Moving Object Detection for Real-Time Augmented Reality Applications in a GPGPU

Carlos Cuevas, Daniel Berjón, Francisco Morán, and Narciso García

Abstract — *The last generation of consumer electronic devices is endowed with Augmented Reality (AR) tools. These tools require moving object detection strategies, which should be fast and efficient, to carry out higher level object analysis tasks. We propose a lightweight spatio-temporal-based non-parametric background-foreground modeling strategy in a General Purpose Graphics Processing Unit (GPGPU), which provides real-time high-quality results in a great variety of scenarios and is suitable for AR applications*[1].

Index Terms — Moving object detection, real-time, augmented reality, spatio-temporal non-parametric modeling, GPGPU.

I. INTRODUCTION

The last generation of consumer electronic devices, such as smart-phones, tablets, or home and portable video game consoles, is endowed with Augmented Reality (AR) tools [1]. These tools require new, fast and efficient (i.e. lightweight) computer vision applications [2] where moving object detection is a key step for several high level analysis tasks such as segmentation, tracking, classification, or event detection [3].

In the last years, a large number of moving object segmentation approaches has been proposed in the literature [4]. Some of these algorithms aim to maximize the speed and to reduce the memory requirements [5], providing satisfactory results for short sequences with quasi-stationary backgrounds [6]. However, these strategies are not efficient in critical situations such as abrupt or gradual illumination changes, shadows, noisy sequences, or dynamic backgrounds (containing fountains, flags, trees, sea waves, rain, etc.) [7], and depend on several thresholds that must be manually set by users [8].

To overcome these limitations, many multimodal strategies have been also proposed [9], which are able to model multiple states for each pixel. One key multimodal reference is the Mixture of Gaussians (MoGs) method [10], which uses a combination of several Gaussians to model the variations of each image pixel. Other multimodal techniques [11] use Hidden Markov Models (HMMs) to try and model the background variations by representing the changes in the sequences with different states. However, these strategies have important limitations: MoG-based methods are not flexible enough to model complex background density functions [12], and in HMM-based methods the selection of appropriate models is difficult and the initialization process is complex [13].

More recently, non-parametric kernel density estimation methods have been proposed to improve the quality of the detections in environments where the pixel variations cannot be described with the abovementioned parametric strategies [14]. These techniques do not consider the values of the pixels as a particular distribution, and build a probabilistic representation of the observations using a recent sample of values for each pixel [15], thus providing very high quality results in a large variety of complex scenarios with multimodal backgrounds [16].

To achieve better results in scenarios where moving objects and background have similar characteristics [17], some proposals estimate not only a background density function but also a foreground model [18]. Additionally, to carry out the detection of moving objects in sequences recorded with portable devices (mobiles, handy-cams, etc.), where both the background and foreground objects change their spatial positions over time, other works propose to estimate the background and foreground models from spatio-temporal reference information [19].

Although non-parametric approaches improve the quality of the detections provided by other strategies, they have some limitations that should be considered [20]. Their main drawback is that they require a very large number of computations to be performed [13] and, thus, they incur in very high memory and computational costs [21]. Moreover, these excessive requirements are noticeably increased when, to detect moving objects in sequences recorded with portable cameras, spatio-temporal background and foreground models are estimated. Consequently, if these strategies are computed in typical Central Processing Units (CPUs), they cannot be suitable for AR applications operating in real-time.

However, this important drawback can greatly be solved by porting these strategies to a modern programmable General Purpose Graphics Processing Unit (GPGPU) [22]. These devices are becoming commonplace in regular computers [23] and are even advancing into mobile computing scenarios [24]. They provide very good numeric performance and a high degree of parallelism, being capable of executing simultaneously hundreds or even thousands of threads concurrently [25]. As a consequence, several approaches for image processing that exploit GPGPUs have been proposed in the last few years [26], providing real-time implementations of computer vision applications on recent consumer electronic devices [27].

[1] This work has been partially supported by the Ministerio de Ciencia e Innovación of the Spanish Government under project TEC2010-20412 (Enhanced 3DTV).

Carlos Cuevas, Daniel Berjón, Francisco Morán, and Narciso García are with Grupo de Tratamiento de Imágenes (GTI), Universidad Politécnica de Madrid (UPM), 28040, Madrid, Spain (e-mail: {ccr, dbd, fmb, narciso}@gti.ssr.upm.es).

Manuscript received 01/14/12
Current version published 03/21/12
Electronic version published 03/21/12.

0098 3063/12/$20.00 © 2012 IEEE

In this paper we propose a novel spatio-temporal background-foreground non-parametric modeling strategy for moving object detection in a GPGPU, which provides real-time high-quality results in a great variety of scenarios and, accordingly, can be used in the AR applications required by last generation portable and non-portable consumer electronic devices.

II. NON-PARAMETRIC MODELING

A. Spatio-temporal non-parametric modeling

Let us define a pixel p^n in the current image I^n, at time n, as a $(D+2)$-dimensional vector, $\mathbf{x}^n=((\mathbf{c}^n)^T,(\mathbf{s}^n)^T)^T \in R^{D+2}$, where $\mathbf{c}^n \in R^D$ is a vector containing appearance characteristics of the pixel (color, gradient, depth, etc.) and $\mathbf{s}^n=(h^n,w^n) \in R^2$ is a vector containing its coordinates. The probability density function for both background, β, and foreground, φ, can be estimated by using multidimensional kernels, from $(D+2)$-dimensional spatio-temporal reference samples, from the previous images into a spatial neighborhood around the spatial position of p^n [17].

Once we have modeled both background and foreground, it is possible to evaluate the probability of p^n to belong to the foreground class. Using Bayes' theorem we carry out this evaluation as

$$\Pr(\phi|\mathbf{x}^n)=\frac{\Pr(\phi)p(\mathbf{x}^n|\phi)}{\Pr(\beta)p(\mathbf{x}^n|\beta)+\Pr(\phi)p(\mathbf{x}^n|\phi)}, \quad (1)$$

where $\Pr(\varphi)$ is the prior probability for the foreground, $\Pr(\beta)=1-\Pr(\varphi)$ is the prior probability for the background, and $p(\mathbf{x}^n|\beta)$ and $p(\mathbf{x}^n|\varphi)$ are, respectively, the estimated background and foreground density functions.

B. Background modeling

Let us consider a set of N_β background reference samples, $\mathbf{x}_{\beta,i}=(\mathbf{c}_{\beta,i},\ \mathbf{s}_{\beta,i})$, from the T_β previous images ($T_\beta \le N_\beta$). The probability density function that $\mathbf{x}^n$ belongs to the image background can be estimated non-parametrically [19] as

$$p(\mathbf{x}^n|\beta)=\frac{1}{N_\beta}\sum_{i=1}^{N_\beta}\left|\boldsymbol{\Sigma}_\beta\right|^{-\frac{1}{2}}K\left(\boldsymbol{\Sigma}_\beta^{-\frac{1}{2}}\left(\mathbf{x}^n-\mathbf{x}_{\beta,i}\right)\right), \quad (2)$$

where $\boldsymbol{\Sigma}_\beta$ is a symmetric definite $(D+2)\times(D+2)$ bandwidth matrix that determines the "width" of the kernel K around each sample point. Looking for a trade-off between computational efficiency and quality [28], we have chosen here diagonal bandwidth matrices,

$$\boldsymbol{\Sigma}_\beta=\mathrm{diag}\left(\sigma_{\beta,1}^2,\sigma_{\beta,2}^2\ldots\sigma_{\beta,D}^2,\sigma_{\beta,H}^2,\sigma_{\beta,W}^2\right), \quad (3)$$

where the first D components specify the bandwidth of the appearance components and the last two are the spatial bandwidths (rows and columns) of the kernels.

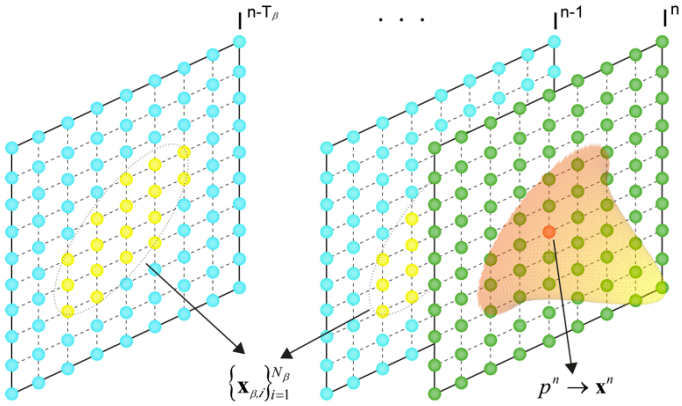

Fig. 1. Reference samples used to estimate the background probability density function.

In this way, applying Gaussian kernels, the background likelihood can be obtained as

$$p(\mathbf{x}^n|\beta)= =\frac{1}{N_\beta(2\pi)^{(D/2+1)}}\sum_{i=1}^{N_\beta}\prod_{j=1}^{D+2}\frac{1}{\left|\boldsymbol{\Sigma}_\beta(j,j)\right|^{\frac{1}{2}}}\exp\left(-\frac{\left(\mathbf{x}^n(j)-\mathbf{x}_{\beta,i}(j)\right)^2}{2\boldsymbol{\Sigma}_\beta(j,j)}\right). \quad (4)$$

Additionally, to prevent the evaluation of reference samples not contributing significantly to the estimation process, we propose to consider exclusively the ones whose spatial distance $(\Delta h, \Delta w)$ to p^n satisfies that

$$\begin{pmatrix}\Delta h & \Delta w\end{pmatrix}\begin{pmatrix}\sigma_{\beta,H}^2 & \sigma_{\beta,H}\sigma_{\beta,W}\\ \sigma_{\beta,H}\sigma_{\beta,W} & \sigma_{\beta,W}^2\end{pmatrix}^{-1}\begin{pmatrix}\Delta h\\ \Delta w\end{pmatrix}\le 3^2. \quad (5)$$

In this way, a reference sample will be not considered if it falls outside the 99% of the spatial Gaussian kernels defined by $\sigma_{\beta,H}$ and $\sigma_{\beta,W}$. Fig. 1 shows an example that relates a pixel in the current image (in red) with the reference samples satisfying equation (5). The considered reference samples are depicted in yellow and the discarded ones are in blue. The spatial Gaussian kernel that is used in the background modeling has been represented on the red pixel in the current image.

C. Foreground modeling

Let p^n be a pixel in the image I^n, defined by the vector $\mathbf{x}^n$ previously described. At first, the probability density function that this pixel belongs to the foreground is uniform. However, if moving objects have been previously detected around p^n, the probability of observing these foreground regions at this pixel will increase. Therefore, the probability density function that $\mathbf{x}^n$ belongs to the image foreground can be estimated as a mixture of a uniform function and a Gaussian kernel density function [19],

$$p(\mathbf{x}^n|\phi)=\alpha\gamma+ +\frac{(1-\alpha)}{N_\phi(2\pi)^{(D/2+1)}}\sum_{i=1}^{N_\phi}\prod_{j=1}^{D+2}\frac{1}{\left|\boldsymbol{\Sigma}_\phi(j,j)\right|^{\frac{1}{2}}}\exp\left(-\frac{\left(\mathbf{x}^n(j)-\mathbf{x}_{\phi,i}(j)\right)^2}{2\boldsymbol{\Sigma}_\phi(j,j)}\right), \quad (6)$$

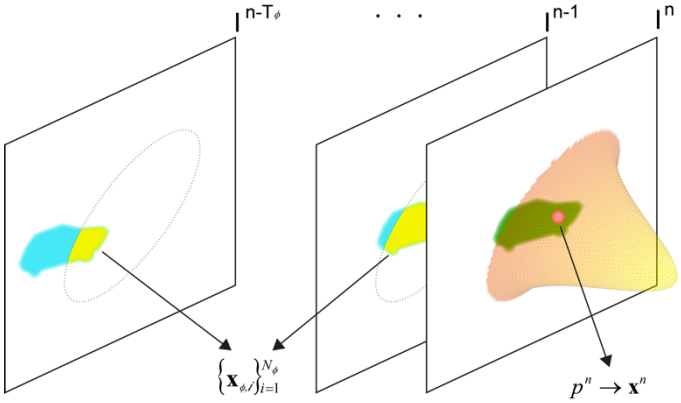

Fig. 2. Reference samples used to estimate the foreground probability density function.

where N_φ is the amount of foreground samples, $\mathbf{x}_{\varphi,i}=(\mathbf{c}_{\varphi,i}, \mathbf{s}_{\varphi,i})$, stored along the previous T_φ images ($T_\varphi \leq N_\varphi$), α is a mixture factor, γ is the constant density of a uniform random variable in the set of D+2 components defined for the feature vector, and $\mathbf{\Sigma}_\varphi$ is the bandwidth matrix that determines the "width" of the Gaussian kernels, analogously to the one defined for the background, $\mathbf{\Sigma}_\beta$:

$$\mathbf{\Sigma}_\phi = \text{diag}\left(\sigma_{\phi,1}^2, \sigma_{\phi,2}^2 \ldots \sigma_{\phi,D}^2, \sigma_{\phi,H}^2, \sigma_{\phi,W}^2\right), \tag{7}$$

where the first D components specify the bandwidth of the appearance components and the last two are the spatial bandwidths (rows and columns) of the kernels.

Just as in the background modeling and for the same reasons, i.e., to avoid the evaluation of reference samples not contributing significantly to the estimation process, we propose to consider exclusively the foreground reference samples satisfying that

$$\begin{pmatrix}\Delta h & \Delta w\end{pmatrix}\begin{pmatrix}\sigma_{\phi,H}^2 & \sigma_{\phi,H}\sigma_{\phi,W} \\ \sigma_{\phi,H}\sigma_{\phi,W} & \sigma_{\phi,W}^2\end{pmatrix}^{-1}\begin{pmatrix}\Delta h \\ \Delta w\end{pmatrix} \leq 3^2 . \tag{8}$$

Fig. 2 presents an example relating a pixel in a moving object (in red) in the current image with the set of foreground reference samples satisfying equation (8). The considered reference samples are depicted in yellow, the discarded ones are in blue, and the spatial Gaussian on the red pixel is that used in the foreground modeling.

III. GPU IMPLEMENTATION

Modern GPUs have evolved to implement the stream processor paradigm, which is a form of Single Instruction, Multiple Data (SIMD) parallel processing. Under this paradigm, the same series of operations (kernel function) are independently applied onto each element in a set of data (stream), in an unspecified order and in batches of an (a priori) undetermined number of elements. This computing model is especially suitable for applications exhibiting data parallelism, data locality and high compute intensity.

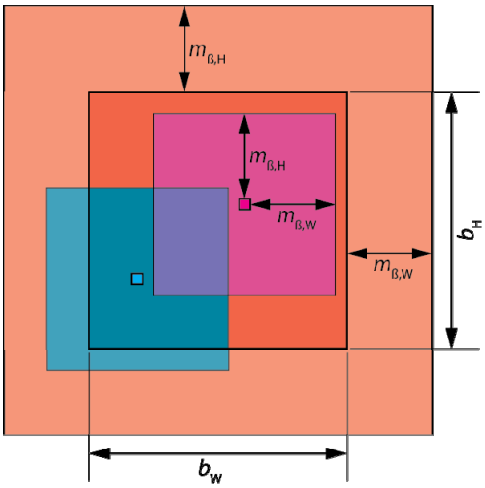

Fig. 3. The areas considered for the spatial modeling of the cyan and magenta pixels overlap. The same occurs for all the pixels processed in this block. The data needed for processing the whole tile/block (dark orange) is therefore that depicted in light orange.

The algorithm just described in the preceding sections is a good candidate to be implemented on a stream processor because it shows, without any modification, two of the three above-mentioned characteristics:

- Data parallelism: for each instant in time, the computations leading to the classification of each pixel as foreground and background do not depend at all on the operations or results of other pixels.
- Data locality: although many operations must be performed on each pixel to determine its class, the only output is the final decision; all the intermediate computations can remain within the stream processor.

If the Gaussian functions are precomputed into lookup tables, the compute intensity is relatively low because for each pixel we read from a reference image only a few sums and multiplications must be performed. However, we can greatly increase the compute intensity and therefore the application throughput by means of careful memory management, as we explain in the remainder of this section.

In order to execute a kernel function on a stream, the workload must be partitioned in execution threads, each taking care of some portion of the data. In our case, one execution thread will be responsible for one pixel. Then, execution threads must be grouped in thread blocks, which are the minimal unit of work that can be scheduled. Finally, depending on the size of the thread blocks and the shared resources they require, one or more blocks are scheduled in run-time to be concurrently executed by each of the execution units in the device (16 in our specific device).

The architecture of the stream processor we have employed features three distinct types of memory:

- Global memory: it is very large (1.5 GiB in our specific device) and its accessible to all threads, but it is located off-chip, is uncached and, therefore, has relatively low bandwidth and high latency.

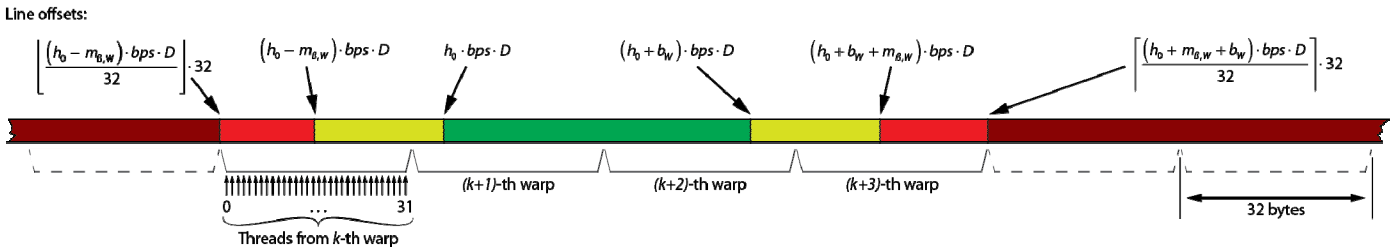

Fig. 4. All threads in a block read lines cooperatively; each thread warp reads one or several 32-byte packages. Useful data (in shades of green) may not be 32-byte aligned, so a small additional amount of data (bright red, not shown to scale) is read together with the desired data.

- Shared memory: it is relatively small (48 KiB per multiprocessor in our device class) and is accessible to all threads belonging to a same thread block. It is located on-chip and has very high bandwidth and low latency. It can be regarded as a manually managed cache.
- Constant memory: it is very small (64 KiB), read-only from the stream processor and is accessible to all threads. It is located off-chip but it is automatically cached, which makes it fast if read many times.

For the spatio-temporal modeling previously defined, we should theoretically consider all data in each reference image for each output pixel. However, reference pixels being weighted by Gaussian functions, we can restrict our region of interest to the following spatial margins (i.e. maximum distance in each dimension from the pixel under study), by eqs. 5 and 8, to consider at least 99% of the accumulated probability:

$$m_{\beta,H} = \left\lceil \frac{3}{\sqrt{2}} \sigma_{\beta,H} \right\rceil \qquad m_{\beta,W} = \left\lceil \frac{3}{\sqrt{2}} \sigma_{\beta,W} \right\rceil$$
$$m_{\phi,H} = \left\lceil \frac{3}{\sqrt{2}} \sigma_{\phi,H} \right\rceil \qquad m_{\phi,W} = \left\lceil \frac{3}{\sqrt{2}} \sigma_{\phi,W} \right\rceil \tag{9}$$

We can see in Fig. 3 that each reference pixel is input for many others; input data must necessarily reside in global memory because it is the only memory zone that is big enough and can be written by the host machine (CPU). In a naïve implementation, each execution thread would directly fetch from global memory the necessary data, thus reading the same data many times over. However, we can massively improve the performance of the application by reducing the number of accesses to the global memory.

Since we are binding each execution thread to a pixel, each thread block corresponds to a tile of the image; the key idea is using the shared memory of each thread block to cache a window of the input data relevant to its threads so that each reference pixel be read only once from the global memory but many times from the shared memory.

For the specific case of the background modeling, the region of interest for a block is the tile mapped to that block, enlarged by the $m_{\beta,W}$ pixels at left and right directions and by the $m_{\varphi,H}$ pixels at the top and bottom (Fig. 3). Therefore, each thread block will need to retrieve $2m_{\beta,W}+b_W$ sub-lines of $2m_{\beta,H}+b_H$ pixels each.

The shared memory region of a block can only be written by the execution threads in that block themselves, which can request data from the global memory and then write them into the shared memory area. The fastest way to perform this memory transfer is to distribute the data among the threads so that they cooperatively read it into the shared memory. However, there are some restrictions imposed by the architecture of our stream processor that must be observed to achieve best performance:

- Execution threads in a thread block are not actually executed all concurrently, but they are partitioned in sub-blocks of 32 threads, called warps.
- Although an execution thread can request single bytes (the size of each of the channels of a pixel in our images) from global memory, the minimal transfer unit is 32 bytes.
- If the memory addresses of the requests from successive threads in the warp are also successive, the requests can be coalesced into a single memory transaction.
- Transfers from global memory are always of naturally aligned blocks. This is, if there are two threads in a warp whose requests are for bytes with different addresses modulo 32, there will be necessarily two separate memory transactions, even if all the addresses in the warp were successive.

Since all variables allocated in the global memory are always guaranteed to be at least 256-byte aligned, if we restrict input images to have a horizontal resolution multiple of 32, the first byte in every line will be 32-byte aligned, regardless of bit depth, denoted by *bps*, and the *D* channels per pixel. Then, we can use the same offsets relative to the start address of each line for reading all the sub-lines in the same block.

As we can see in Fig. 4, h_0 being the horizontal coordinate of the leftmost pixel in a tile/thread block, we need to read pixels from (h_0 - $m_{\beta,H}$) to (h_0 - $m_{\beta,H}$ + b_H - 1) inclusive. In order to respect alignment requirements and the number of pixels to be read being a multiple of the warp size, we will read from the byte offset $\left\lfloor (h_0 - m_{\beta,W})\, bps\, D/32 \right\rfloor \times 32$ up to (not including) the byte offset $\left\lfloor (h_0 + m_{\beta,W} + b_W)\, bps\, D/32 \right\rfloor \times 32$.

Thus, we will read several whole 32-byte blocks per line, and each block will be completely read by a single warp. Distributing all the 32-byte blocks in all the lines to be retrieved uniformly among warps in the block we get the

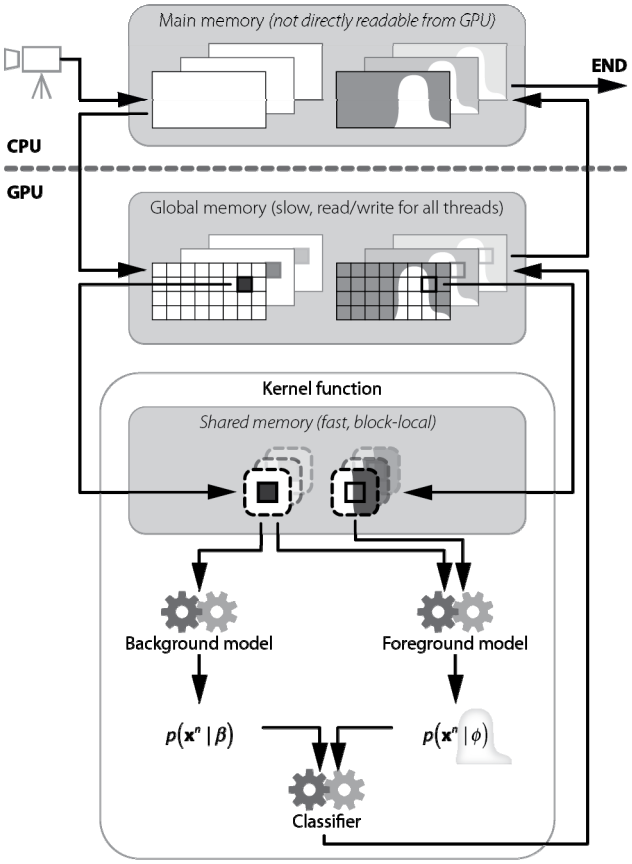

Fig. 5. General flow of the program. The final detections are fed back into the foreground model within the GPU.

TABLE I
DESCRIPTION OF THE TEST SEQUENCES (SORTED BY INCREASING SIZE)

Sequence	Size (height x width)	No. of images	Duration (sec.)
Wall_001	128 x 160	293	14
Wall_002	128 x 160	287	11
Lab_003	192 x 256	550	22
Lab_004	192 x 256	500	20
Lab_001	288 x 352	325	13
Lab_002	288 x 352	380	15
Lab_005	288 x 352	250	10
Pets_004	288 x 352	795	31
Pets_005	288 x 352	795	31
Pets_002	288 x 384	500	20
Pets_001	576 x 768	1452	58
Pets_003	576 x 768	435	17
Total	-	6562	262

maximum transfer speed from global to shared memory and now each thread can proceed to use the shared data to compute the contributions of each reference pixel to the background model of its own pixel. This process is followed iteratively for each reference image.

For the foreground model we follow largely the same process. However, in this case not all the pixels that fall within the spatial margins of the one to be classified must be considered, but only those that were classified as foreground in the reference images. This requires only a small adaptation of the background modeling: because the final product of our computation is, indeed (see Fig. 5), a binary decision on the class of each pixel, we need only retrieve these past images and use them as masks for the reference pixels. In order to read these images we will use the same cooperative reading procedure that we described for color images.

If $m_{\beta,H}=m_{\varphi,H}$ and $m_{\beta,W}=m_{\varphi,W}$ the color data to be read is the same for the background and foreground models. If the spatial margins for the foreground are bigger, the color data needed for the background are a subset of those needed for the foreground. Therefore, we can use the spatial margins of the foreground and process both background and foreground in the same kernel function, thus avoiding read the same data twice.

IV. RESULTS

We have tested our system in several indoor and outdoor sequences with different sizes, recorded with non-stabilized cameras, and containing critical situations such as dynamic backgrounds, multiple moving objects or illumination changes. These sequences have been extracted from the PETS database [29] (Pets_00x), the Wallflower database [30] (Wall_00x) and our own database (Lab_00x). Their main characteristics are shown in Table I.

We have used a buffer of $N_\beta=150$ images and a buffer of $T_\varphi=10$ images to model the background and the foreground, respectively. For simplicity, we have assumed that the prior probabilities of the two classes, background and foreground, is identical, so $\Pr(\beta)=\Pr(\varphi)=½$. The appearance information of the pixels are their RGB color components, so $D=3$, and we have set the background and foreground appearance bandwidths as $\sigma_{\beta,j}=\sigma_{\varphi,j}=16$ in all our experiments. We have also used precalculated lookup tables for the kernel function values given the bandwidth matrices, $\mathbf{\Sigma}_\beta$ and $\mathbf{\Sigma}_\varphi$, and the differences $(\mathbf{x}^n - \mathbf{x}_{\beta,i})$ and $(\mathbf{x}^n - \mathbf{x}_{\varphi,i})$.

The achieved computational efficiency has been compared to that obtained with a 2.66 GHz CPU with 2 GB RAM, where lookup tables have been used as well.

A. Background modeling

In the first place we have analyzed the computational cost in the background modeling using several block sizes (b_H, b_W) and different background spatial margins ($m_{\beta,H}$, $m_{\beta,W}$). The obtained results appear in Fig. 6, which shows how processing times depend exponentially on the spatial margin as a general rule. However, there is no optimal block size for all spatial margins. To an extent, small blocks perform better for small spatial margins and large blocks perform better for large spatial margins. The main factors we have identified as

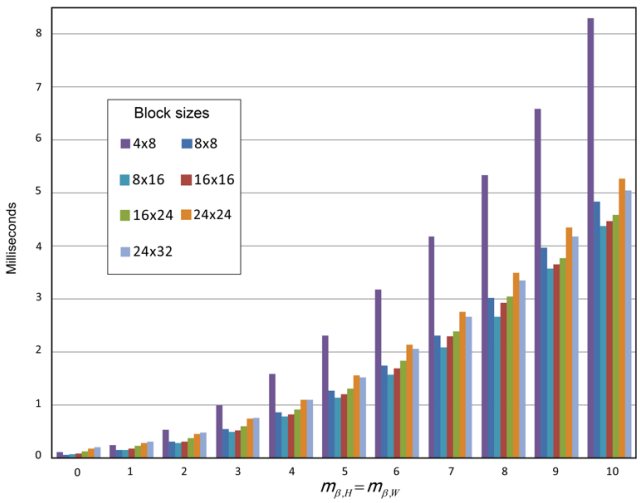

Fig. 6. Computational cost in background modeling for different block sizes (b_H, b_W) and different spatial margins ($m_{\beta,H}$, $m_{\beta,W}$).

responsible for that behavior are:

- Duplicated data: in previous sections we have seen how for each tile/block a region-of-interest window must be read. The larger the spatial margins in comparison to the block size, the more times each datum must be read from global memory.
- Synchronization costs: each region-of-interest is cooperatively read by all the threads in a block. Before each thread can proceed with the computation, a synchronization barrier is necessary. The larger the block size, the more threads need to be synchronized.
- Scheduling overhead: dispatching a block to a multiprocessor adds a small overhead to the processing time. If each block does too little work (i.e. is too small), this overhead is significant.

To avoid large amounts of false detections in sequences containing local or global background displacements (such as those containing non-static background regions or sequences recorded with portable or non-stabilized cameras) the use of spatial information in the background modeling is essential [19]. In addition, more significant background displacements require the use of more spatial information (larger spatial bandwidths). However, the performed experiments have shown that using small spatial bandwidths the background is correctly modeled and most false detections due to local or global background displacements are avoided.

Fig. 7 shows some obtained detections in two sequences with non-static backgrounds. The first column presents a sequence with a local background displacement (a tree shaken by the wind), while the second column shows a sequence with a global background displacement (a vibration of the camera). The results in this figure allow us to appreciate that using $\sigma_{\beta,H}=\sigma_{\beta,W}=\sqrt{2}/3$ $\left(m_{\beta,H}=m_{\beta,W}=1\right)$ most false detections due to background displacements are avoided. Therefore, as larger spatial bandwidths increase the computational cost (see Fig. 6), we have decided to use $\sigma_{\beta,H}=\sigma_{\beta,W}=\sqrt{2}/3$ in all the test sequences.

Table II presents the mean processing times (once the background model has been fully initialized) obtained with the

Fig. 7. Moving object detection with different spatial bandwidths in the background modeling. (a) Original images from two sequences. (b) Detections with $\sigma_{\beta,H}=\sigma_{\beta,W}=0.01$. (c) Detections with $\sigma_{\beta,H}=\sigma_{\beta,W}=\sqrt{2}/3$.

TABLE II
MEAN PROCESSING TIMES (MILLISECONDS) IN BACKGROUND MODELING

Sequence	CPU	GPGPU	Speedup
Wall_001	1697	9	199×
Wall_002	1707	10	171×
Lab_003	3958	19	206×
Lab_004	3972	18	215×
Lab_001	8551	38	227×
Lab_002	8415	38	221×
Lab_005	8397	38	220×
Pets_004	8549	35	245×
Pets_005	8517	35	243×
Pets_002	9162	39	238×
Pets_001	33014	145	228×
Pets_003	33108	139	238×

proposed implementation, compared to the results obtained with a CPU. These results show that the computational cost to model the background is roughly proportional to the size of the sequences and demonstrate that the proposed GPGPU-based implementation improves the computational efficiency very significantly.

B. Foreground modeling

As has been done for the background modeling, in the first place we have analyzed the computational cost using several block sizes (b_H, b_W) and different foreground spatial margins ($m_{\varphi,H}$, $m_{\varphi,W}$). Fig. 8 summarizes the measured processing times for several block sizes. Unsurprisingly, since the nature of the

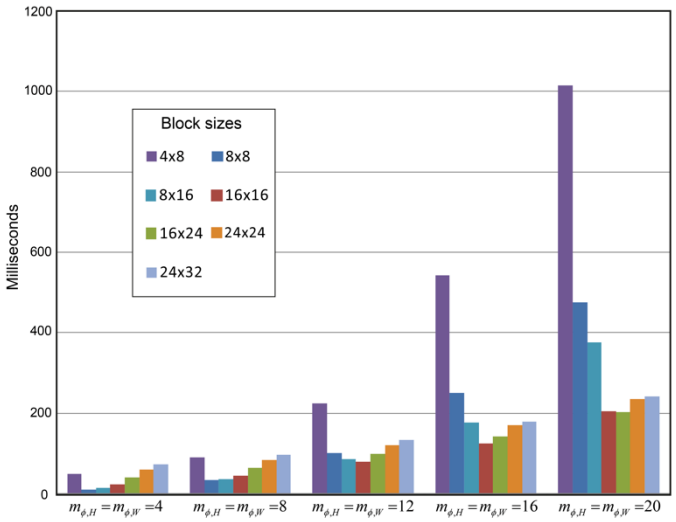

Fig. 8. Computational cost in foreground modeling for different block sizes (b_H, b_W) and different spatial margins ($m_{\varphi,H}$, $m_{\varphi,W}$).

TABLE III
MEAN PROCESSING TIMES (MILLISECONDS) IN FOREGROUND MODELING

Sequence	Foreground pixels	Reference pixels	CPU	GPGPU	Speedup
Wall_001	6.79 %	30.12 %	1415	16.62	85×
Wall_002	1.52 %	2.56 %	86	17.79	5×
Lab_003	2.72 %	13.11 %	2057	37.20	55×
Lab_004	2.79 %	31.35 %	7611	36.42	209×
Lab_001	1.00 %	12.24 %	12154	70.06	173×
Lab_002	2.47 %	29.49 %	19440	72.21	269×
Lab_005	0.94 %	9.94 %	8597	66.90	129×
Pets_004	5.40 %	14.55 %	11218	76.52	147×
Pets_005	6.27 %	17.96 %	14478	77.05	188×
Pets_002	0.00 %	0.00 %	9	76.98	9÷
Pets_001	1.12 %	5.57 %	16944	322.21	53×
Pets_003	0.37 %	7.39 %	19125	296.03	65×

background and foreground models is largely the same, the same dependencies between block sizes and spatial margins can be found: large spatial margins are best processed using large block sizes, for the same reasons explained earlier.

After performing numerous experiments we have found that using a spatial margin of $m_{\varphi,H}= m_{\varphi,W}= 12$ ($\sigma_H= \sigma_W\approx 5.5$) is sufficient to take into account most reference data in the T_φ =10 reference images (see Fig. 2). Therefore, we have decided to use these values.

Taking into account the spatial margins finally used to model both the background ($m_{\beta,H}=m_{\beta,W}=1$) and the foreground ($m_{\varphi,H}=m_{\varphi,W}=12$), and considering the performance obtained with the analyzed block sizes (Fig. 6 and Fig. 8), we have decided to use blocks of size (b_H,b_W)=(*16*,*16*).

Table III shows the mean processing times (compared to the results obtained with a CPU) in foreground modeling, using the above mentioned block sizes and spatial margins. These

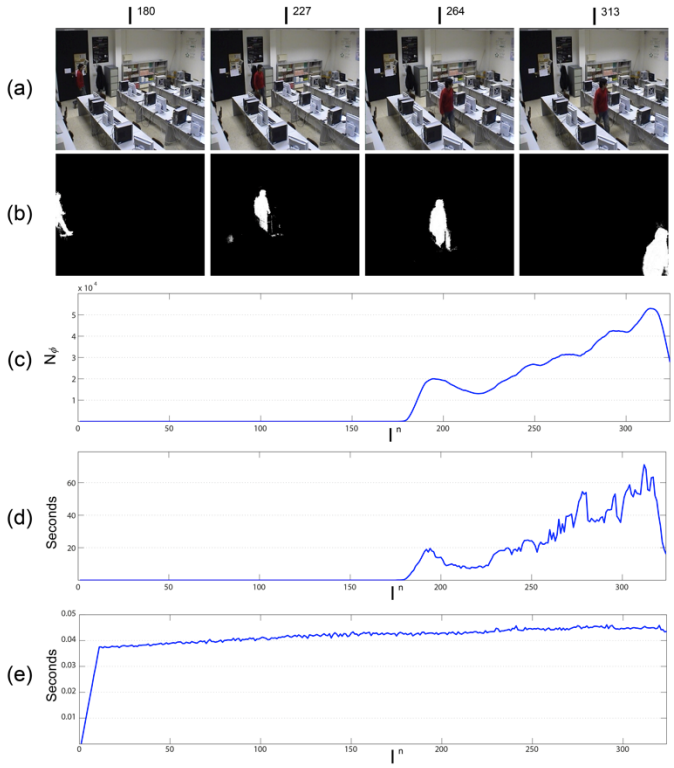

Fig. 9. Computational cost in foreground modeling along a sequence with 325 images. (a) Representative original images. (b) Final detections. (c) Number of foreground reference samples. (d) Processing time in a CPU. (e) Processing time in a GPGPU.

results demonstrate that, similarly to the background timing analysis (Table II), the cost of modeling mainly depends on the size of images. Additionally, data from this table show that, in contrast to the results obtained in a CPU, where the foreground computational cost is highly dependent on the number of reference pixels, the proposed implementation reduces drastically that dependence.

Fig. 9 illustrates this dependence over a sequence of 325 frames. The first row of images (Fig. 9.a) shows some representative frames of the sequence, while the second (Fig. 9.b) presents the obtained detections. Fig. 9.c shows the amount of used reference data (N_φ) along the sequence, Fig. 9.d presents the computational cost using a CPU, and Fig. 9.e details the computational cost with the proposed implementation in a GPGPU. As can be observed, while the cost in a CPU is roughly proportional to N_φ, thanks to the GPGPU-based implementation we have removed this dependence. While in most cases this is an advantage, because it allows to establish a maximum processing time depending on the image size of the sequences, this can be a disadvantage in sequences with low amount of motion (e.g.Pets_002), because in the GPGPU-based implementation we incur a minimum cost unconditionally.

C. Final detections

Finally, Fig. 10 and Table IV show some final results: Fig. 10 compares the quality of the detections obtained with the proposed strategy and with a MoG-based algorithm; Table

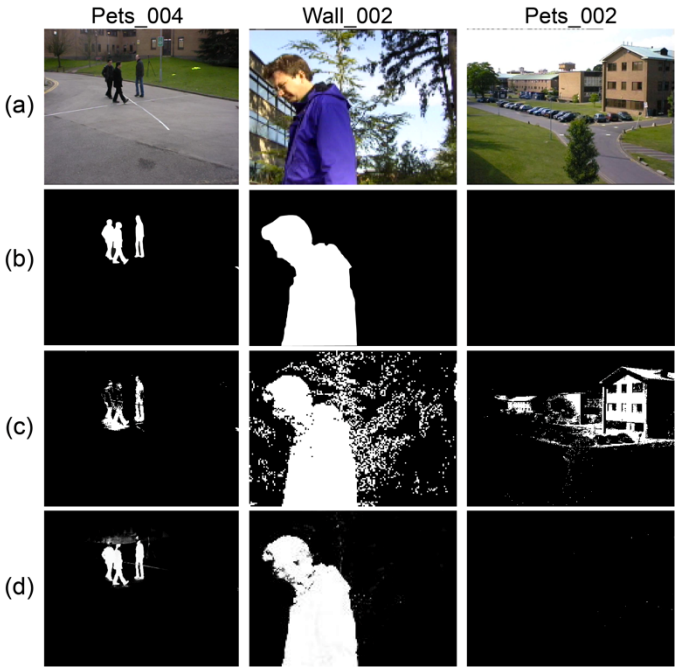

Fig. 10. Results in three different sequences. (a) Original images. (b) Ground truth. (c) Results using the MoG method. (d) Results by applying the proposed strategy.

TABLE IV
TOTAL MEAN PROCESSING TIMES (MILLISECONDS)

Sequence	CPU	GPGPU	Speedup	fps
Wall_001	2678	27	101×	38
Wall_002	1348	28	48×	36
Lab_003	5476	57	96×	17
Lab_004	10992	56	195×	18
Lab_001	18732	100	187×	10
Lab_002	26194	102	256×	10
Lab_005	14475	97	149×	10
Pets_004	18960	107	178×	9
Pets_005	22194	107	207×	9
Pets_002	7797	107	73×	9
Pets_001	48253	462	104×	2
Pets_003	46608	406	115×	2

IV shows the final computational costs, considering jointly the background and the foreground modeling.

The detections shown in Fig. 10 demonstrate that the proposed non-parametric strategy improves the quality of the detections provided by other popular strategies in multiple complex scenarios: in sequences with moving objects similar to some background regions (Pets_004), in sequences with dynamic backgrounds (Wall_002), or in sequences with illumination changes (Pets_002).

Results in Table IV allow us to appreciate that our strategy can very significantly improve the results on a CPU. Furthermore, as shown by the results of the last column of the table, we are able to process multiple images per second (even for very large sequences as Pets_001 and Pets_003).

Consequently, since the proposed strategy provides high quality detections and is also able to process at high speed, it seems ideal for integration into AR applications required by last generation consumer electronic devices.

V. CONCLUSION

We have described a novel background-foreground non-parametric-based moving object detection strategy that we have efficiently implemented in a GPGPU.

By modeling both background and foreground from spatio-temporal reference information we are able to provide real-time high quality detections in sequences recorded with portable devices (mobiles, handy-cams, etc) in a large variety of multimodal and complex scenarios (e.g. dynamic backgrounds, multiple moving objects or illumination changes). Therefore, the proposed strategy is suitable for AR tools required by the last generation of consumer electronic devices endowed with integrated smart cameras.

REFERENCES

[1] D. W. F. van Krevelen and R. Poelman, “A Survey of augmented reality technologies, applications and limitations”, The International Journal of Virtual Reality, vol. 9, no. 2, pp. 1-20, 2010.

[2] A. Mulloni and T. Drummond, “Real-time detection and tracking for augmented reality on mobile phones”, IEEE Trans. Visualization and Computer Graphics, vol. 16, no. 3, pp. 355-368, 2010.

[3] W. Lao, J. Han, and P. h. N. de With, "Automatic video-based human motion analyzer for consumer surveillance system", IEEE Trans. Consumer Electronics, vol. 55, no. 2, pp. 591-598, 2009.

[4] T. Bouwmans, F. E. Baf, and B. Vachon, “Background modeling using mixture of Gaussians for foreground detection – a survey”, Recent Patents on Computer Science, vol. 1, no. 3, pp. 219-237, 2008.

[5] L. Wixton, “Detecting salient motion by accumulating directionally-consistent flow”, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 22, no. 8, pp. 774-780, 2000.

[6] M. Piccardi, “Background subtraction techniques: a review”, IEEE Int. Conf. Systems, Man and Cybernetics, vol. 4, pp. 3099-3104, 2004.

[7] A. Tavakkoli, M. Nicolescu, G. Bebis, and M. Nicolescu, “Non-parametric statistical background modeling for efficient foreground region detection,” Machine Vision and Applications, vol. 20, no. 6, pp. 395-409, 2009.

[8] R. Li, S. Yu, and X. Yang, “Efficient spatio-temporal segmentation for extracting moving objects in video sequences”, IEEE Trans. Consumer Electronics, vol. 53, no. 3, pp. 1161-1167, 2007.

[9] R. Pless, J. Larson, S. Siebers, and B. Westover, “Evaluation of local models of dynamic backgrounds”, IEEE Conf. Computer Vision and Pattern Recognition, vol. 2, pp. 73-78, 2003.

[10] C. Stauffer and W. E. L. Grimson, “Learning patterns of activity using real-time tracking,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 22, no. 8, pp. 747–757, 2000.

[11] M. Bicego, M. Cristani, and V. Murino, “Unsupervised scene analysis: a hidden Markov model approach,” Computer Vision and Image Understanding, vol. 102, no. 1, pp. 22–41, 2006.

[12] M. Cristani, M. Farenzena, D. Bloisi, and V. Murino, “Background subtraction for automated multisensor surveillance: a comprehensive review”, EURASIP Journal on Advances in signal Processing, pp. 43-66, 2010.

[13] S. Elhabian, K. El-Sayed, and S. Ahmed, “Moving object detection in spatial domain using background removal techniques-state-of-art”, Recent Patents on Computer Science, vol. 1, no. 1, pp. 32-54, 2008.

[14] J. Ding, M. Li, K. Huang, and T. Tan, "Modeling complex scenes for accurate moving objects segmentation", Computer Vision-ACCV, pp. 82-94, 2011.
[15] T. Tanaka, A. Shimada, R. Taniguchi, T. Yamashita, and D. Arita, "Towards robust object detection: integrated background modeling based on spatio-temporal features", Computer Vision and Image Understanding, vol. 113, no. 1, pp. 63-79, 2009.
[16] A. Mittal and N. Paragios, "Motion-based background subtraction using adaptive kernel density estimation", IEEE Conf. Computer Vision and Pattern Recognition, vol. 2, pp. 302-309, 2004.
[17] Y. Sheikh and M. Shah, "Bayesian modeling of dynamic scenes for object detection", IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 27, no. 11, pp. 1778-1792, 2005.
[18] N. Martel-Brison and A. Zaccarin, "Unsupervised approach for building non-parametric background and foreground models of scenes with significant foreground activity", Proc. ACM workshop on Vision networks for behavior analysis, pp. 93-100, 2008.
[19] Y. Sheikh, O. Javed, and T. Kanade, "Background subtraction for freely moving cameras", IEEE Int. Conf. Computer Vision, pp. 1219-1225, 2009.
[20] L. F. Wang, H. Y. Wu, and C. H. Pan, "Adaptive εLBP for background subtraction", Computer Vision-ACCV, pp. 560-571, 2011.
[21] C. Cuevas, R. Mohedano, F. Jaureguizar, and N. García, "High-quality real-time moving object detection by non-parametric segmentation", Electronics Letters, vol. 46, no. 13, pp. 910-911, 2010.
[22] T. D. Han and T. S. Abdelrahman, "hicuda: High-level GPGPU programming", IEEE Trans. Parallel and Distributed Systems, vol. 22, no. 1, pp. 78-90, 2011.
[23] B. Neelima and P. S. Raghavendra, "Recent trends in software and hardware for GPGPU computing: a comprehensive survey", IEEE Int. Conf. Industrial and Information Systems, pp. 319-324, 2010.
[24] S. W. Keckler, W. J. Dally, B. Khailany, M. Garland, and D. Glasco, "GPUs and the future of parallel computing", IEEE Micro, vol. 31, no. 5, pp. 7-17, 2011.
[25] S. F. Tsai, C. C. Cheng, C. T. Li, and L. G. Chen, "A real-time 1080p 2D-to-3D video conversion system", IEEE Trans. Consumer Electronics, vol. 57, no. 2, pp. 915-922, 2011.
[26] H. C. Shin, Y. J. Kim, H. Park, and J. I. Park, "Fast view synthesis using GPU for 3D display", IEEE Trans. Consumer Electronics, vol. 54, no. 4, pp. 2068-2076, 2008.
[27] S. W. Ryu, S. H. Lee, S. Ahn, and J. I. Park, "Tangible video teleconference system using real-time image-based relighting", IEEE Trans. Consumer Electronics, vol. 55, no. 3, pp. 1162-1168, 2009.
[28] C. Cuevas and N. García, "Automatic bandwidth estimation strategy for high-quality non-parametric modeling based moving object detection", IEEE Int. Conf. Image Processing, pp. 1757-1760, 2011.
[29] Computational Vision Group, "Pets: Performance evaluation of tracking and surveillance", Computational Vision Group, University of Reading, England.
[30] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: principles and practice of background maintenance", IEEE Int. Conf. Computer Vision, vol. 1, pp. 255-261, 1999.

BIOGRAPHIES

Carlos Cuevas received the Ingeniero de Telecomunicación degree (integrated BSc-MSc accredited by ABET) in 2006 and the Doctor Ingeniero de Telecomunicación degree (PhD in Communications) in 2011, both from the Universidad Politécnica de Madrid (UPM), Spain.
Since 2006 he is a member of UPM's Grupo de Tratamiento de Imágenes (Image Processing Group). His research interests include signal and image processing, computer vision, pattern recognition and automatic target recognition.

Daniel Berjón received the Ingeniero de Telecomunicación degree (integrated BSc-MSc accredited by ABET) from the Universidad Politécnica de Madrid (UPM), Spain, in 2005, and is currently a student of UPM's PhD in Communications program.
Since 2008 he is a member of UPM's Grupo de Tratamiento de Imágenes (Image Processing Group). His research interests include computer graphics, parallel processing and real-time systems.

Francisco Morán received the Ingeniero de Telecomunicación degree (six years engineering curriculum) in 1992 and the Doctor Ingeniero de Telecomunicación degree (PhD in Communications) in 2001, both from the Universidad Politécnica de Madrid (UPM), Spain.
Since 1992 he is a researcher at UPM's Grupo de Tratamiento de Imágenes (Image Processing Group), and since 1997 a member of UPM's faculty. His research interests include hierarchical modeling and coding, and adaptive transmission and visualization of 3D objects. He has been actively involved in research projects funded by the European Union, and participates since 1996 in the standardization activities from ISO's Moving Picture Experts Group (MPEG), where he is the Head of the Spanish Delegation since 2006, and has served as (co-)editor of eight standards, amendments and corrigenda related to MPEG-4 (formally, ISO/IEC 14496).

Narciso García received the Ingeniero de Telecomunicación degree (five years engineering curriculum) in 1976 (Spanish National Graduation Award) and the Doctor Ingeniero de Telecomunicación degree (PhD in Communications) in 1983 (Doctoral Graduation Award), both from the Universidad Politécnica de Madrid (UPM), Spain.
Since 1977 he is a member of UPM's faculty, currently a Professor of Signal Theory and Communications. He leads UPM's Grupo de Tratamiento de Imágenes (Image Processing Group). He has been actively involved in Spanish and European research projects, serving also as evaluator, reviewer, auditor, and observer of several research and development programmes of the European Union. He was a co-writer of the EBU proposal, base of the ITU standard for digital transmission of TV at 34-45 Mb/s (ITU-T J.81). His professional and research interests are in the areas of digital image and video compression and of computer vision.