrying Objects Using Silhouettes", *Proceedings of the Seventh IEEE International Conference on Computer Vision*, 1999. Pages: 102 - 107 vol.1

[10] Prescient Systems, "Gotcha! Video surveillance for home or office", http://www.gotchanow.com/

[11] A. N. Rajagopalan, P. Burlina, P. Chellappa, "Higher Order Statistical Learning for Vehicle Detection in Images", *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, 20-27 Sept. 1999, Pages 1204 - 1209 Vol. 2 ISBN: 0-7695-0164-8

[12] G. Rigoll, B. Winterstein, S. Muller, "Robust Person Tracking in Real Scenarios with Non-Stationary Background Using a Statistical Computer Vision Approach", *Second IEEE Workshop on Visual Surveillance*, 1999 (VS'99). Pages: 41 - 47

[13] C. Stauffer, W.E.L Crimson, "Adaptive Background Mixture Models for Real-Time Tracking", *Conference on Computer Vision and Pattern Recognition*, 1999. IEEE Computer Society, Pages: 246 - 252

[14] K. Toyama, J. Krumm, B. Brumitt, B. Meyers, "Wallflower: Principles and Practice of Background Maintenance", *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, 1999. Pages: 255 - 261 vol.1

[15] A. Utsumi, J. Ohya, "Image Segmentation for Human Tracking Using Sequential-Image-Based Hierarchical Adaptation", *Conference on Computer Vision and Pattern Recognition*, Proceedings of IEEE Computer Society, 1998. Pages: 911 - 916

[16] E. Viarani, "Elaborazione di Sequenze di Immagini per Monitoraggio del Traffico e Videosorveglianza", *PhD Thesis*, 2001, 1st March, Bologna Italy

[17] K. Yoshinari, M. Michihito, "A Human Motion Estimation Method Using 3-Successive video frames", *Proc. Of Intern. Conf. On Virtual systems and multimedia* Gifu, 135-140 1996