International Journal of Approximate Reasoning 51 (2009) 99–114

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

journal homepage: www.elsevier.com/locate/ijar

Real-time moving object segmentation in H.264 compressed domain based on approximate reasoning

C. Solana-Cipres [a,*], G. Fernandez-Escribano [c], L. Rodriguez-Benitez [a], J. Moreno-Garcia [b], L. Jimenez-Linares [a]

[a] ORETO Research Group, University of Castilla-La Mancha, Tecnologias y Sistemas de Informacion, Paseo de la Universidad s/n, 13071 Ciudad Real, Spain
[b] Escuela Ingenieria Tecnica Industrial, Avda. Carlos III s/n, 45071 Toledo, Spain
[c] Instituto de Investigacion en Informatica, Campus Universitario s/n, 02071 Albacete, Spain

ARTICLE INFO

Article history:
Received 3 March 2009
Received in revised form 3 September 2009
Accepted 3 September 2009
Available online 9 September 2009

Keywords:
Moving object detection
Compressed video segmentation
H.264 advanced video coding
Dynamic fuzzy sets
Approximate reasoning

ABSTRACT

This paper presents a real-time segmentation algorithm to obtain moving objects from the H.264 compressed domain. The proposed segmentation works with very little information and is based on two features of the H.264 compressed video: motion vectors associated to the macroblocks and decision modes. The algorithm uses fuzzy logic and allows to describe position, velocity and size of the detected regions in a comprehensive way, so the proposed approach works with low level information but manages highly comprehensive linguistic concepts. The performance of the algorithm is improved using dynamic design of fuzzy sets that avoids merge and split problems. Experimental results for several traffic scenes demonstrate the real-time performance and the encouraging results in diverse situations.

1. Introduction

In Computer Vision, moving object segmentation refers to the process of identifying and partitioning the meaningful regions present in video sequences. Object segmentation is an essential component of an intelligent video surveillance system. Accurate and real-time moving object segmentation will greatly improve the performance of object tracking, activity analysis and high-level event understanding. In fact, segmentation could be considered as the main column of the surveillance systems due to motion detection has to support the next stages in the third generation surveillance systems [28]. However, the problem of multiple objects segmentation in complex scene is still far from being completely solved because segmentation is difficult for several reasons: there could be multiple moving objects in a scene, the objects are usually small and poorly textured, illumination conditions may be poor and change rapidly or there could be shadows and multiple occlusions [10]. Even though, the last researches in the field show promising results in accuracy as well as in speed processing.

Classical techniques like *background subtraction*, *temporal differences* and *optical flow* use low level information about each pixel. These algorithms have a good performance and can identify with accuracy the border of the regions, but the most of them cannot work in real-time because they have to decode and process each pixel of each frame. So fast algorithms to segment moving objects usually work directly on compressed video. The proposed algorithm uses only the information related to the motion compensation and works with groups of pixels called *macroblocks*.

* Corresponding author. Tel.: +34 647 756 464.
E-mail address: cayetanoj.solana@uclm.es (C. Solana-Cipres).

0888-613X/$ - see front matter Crown Copyright © 2009 Published by Elsevier Inc. All rights reserved.
doi:10.1016/j.ijar.2009.09.002

1.1. Contributions of this work

A new approach to the segmentation of moving objects in compressed domain is presented. The proposed segmentation algorithm is designed for the H.264 advanced video coding standard. This standard has a better compression ratio than the MPEG compressed video family standards and is widely used at present. This work is based on previous algorithms for MPEG [21,22], but the new proposal is faster and can fulfil the requeriments of real-time applications. The average processing time goes from 17 ms to 42 ms per frame depending on the video resolution, thus it satisfies the requirements of the most real-time systems. This fact is relevant if the target of the segmentation is a video surveillance system.

The proposed approach is based on fuzzy logic to detect the moving objects; fuzzy sets allow to avoid the noise inherent to the encoding process and to obtain conceptual representations that describe the regions detected in a comprehensive way. By using approximate reasoning and a clustering algorithm, the segmentation method obtains the moving regions of each frame and describe them with common terms like shape, size, position and velocity. These linguistic representations allow to understand the objects in the scene and make classification of the different objects easier. That is, the conceptual representations could allow to classificate the detected regions as different kinds of objects like persons, groups of persons, motorbikes, cars or trucks [1].

Another contribution of this work is the improvement of the algorithm's robustness. This improvement can be obtained because the proposed technique can adapt itself to the scenario and to the objects in a dynamic way. Thus, the segmentation procedure allows to give itself some feedback to improve its results. The algorithm has an initial configuration, but it could change the values of the fuzzy sets in function of the obtained results taking into consideration the size and shape of the obtained regions in previous frames. More concretely, the fuzzy sets describing the linguistic variables are updated dynamically and this fact allows to reduce the merge and split ratios.

Finally, the last contribution of this approach is the use of the H.264 macroblock (MB) decision mode, known as tree structured motion compensation, for helping our moving detection system in a scene. In H.264, inter frame motion estimation is performed for different sizes from 16×16 to 4×4 pixels. For each MB, all the sizes are tried and the one that leads to the minimum cost is selected (the cost is evaluated using different kinds of mathematical functions). This *try all and select the best* philosophy is optimal for deciding the block size and presents a side effect; it selects small size partitions when high motion is achieved. In this way, this information allows to classify the amount of movement based on the macroblock mode selection.

1.2. Paper structure

The paper is organized as follows. Section 2 briefly reviews some related work of linguistic representations using approximate reasoning and segmentation over compressed domain. Then, some relevant features of the H.264 advanced video coding are described in Section 3. Fuzzy linguistic concepts are shown in Section 4. Later, in Section 5 it is analyzed the proposed method of moving object segmentation. The experimental results in several video traffic scenes are presented in Section 6. Finally, conclusions and future works are described in Section 7.

2. Related work

The main problem related to the **compressed video** processing is the shortage of information and the uncertainty that it implies, but it has the advantage of low computational complexity. In this segmentation method luminance and chrominance values are not used, but information related to the motion compensation between frames is needed. This fact implies the increase of the uncertainty related to the input information. In this noisy environment, fuzzy logic presents itself as the adequate theoretical medium to reduce the negative influence of the uncertainty [18,25].

Fuzzy set methods have been used to model and manage uncertainty in various aspects of pattern recognition, image processing and computer vision [14,23] because of its potential and its link with the natural language. Several fuzzy techniques have been used to solve computer vision problems like evidential filters [17], fuzzy classifiers [8] or genetic algorithms [26], between many others. It allows to translate video information in linguistic representations capable to be processed through fuzzy techniques. **Approximate reasoning** is used in this work to carry out a segmentation algorithm of moving objects that obtains conceptual representations which describe position, velocity and size of the regions detected in a comprehensive way. It is important to have a semantic interpretation of the behaviours of the recognised objects in order to build an automated surveillance system that is able to recognise and learn from the events and interactions that occur in a monitored environment [28].

The segmentation algorithms that work directly over **MPEG-2 compressed domain** exploit two features of macroblock: motion vectors and DCT coefficients. The first class of approaches uses only motion vector information; Schonfeld and Lelescu [24] track multiple objects from MPEG video by filtering and analyzing motion vectors information. Mezaris et al. [16] use a macroblock-level tracking algorithm to track the connected regions and finally obtain the objects. The second class exploits only DCT coefficients by using an adaptative K-means clustering algorithm [27] or by using a matching template [4]. Finally, the third class of algorithms exploits both DCT coefficients and motion vectors; for example, Jamrozik and Hayes [9]

obtain moving regions through the quantized magnitude of motion vectors or Eng and Ma [6] use a maximum entropy fuzzy clustering algorithm.

However, very few approaches have been proposed for video object segmentation in the **H.264 compressed domain**. Zeng et al. [30] present an algorithm that employs a block-based Markov Random Field (MRF) model to segment moving objects from the sparse motion vector field obtained directly from the H.264 bitstream. This approach is only applicable to video sequences with stationary background. A later approximation has been proposed in Liu et al. [13], where a real-time spatiotemporal segmentation is presented. In this case, spatial segmentation only exploits the motion vector field extracted from the H.264 compressed video. Regions are classified using the block residuals of global motion compensation and the projection is exploited for inter-frame tracking of video objects. However, this algorithm uses temporal information and could be considered as a tracking algorithm.

3. Motion compensation in H.264 advanced video coding

H.264 [11], also known as MPEG-4 Part 10, is a standard for video compression developed jointly between the *Motion Picture Expert Group* (MPEG) and the *Video Coding Experts Group* (VCEG). Richardson explains deeply in [20] the features of the H.264 compressed domain, but in this section only the motion compensation in H.264 is described. This standard provides mechanisms for video coding that are optimized for a better compression efficiency and aims to meet the multimedia communication applications.

The basic unit in which an image is divided into is the **macroblock**. It contains the information of a 16×16 pixels region and there are two types depending on the encoding: **Intra macroblock**, in which Intra-prediction algorithms are applied directly to exploit the spatial redundancy according to the H.264 standard, and **Inter macroblock**, in which motion compensation is used to exploit temporal redundancy from a reference macroblock (earlier, later or a combination of both).

H.264 uses *block-based motion compensation*, the same principle adopted by every major coding standard since H.261. This motion compensation is done through the redundant information between consecutive frames looking for a pattern that captures the kind of movement between pictures. This pattern is represented as a **motion vector**, which defines a distance and a direction and has two dimensions: *right_x* and *down_x*. Important differences from earlier standards include the support for a range of block sizes and the use of multiple reference frames to improve the performance of the coding.

H.264 supports motion compensation block sizes ranging from 16×16 to 4×4 samples. Each macroblock may be split up in four ways: 16×16, 16×8, 8×16 or 8×8. Each of the sub-divided regions is a **macroblock partition**. If the 8×8 mode is chosen, each of the four 8×8 macroblock partitions within the macroblock may be further split in four ways: 8×8, 8×4, 4×8 or 4×4 (known as **sub-macroblock partitions**). These partitions and sub-partitions give rise to a large number of possible combinations within each macroblock (Fig. 1). This method of partitioning macroblocks into motion compensated sub-blocks of varying size is known as *tree structured motion compensation*.

Since each macroblock and sub-macroblock partition has a motion vector associated, a macroblock has from 0 to 16 pairs of *motion vectors*. For example, an Intra macroblock has not any motion vector, an Inter macroblock partitioned into two 16×8 blocks has two pairs of motion vectors and an Inter macroblock partitioned into four 8×8 blocks -each of them partitioned into 4×4 sub-macroblocks- has 16 pairs of motion vectors ($4 * 4 = 16$).

Then, a macroblock can be encoded with different *block sizes* (Fig. 1). Since H.264 allows block-based motion compensation, the *decision mode* and the *block size* give additional information about the motion degree on a specific region of a frame (Fig. 2). Thus the more partitions into a macroblock, the more implicit motion ratio over that region. Note that light gray macroblocks in Fig. 2c are skip macroblocks and dark gray ones are intra-frame macroblocks, i.e., those with decision mode greater than 8. The H.264 standard supports the **decision modes** illustrated in Table 1, where can also be seen the average frequency detected in the experimental examples. This information is relevant to the segmentation algorithm because the

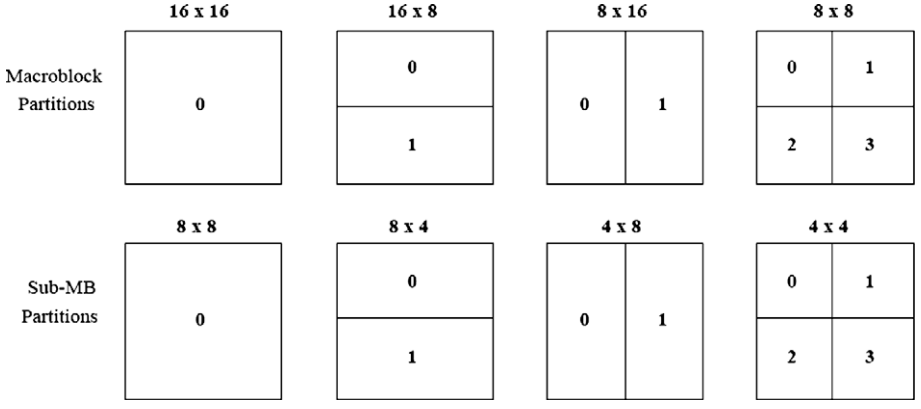

Fig. 1. Macroblock types.

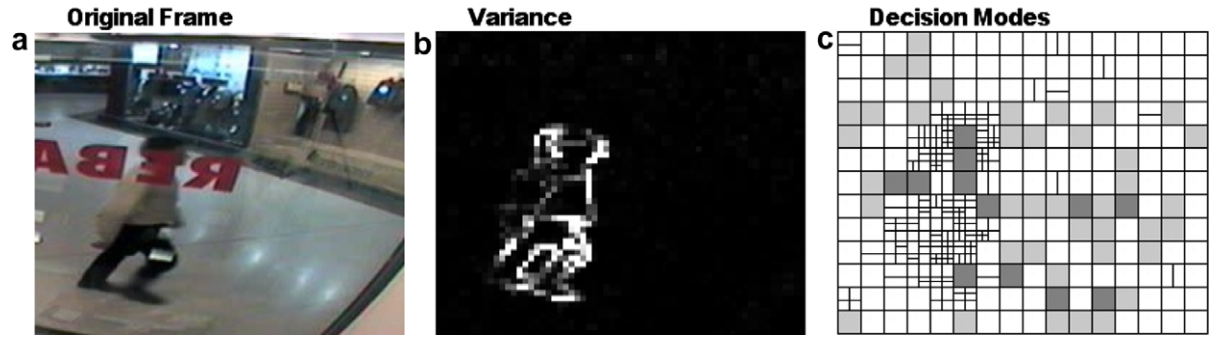

Fig. 2. Processing a frame: (a) original frame extracted from [19], (b) variance of the residual frame deleting temporal redundancy and (c) decision modes or macroblock partitions of the frame due to the H.264 encoding.

Table 1
Adopted nomenclature for decision modes.

Id	Macroblock type	Frequency 1 (%)	Frequency 2 (%)
0	Skip, without added information	41.46	0.26
1	One 16×16 partition	39.51	4.73
2	Two 16×8 partitions	1.36	2.79
3	Two 8×16 partitions	0.94	1.21
4	Four 8×8 partitions	6.5	87.47
5	Two 8×4 sub-partitions in an 8×8 partition	–	–
6	Two 4×8 sub-partitions in an 8×8 partition	–	–
7	Four 4×4 sub-partitions in an 8×8 partition	–	–
8	Four 8×8 partitions without sub-partitions	0.13	0.00
9	4×4 Intra-frame macroblock	5.26	3.53
10	16×16 Intra-frame macroblock	2.89	0.00
11	Without using in a standard encoding	–	–
12	Without using in a standard encoding	–	–
13	8×8 Intra-frame macroblock	1.92	0.00

greater the decision mode identificator is, the more motion degree is considered by the motion estimation and compensation of the H.264 codec. In Table 1, *Frequency 1* represents the overall percentage of macroblocks of each type where the result has been obtained as the average of three video sequences (Fig. 13) with different resolution, frames and conditions. *Frequency 2* represents the percentage of macroblocks belonging to moving regions. Modes from 5 to 7 are sub-partitions of a macroblock type 4, so the results would be confused by adding the frequency of each sub-partition.

4. Fuzzy linguistic concepts

The motion vectors of a H.264 video flow are imprecise approximations to the real *optical flow fields* [2]. In H.264 frames, the more sensitive areas to contain wrong motion vectors [7] are those with imperceptible variation of luminance values (Fig. 3a) and those corresponding to the boundary objects (Fig. 3b). In cases like that, it is complex to the motion compensation algorithms to find correspondences between macroblocks in consecutive frames. The most compressed domain tech-

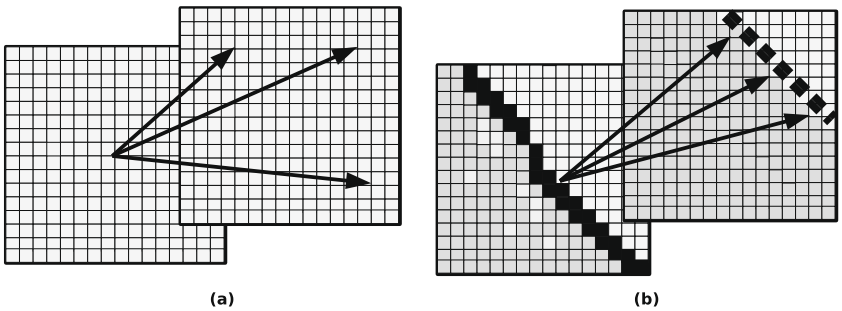

Fig. 3. Uncertainty inherent into the H.264 coding: (a) imperceptible variation of luminance values in a region and (b) boundary objects.

niques for image manipulation apply any filter to the motion vectors field to avoid the anomalous values as consequence of the mistakes introduced by the encoders. The use of the filters implies the growth of the information used in the algorithm and this kind of operations use to be expensive from a computational point of view; thus, these procedures are inappropriate for real-time algorithms.

The related uncertainty justifies the use of the fuzzy logic in the H.264 compressed domain. The algorithm uses approximate reasoning to compact the information related to the video flow, so values not much affected by the encoding error will have similar fuzzy values. Besides, given that the goal of this work is to obtain the linguistic description of the moving regions present in a video scene, it is necessary to define the set of linguistic elements used in the description of the regions.

4.1. Linguistic variable

The proposed algorithm uses four linguistic variables [29] to represent the horizontal position (*HP*), vertical position (*VP*), horizontal velocity (*HV*) and vertical velocity (*VV*). One of the critical points of the system is the definition of the membership function for each variable because it depends on the scenario under observation. For example, Fig. 4 illustrates the linguistic labels belonging to the linguistic variable *HP* used as initial values in this study. These values can be adapted to obtain the optimal design for the specific scenario.

The linguistic variables *HP* and *VP* represent the position of a macroblock and depends on the frame size. For example, in a video sequence with 320×240 pixels, the *HP* variable has a domain from 0 to 19 because there are 20 macroblocks for each frame row ($20 * 16 = 320$). For the cited example, the *VP* variable has a domain from 0 to 14 because each column has 15 macroblocks.

4.2. Linguistic interval

A *linguistic interval* $I_X^{p,q}(a)$ is an ordered set of consecutive pairs of linguistic labels and is the linguistic representation of a real value a fuzzified into the fuzzy sets of the corresponding linguistic variable X:

$$I_X^{p,q}(a) = \begin{cases} \left\{A_X^p : \mu_{A_X^p}(a);\ A_X^q : 1 - \mu_{A_X^p}(a)\right\} & Si\ p \neq q \\ \{A_X^p : 1\} & Si\ p = q \end{cases} \tag{1}$$

where p and q are the position in X of the two linguistic labels which make up the interval and $\mu_{A_X^p}(a)$ has to be always between zero and one. For example, if a macroblock is into column $7 (c = 7)$, the result of fuzzificating that value into the variable *HP* (Fig. 4) is $I_{HP}^{2,3}(7) = \{Left : 0.5\ ;\ Horizontal\ Centre\ :\ 0.5\}$.

4.3. Linguistic motion vector

A *linguistic motion vector* is the fuzzy representation of a motion vector belonging to a macroblock. It represents a linguistic description of the motion of a macroblock between consecutive frames and is defined as:

$$LMV = \langle FN, I_{HP}(mb_{row}), I_{VP}(mb_{col}), I_{HV}(mv_{right}), I_{VV}(mv_{down}) \rangle \tag{2}$$

where the first element identifies the frame number where the motion vector is and the next four elements are four linguistic intervals. The I_{PH} and I_{PV} intervals represent the column and row of the macroblock. The I_{HV} e I_{VV} intervals are obtained from the *right_x* and *down_x* values of the motion vector. Table 2 presents an example of a linguistic motion vector.

A *valid linguistic motion vector* is a linguistic motion vector that contains relevant information about the direction and velocity of an object, i.e., at least one of the two magnitudes concerning the velocity (I_{HV} and I_{VV}) of the vector is distinct of the label *No Motion*. The linguistic motion vector shown on Table 2 is a valid one because the I_{HV} interval contains the label *Fast Left*.

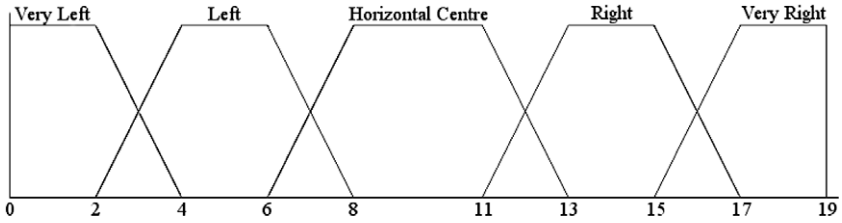

Fig. 4. *Horizontal Position* linguistic labels.

Table 2
Example of a linguistic motion vector.

Frame number	23
$I_{HP}(3)$	{*Very Left*: 0.5; *Left*, 0.5}
$I_{VP}(3)$	{*Very Up*: 0.25; *Up*, 0.75}
$I_{HV}(45)$	{*Fast Left*, 1.0}
$I_{VV}(0)$	{*No Motion*, 1.0}

Table 3
Example of a linguistic blob.

Frame number	23
Size	8
Macroblocks	{21; 22; 23; 41; 42}
I_{HP}	{*Very Left*: 0.75; *Left*: 0.25}
I_{VP}	{*Very Up*: 0.82; *Up*: 0.18}
I_{HV}	{*Fast Right*: 1.0}
I_{VV}	{*No Motion*: 0.90; *Slow Down*: 0.10}

4.4. Linguistic blob

A *linguistic blob* is an 7-tuple composed by one or more conceptually similar motion vectors that represents in a linguistic way a region (*blob*) in a frame. In fact, the output of the moving object segmentation is a list of blobs ideally corresponding to the moving regions in each frame. A linguistic blob is defined as:

$$LB = \langle FN, Size, MBs, I_{HP}, I_{VP}, I_{HV}, I_{VV} \rangle \tag{3}$$

where *FN* is the frame number in which is located the blob, *Size* is the number of linguistic motion vectors grouped in the blob, *MBs* is a list of macroblocks belonging to the blob and the last four elements are the linguistic intervals that represent the position and velocity of the blob. Table 3 presents an example of a linguistic blob.

5. Moving object segmentation

In this work, the real displacement between frames is computed by using a fuzzification procedure working on motion vectors, neither DCT coefficients information nor statistical filtering is needed; so the algorithm requires very little of data; from a frame size of 320 × 240 pixels and about 35–40 KB, the algorithm only needs about 2–4 KB and around 18 KB per frame for a 640 × 480 video sequence with a frame size of 120–140 KB.

Fig. 5 shows the overview of the proposed system. The H.264 compressed video stream is the input of the algorithm. The first stage is the decoding of the video streaming with the goal of extracting the motion vectors and the decision modes. In the next stage, the information extraction is carried out by processing the H.264 decoded data. This stage has several steps. First, a k-neighbor algorithm is applied to the decision mode matrix to obtain the motion vectors to take into account. After that, in the fuzzification step, motion vectors are converted into linguistic motion vectors in order to obtain the fuzzy representation of them. Then the noisy information is removed and the valid linguistic motion vectors are obtained. The algorithm rules out the motion vectors with low values of *right_x* and *down_x* because they could be noisy and they do not provide information about moving objects. Next stage is the clustering algorithm, i.e., the valid motion vectors are grouped into linguistic blobs, each of them could be identified as a moving object of the video scene. Finally, the linguistic blobs are filtered to delete noisy ones. Note that the result of the algorithm is not a set of objects detected in a video sequence, but a set of linguistic blobs or regions which form the previous step to the identification of the objects. In a parallel process to the segmentation, the fuzzy sets defined over each linguistic variable are modified in a dynamic way to adapt the algorithm to a specific scenario, as it is explained in Section 5.4. Thus, the linguistic labels are modified in accordance with the size of the blobs identified in the scene and this fact has influence on the performance of the algorithm. This is the fourth and last stage of the segmentation approach.

5.1. Decoding the video streaming

The segmentation method has been developed by using the H.264/AVC JM 14.2 reference software [12]. The *baseline profile* is chosen for two reasons: first of all, this profile would not have license payments required for a future commercial implementation; and second, because it is the common profile used in most of the real-time applications, such as video and mobile TV, video conference and video surveillance systems, among others; since no buffers are needed for storing the frames and re-ordered them. Moreover, the presented approach only uses I and P frames to carry out the segmentation due to the encoding and decoding process using other frame types (B, SP and SI) is slow and can not satisfy the real-time

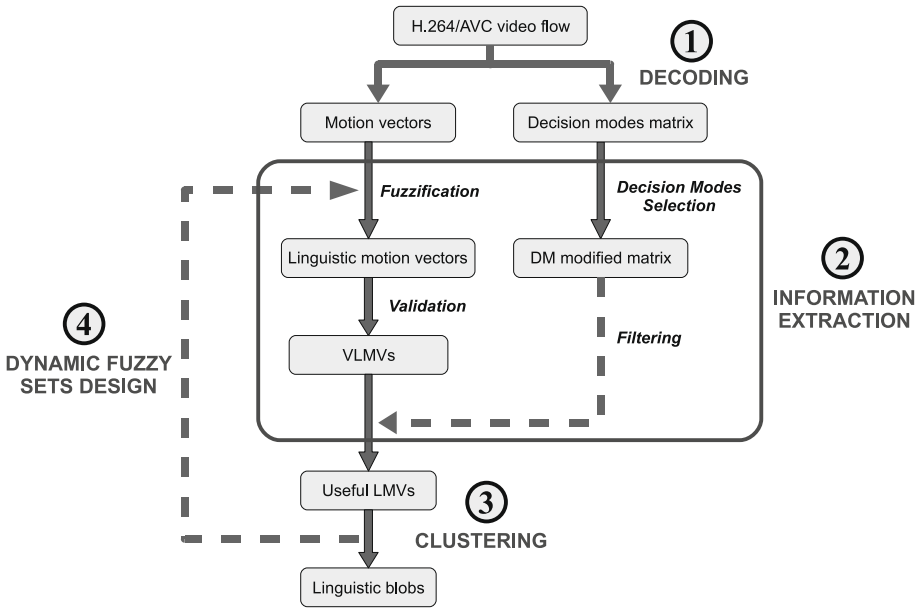

Fig. 5. Overview of the proposed system.

Table 4
Values of the H.264 codec configuration parameters.

Parameter	Value	Description and justification
FrameRate	25	Frames per second
SourceWidth	640	Width of frame in pixels
SourceHeight	480	Height of frame in pixels
ProfileIDC	66	Baseline Profile
IntraPeriod	12	GOP format is I11(P) (one I-frame each 12 P-frames)
UseHadamard	0	To speed up the decoding process
DisableSubpelME	1	Disable subpixel motion estimation
SearchRange	32	Maximum search range to the motion vectors
NumberRefFrames	1	Use only the previous frame as reference
NumberBFrames	0	B-frames are not used
RDOptimization	0	Low complexity mode
SearchMode	0	Fast full search

requirements. Also, the *Rate-distortion Optimization* option is disabled [11]. Other configuration parameters are shown in Table 4.

The H.264/AVC JM reference software encoder selects the best macroblock mode by using the *Sum of Absolute Errors* (SAE). This fact implies that for each existing macroblock partition (and sub-partition) within the MB, a predictor within the pixel-domain is created from the motion estimation of the current partition, and the SAE cost is evaluated. In other words, the original block and the predicted one are compared: the difference pixel by pixel in absolute value is calculated. The best mode is determined corresponding to the mode exhibiting the minimum SAE cost; the minimum sum of the absolute differences. One of the main advantages of this method is its low computational cost, so it is recommendable for real-time use.

5.2. Information extraction: decision modes selection

The second stage of the segmentation approach is the information extraction. In this section, one of the steps belonging to this stage is analyzed; the selection of the valid decision modes is specifically presented. Table 1 shows that the most usual decision modes to partition a macroblock are 0 and 1 modes, i.e., *skip* macroblocks and 16×16 macroblocks. This information is not useful to know the motion activity in a region of a frame because the H.264 encoder by using the minimum SAE cost do not use these modes to encode moving objects, thus the motion vectors in these kinds of macroblocks are removed. This fact is related with the features of the H.264 encoder because it is designed to work with high quality precision. A car or a person have a lot of details, so they will be encoded with partitioned macroblocks. Besides skipped macroblocks do not have motion vectors, so no motion vectors can be obtained from them. Referring to 16×16 MBs, on the one hand, they

are not usually employed to encode moving objects, only in exceptional occasions. On the other hand, 16 × 16 MBs are about 40% of the total (Table 1), so their processing time would waste a considerable amount of time and then the algorithm could not fulfill the real-time requirement.

The segmentation algorithm has to choose the valid motion vectors to obtain the right blobs. It can be selected between two ways of decision modes selection. In the first one, the algorithm takes into account only motion vectors belonging to a **macroblock of type 4** (four 8 × 8 partitions) because it has been observed experimentally that the moving objects are usually coded with this decision mode (Fig. 6a). Table 5 shows the distribution of each decision mode on all the frames and on moving regions. The results obtained show that 87.47% of macroblocks inside the moving regions are codified with decision mode 4 and the 6.50% of all the macroblocks are codified with this decision mode, i.e., the algorithm could identify the 87.47% of macroblocks belonging to moving regions only by processing the motion vectors of the 6.50% of macroblocks.

In the second way, it is obtained the decision modes matrix corresponding to a frame and it is applied a **k-neighbor algorithm** to obtain the implicit motion degree in a region of the frame. The decision modes has been named in increasing motion degree, thus the higher is the k-neighbor macroblock value, the more is the movement inside the macroblock (Fig. 6b). In this case, the segmentation procedure works only with motion vector values greater than a predefined threshold U_{kn} in the k-neighbor algorithm. This threshold discriminates between motion vectors belonging to high-numbered decision modes and neighbors and it helps to reduce noise because it allows to rule out isolated motion vectors. For example, in Fig. 6b, threshold U_{kn} rule out motion vectors belonging to macroblocks drawn in white or light gray. In Section 6 the results using both decision modes selection are analyzed.

Fig. 7 shows an example of modes selection. Fig. 7a shows the original frame of a surveillance video and in Fig. 7b the decision modes are drawn (light gray macroblocks are skipped and dark gray ones are intra-frames). Fig. 7c shows macroblocks encoded with the fourth decision mode and Fig. 7d shows macroblocks in which the k-neighbor algorithm obtains a value greater than the threshold U_{kn}. So depending of the decision modes selection way, the motion vectors to analyze will be different.

5.3. Clustering algorithm

After fuzzification of motion vectors, the proposed system clasiffies the valid linguistic motion vectors into linguistic blobs through a *clustering* algorithm. If there is a linguistic motion vector suitable to belong to a blob, it is included into

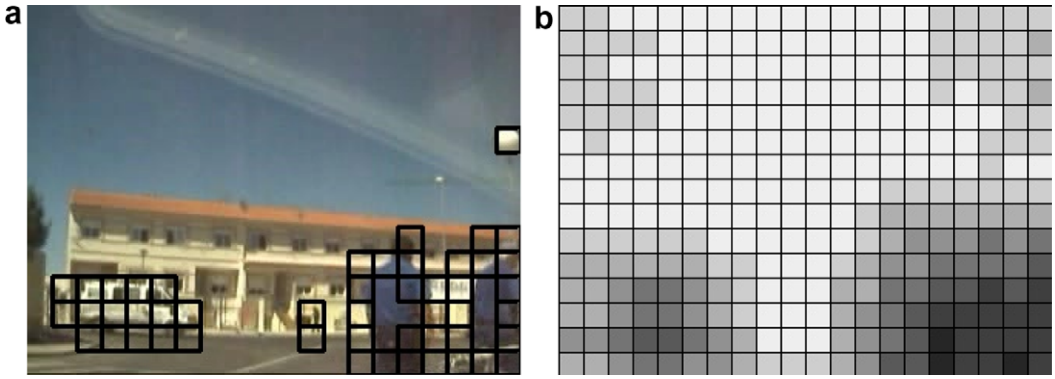

Fig. 6. Decision modes selection: (a) drawn the macroblocks encoded with the fourth decision mode and (b) degree of the values of the decision modes matrix by using the k-neighbor algorithm.

Table 5
Frequency of each decision mode in the full frames (Frequency 1) and in moving regions (Frequency 2).

Sequence	Lorry sequence		Shop window sequence		Street sequence	
	Frequency 1 (%)	Frequency 2(%)	Frequency 1 (%)	Frequency 2 (%)	Frequency 1 (%)	Frequency 2 (%)
Resolution	320 × 240		640 × 240		640 × 480	
Frames	1300		652		5200	
0	70.44	0.00	22.6	0.78	31.35	0.00
1	15.55	5.38	50.6	3.88	52.38	4.93
2	0.35	4.79	3.3	3.14	0.43	0.45
3	0.22	2.39	2.2	0.78	0.40	0.45
4	3.1	85.63	10.7	82.60	5.70	94.17
8	0.00	0.00	0.4	0.00	0.00	0.00
9	4.86	1.80	4.8	8.80	6.12	0.00
10	4.46	0.00	1.6	0.00	2.61	0.00
13	1.02	0.00	3.7	0.00	1.03	0.00

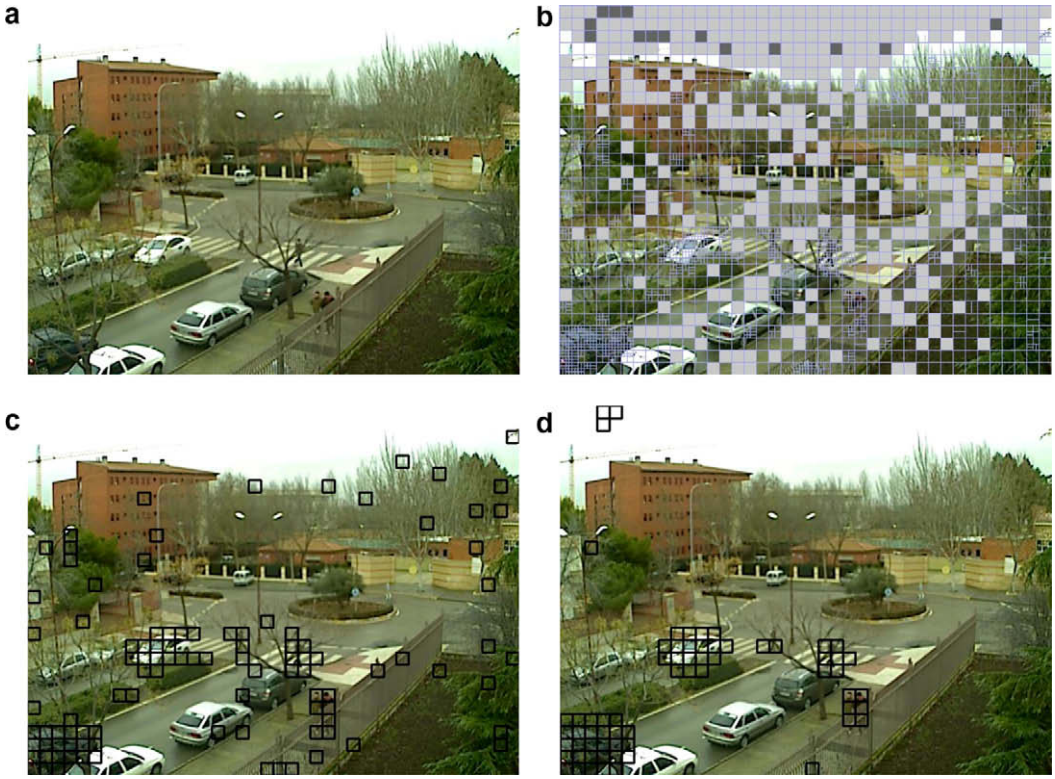

Fig. 7. Decision modes selection: (a) original frame, (b) decision modes matrix, (c) macroblocks encoded with the forth decision mode and (d) macroblocks selected by using the k-neighbor algorithm.

the linguistic blob and is modified the representation of this one according to the weighted aggregation defined in this section. The similarity between a vector and a blob is calculated with a distance measure based in the numbering order of the defined linguistic labels over a variable [21,22]. Then, a **Euclidean distance** between two linguistic intervals is defined as:

$$D_X = D\left(I_X^{p,q} - I_X^{r,s}\right) = \left|\frac{p+q}{2} - \frac{r+s}{2}\right| \tag{4}$$

For example, considering the linguistic variable *HP* (Fig. 4), the distance between the intervals:

$$I_{HP}^{2,3}(7) = \{Left : 0.5\ ;\ Horizontal\ Centre : 0.5\}$$
$$I_{HP}^{3,3}(9) = \{Horizontal\ Centre :\ 1.0\}$$

will be:

$$D_{HP}\left(I_{HP}^{2,3}(7), I_{HP}^{3,3}(9)\right) = \left|\frac{2+3}{2} - \frac{3+3}{2}\right| = 0.5$$

The motivation of using this Euclidean distance is to enhance the robustness of the segmentation approach. The nature of the proposed equation allows to incorporate expert knowledge into the scenario: the linguistic variables can be designed to adapt themselves to the application domain. For example, if there is a scenario with two doors, the values of the position linguistic variables can be adapted to this situation, as shown in Fig. 8. The distribution of the linguistic labels allows to recognize the position of an object and the Euclidean distance increases the relevance of each linguistic label in function of its size. However, if the segmentation algorithm use a support-based distance, the distance does not mind the design of the linguistic labels. It has been experimentally proved that the best performance is achieved by using a Euclidean distance and it improves the understanding of the linguistic representations.

Since this Euclidean distance, the **total distance** between a linguistic motion vector $VLMV_x$ and a blob $LB_y(TD(VLMV_x, LB_y))$ is defined as the maximum of the differences between the four Euclidean distances referring the linguistic intervals I_{HP}, I_{VP}, I_{HV} and I_{VV}:

$$TD = max(\mu_{PD}(D_{HP}), \mu_{PD}(D_{VP}), \mu_{VD}(D_{HV}), \mu_{VD}(D_{VV})) \tag{5}$$

where *PD* and *VD* are two linguistic variables named *position difference* and *velocity difference*. These variables measure the membership of the Euclidean distance to the *difference* label. It has been experimentally proved that the use of both variables

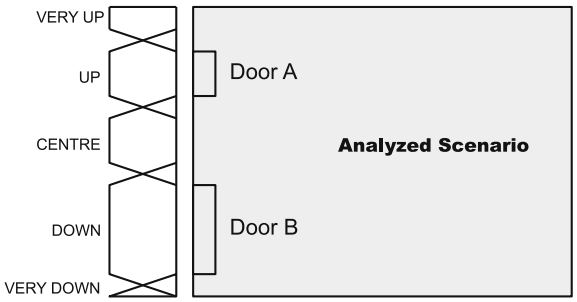

Fig. 8. Linguistic labels of different size incorporate expert knowledge.

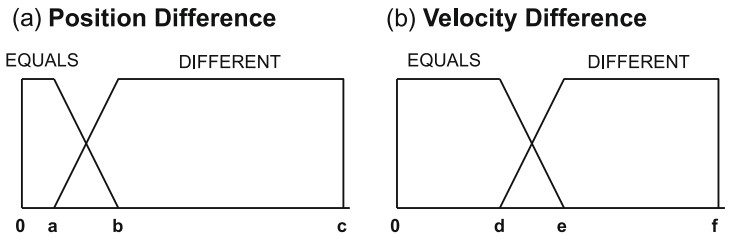

Fig. 9. *Position difference* and *velocity difference* variables.

reduce the uncertainty associated with the H.264 encoding. This fact occurs due to the motion compensation: the motion vectors associated to each macroblock (or macroblock partition) can be slightly wrong to our purpose and not indicate the real displacement of the corresponding object, but the features of the encoding have to be considered. In this way, the total distance between $VLMV_x$ and LB_y have into account the sensitivity of the encoding defining two linguistic variables which are more tolerant to velocity differences than position differences. Fig. 9 shows the linguistic labels of the difference variables where it is important to underline that the values have to fulfil the next requirements: $a < d$ and $b < e$.

A vector $VLMV_x$ will be added to the blob LB_y, i.e., will be considered in the same region, if the next conditions are satisfied:

(1) The total distance between the vector $VMLV_x$ and the blob LB_y is the least that any total distance between another linguistic motion vector and LB_y.
(2) $TD(VLMV_x, LB_y)$ is less than a threshold U_{mv} since a vector is considered susceptible to belong a blob. This threshold is a value between 0 and 1 and the optimum value is around 0.2. U_{mv} is different in function of the scene: a low value means high discrimination condition, so over-segmentation ratio could be increased, and a high value of U_{mv} means a relaxed clustering condition, so merge ratio could be increased. The optimum value should minimize both merge and over-segmentation ratios.

The **weighted aggregation** of $VLMV_x$ into the linguistic blob LB_y generates the modified representation of the blob LB'_y and is executed in four stages:

(1) $FN(LB'_y) = FN(LB_y)$, i.e., the frame number is the same.
(2) $size(LB'_y) = size(LB_y) + 1$, i.e., the blob size is increased.
(3) $MBs(LB'_y) = MBs(LB_y) + MB(VLMV_x)$, i.e., the macroblock identificator of the motion vector $VLMV_x$ is added to the macroblocks list of BL'_y.
(4) The linguistic intervals I_{HP}, I_{VP}, I_{HV} and I_{VV} are modified according to a *weighted union* that evaluates the size of the blob:

$$\mu_{A_j^i}(I'_j(LB_y)) = \frac{\mu_{A_j^i}(I_j(LB_y)) \cdot size(LB_y)}{size(LB_y) + 1} + \frac{\mu_{A_j^i}(I_j(VLMV_x))}{size(LB_y) + 1} \tag{6}$$

For example, considering the linguistic intervals referring to the *Horizontal Position* variable:

$I_{HP}(LB_y) = \{Very\ Left : 0.5;\ Left : 0.5\}$
$I_{HP}(VLMV_x) = \{Left : 1.0\}$

and $size(LB_y) = 8$, the weighted union between the intervals is $\{Very\ Left : 0.44\ ; Left\ :\ 0.56\}$:

$$I_{HP}(LB'_y) = \left\{ Very\ Left : \frac{0.5 \cdot 8}{9} + \frac{0}{9};\ Left : \frac{0.5 \cdot 8}{9} + \frac{1.0}{9} \right\}$$

The presented weighted aggregation gives a relevant value to the size of the linguistic blob and it is fundamental to get the conceptual representation of the modified blob.

5.4. Dynamic design

The segmentation algorithm proposed presents the typical problems of any segmentation system: **merge** (two or more objects are identified as one; it could happen with little objects) and **split** (one object is identified as two or more; it could happen with big objects). One of the reasons of these problems come from the static design of the linguistic variables used in the algorithm, i.e., their values remain unaltered for different kinds of objects. Since the definition of the total distance and the weighted aggregation, this segmentation system has a rigid behaviour and cannot adapt itself to the nature of the objects. Besides, the initial version of the algorithm is strongly dependent of the expert knowledge because the fuzzy sets defined over each linguistic variable are manually established and their values are invariables. Since the application is designed for real traffic scenarios, the linguistic variables are initially configured for the most frequently appeared object in that domain, i.e., the car. However, there could be a merge problem if an object little than a car (for example, a person) appears and there could be an split problem if a big object like a truck appears on the video. Then, if the number and the values of the fuzzy sets are selected in a wrong way, the results of the algorithm will be poor.

For this reason, this algorithm introduces the dynamic design of the linguistic labels to adapt them to each specific scenario, where the *dynamic design of a linguistic variable X* is understood as the adaptation and modification of the values of the fuzzy sets belonging to X during the execution of a fuzzy procedure. Thus, the segmentation procedure allows to give itself some feedback to improve its results. The algorithm has an initial configuration, but it could change the values of the fuzzy sets in function of the obtained results taking into consideration the size and shape of the obtained blobs in previous frames. So the automatic changes allow to improve the quality of the algorithm because they have positive influence on the merge and split rates, as shown in Figs. 10 and 11.

The dynamic design of the fuzzy sets is based on some common characteristics of the fuzzy systems like the **support set**, the **height** of a fuzzy set and the **alpha cut threshold** [3]. Being X a linguistic variable, A a fuzzy set of X, D the domain of X and α the threshold of the alpha cut set:

$$support\ (A) = \{x \in D : \mu_A(x) > 0\} \tag{7}$$

$$height\ (A) = max(\mu_A(x)) \tag{8}$$

$$alpha\_cut_\alpha\ (A) = \{x \in D : \mu_A(x) \geqslant \alpha\} \tag{9}$$

The linguistic variables whose values are updated dynamically are HP and VP. Each variable X has a set of fuzzy sets $\{A_1, A_2, \ldots, A_{size}\}$ over his domain D_{HP} or D_{VP}. Only normal fuzzy sets are used, so the height of them is always equal to one. The algorithm is ready to work with both triangular and trapezoidal fuzzy sets. The support of each fuzzy set A is $support(A) = alpha\_cut_1(A) + 4$, except for the first and the last sets (A_1 and A_{size}) of a linguistic variable, where $support(A) = alpha\_cut_1(A) + 2$.

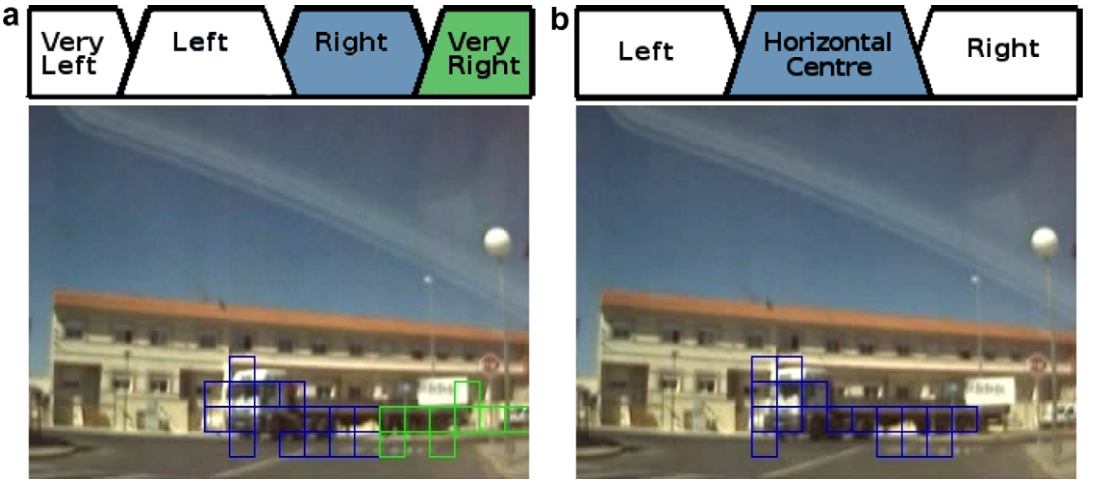

Fig. 10. Dynamic fuzzy sets avoid the merge problem.

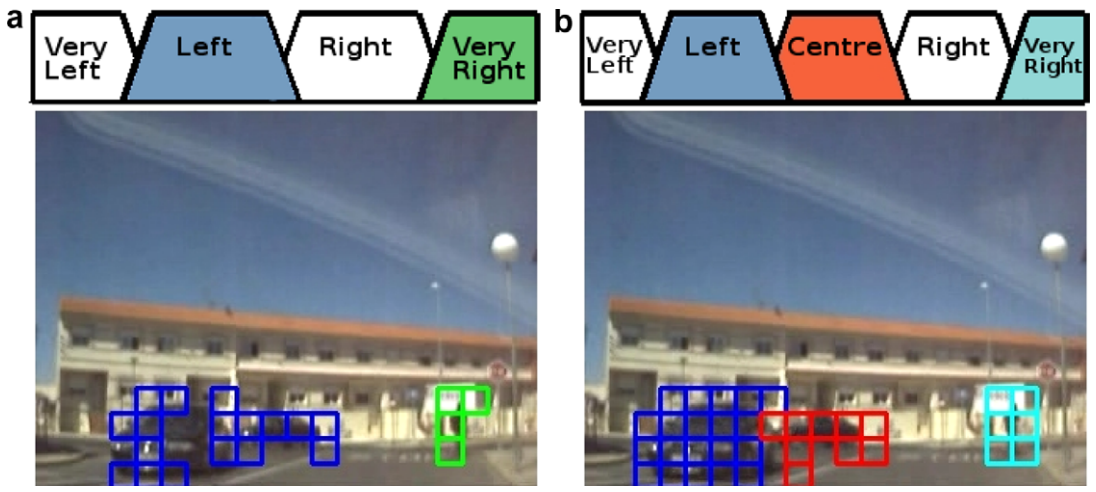

Fig. 11. Dynamic fuzzy sets avoid the split problem.

The dynamic design of the fuzzy sets is executed at the end of the segmentation algorithm, as shown in Fig. 5. It is only processed to the biggest blob detected in the frame, if there is at least one blob detected in said frame. Being the macroblocks $MBs = \{MB_1, MB_2, \ldots, MB_m\}$ belonging to the biggest blob LB, it can be calculated the **rectangular size** of the blob by:

$$height(LB) = row(MB_m) - row(MB_1) + 1 \tag{10}$$

$$width(LB) = max_i(col(MB_i)) - min_i(col(MB_i)) + 1 \tag{11}$$

where *col* is the column of the macroblock, *max* is the maximum of the values -in this case, the maximum of the columns of all the macroblocks in *LB*- and *min* is the minimum value, i.e., the lowest value of the columns.

The system calculates the **number** of the new fuzzy sets defined to the linguistic variables *HP* and *VP* in function of the size of the biggest blob in the frame. Being *W* and *H* the width and the height in macroblocks of each frame, the number of the redesigned linguistic labels can be calculated as $size(HP) = W/width(LB)$ and $size(VP) = H/height(LB)$. The **names** of the linguistic labels of *HP* and *VP* are got from two configuration files that have a correspondence between the number of labels and the names of them. After that, if trapezoidal fuzzy sets are used, it is necessary to calculate the support and the alpha-level set with threshold 1 to define the **shape** of the corresponding fuzzy sets. Since $size(HP)$ and $size(VP)$, it can be calculated the size of the alpha-level set with threshold value 1 with the same size to all the fuzzy sets for each linguistic variable. Being A_{HP} a fuzzy set of *HP* and A_{VP} a fuzzy set of *VP*, the size of the alpha cut set is calculated as:

$$size(alpha\_cut_1(A_{HP})) = \frac{(W-1) - 2 * (size(HP) - 1)}{size(HP)} \tag{12}$$

$$size(alpha\_cut_1(A_{VP})) = \frac{(H-1) - 2 * (size(VP) - 1)}{size(VP)} \tag{13}$$

For example, being the fuzzy sets of the linguistic variable *HP* shown in Fig. 4, if the algorithm finds a full region in a frame which a width of six macroblocks ($width(LB) = 6$), the module which updates the values of the linguistic variable *HP* will change its values. First, it calculates the number of the labels as $size(HP) = W/width(LB) = 20/7 \backsimeq 3$ and reads the names of them in the configuration file (*LE*, *HC* and *RI*). Then, the size of the alpha cut set and the support of each set are calculated as:

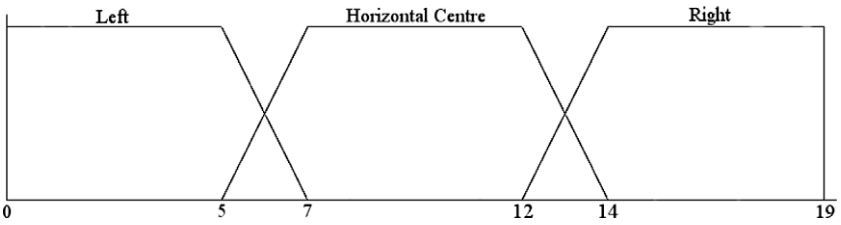

Fig. 12. New *Horizontal Position* linguistic labels.

$$size(alpha\_cut_1(A_{HP})) = \frac{20 - 1 - 2 * (3 - 1)}{3} = 5$$
$$support(A_1) = alpha\_cut_1(A) + 2 = 5 + 2 = 7$$
$$support(A_2) = alpha\_cut_1(A) + 4 = 5 + 4 = 9$$

The module can redefine the values of the linguistic variable *HP* with this information and obtains the new values, which are shown in Fig. 12.

6. Experimental results

Segmentation is an ill-posed problem: for the same video, the optimum segmentation can be different depending on the application. Furthermore, the goal of the segmentation process can change in function of the application domain; for example, object detection should be fast in real-time applications or accurate in medical domain. A lot of criteria and measures have been proposed [5,15]. Some of them are based on the fidelity of the detected objects like shapes and edges; others work at pixel level like normalized uniformity measures and others calculate the penalties of the segmentation approaches based on misclassification, shape or motion penalty.

In this context, designing a good measure for segmentation quality is a hard problem. The ideal evaluation is the objective evaluation, i.e., the unsupervised methods that do not require reference image and can be used in a real-time system [31]; however, these algorithms evaluate the quality of the segmentation through mathematical measures like the normalized uniformity measure or the contrast. These measures can not be applied to the algorithm because it does not work with DCT coefficients or pixel properties, but it only obtains the moving regions in a frame. Then, in this work a supervised method to evaluate the segmentation quality is used.

Two measurements are used to describe the segmentation performance. The first measurement is *detection possibility* that is defined as the percentage of the real regions detected. The second one is *precision* that measures the percentage of the detected objects corresponding to real objects. Then, high detection possibility means less miss-segmentation and high precision means less false segmentation. Besides, two measurements are defined to analyze the *processing time* and the *segmentation size*, i.e., the temporal and spatial requirements of the algorithm.

The segmentation approach has been evaluated on several video sequences compressed using the H.264 encoder of JM 14.2. The encoder configuration has been set with *baseline profile* and low complexity mode on RD Optimization. The video tested resolution varies from 320×240 to 640×480 pixels. The specific configuration parameters of the algorithm

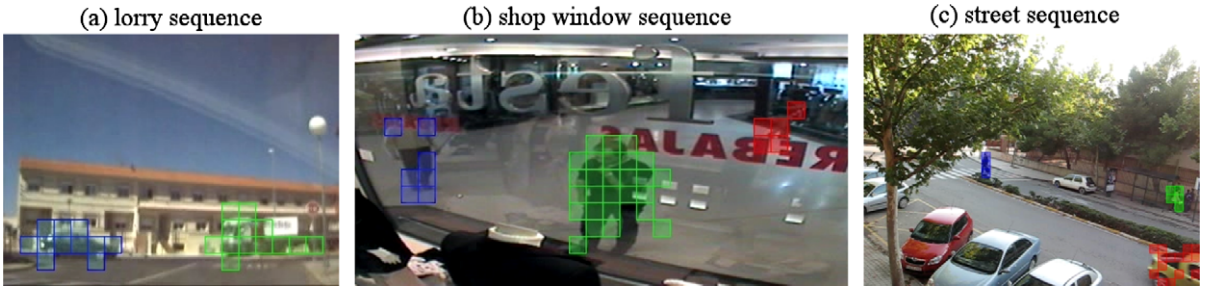

Fig. 13. Snapshots of three video sequences with the detected blobs drawn.

Table 6
Improvement using dynamic design.

Measurement	Lorry sequence		Shop window sequence		Street sequence	
	Static	Dynamic	Static	Dynamic	Static	Dynamic
Resolution	320×240		640×240		640×480	
Frames	1300		652		5200	
Ground truth	353	353	808	808	1355	1355
Detected blobs	378	362	649	684	1156	1138
True positives	293	302	631	665	996	1077
False positives	85	60	18	19	160	61
False negatives	60	51	177	143	359	278
Detection possibility (%)	82	85	78	82	74	80
Precision (%)	78	83	97	97	86	95
Processing time (ms)	17.4	19.7	25.5	32.7	37.2	41.1

(U_{mv}, U_{kn} and the minimum blob size) change depending on the application scenario. Fig. 13 shows three snapshots of three different video sequences tested where the detected blobs are drawn over the frames. These snapshots are referred to the results shown in Table 6, where the improvement of the algorithm using dynamic design of the fuzzy sets can be noted.

As it has been explained in Section 5.2, the algorithm can be executed in two modes depending of the decision modes selection: only *type 4* macroblocks or *k-neighbor algorithm* with a threshold. Table 7 shows the differences between both modes in several experiments (320×240 pixel resolution) using the same configuration parameters. It can be seen that one of them is faster but its performance is worse. Thus the choice will depend on the application field.

Fig. 14 shows nine characteristic frames from a surveillance scene. Each one of them shows a different interesting situation, as detailed next. Fig. 14a shows three groups of people detected as moving objects and a moving sheet of a tree on the top right corner. In Fig. 14b, four moving objects are detected; the blue one is a car appearing in the scene and the segmentation algorithm recognizes it behind the tree. Besides, there is a person not detected on the left of the green blob, so false negative is 1/5 in this frame. Fig. 14c shows five blobs detected; the blue blob is a car appearing in the scene. In Fig. 14d, the segmentation algorithm recognizes two blobs – magenta and red – in the same region because people are walking in the opposite direction. In Fig. 14e, there is a bike on the road (blue blob). Fig. 14f illustrates a frame with two detected blobs and the green one is a person behind the fence. Fig. 14g and f show three cars detected as different blobs in consecutive frames. Finally, in Fig. 14i there is a representative frame with different type of objects: a lorry, a van, a car and a person. There is a merge problem because a person is recognized in the same blob that the van.

Table 7
Decision modes selection.

Measurement	Type 4	k-Neighbor
Detection possibility (%)	73	81
Precision (%)	78	89
Processing time (ms)	16.2	18.6
Segmentation size (bits/frame)	148	103

Fig. 14. Snapshots with the detected blobs drawn in a video surveillance scene (frames 16, 90, 235, 285, 483, 573, 1093, 1111 and 3100). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

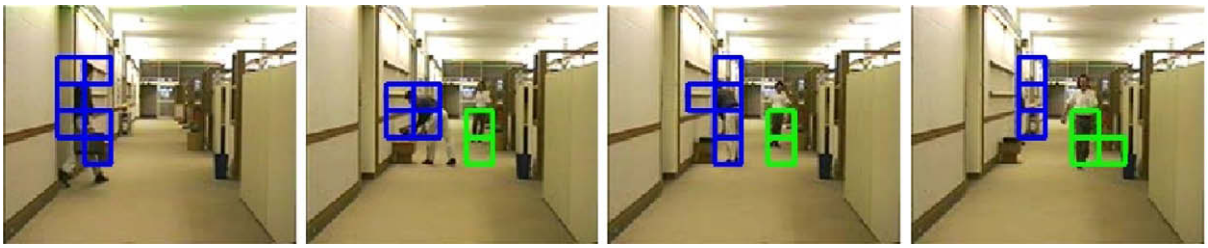

Fig. 15. Segmented frames in *Hall Monitor* sequence (frames 29, 106, 138 and 243).

Fig. 16. Segmented frames in *Erik* sequence (frames 10, 22, 37 and 44).

Table 8
Average processing time.

Frame resolution (pixels)	Processing time (ms)
320×240	20
640×240	26
640×480	38

The segmentation approach has been also tested with other standard surveillance videos like *Hall Monitor* and *Erik* sequences and compared with other approaches. Fig. 15 shows results obtained in four different frames in *Hall Monitor* sequence: segmentation process recognizes moving objects but do not extract well the boundary of the objects since this sequence has a QCIF resolution. However, the processing time is near to 8 ms The approach shown in [30] has worse temporal performance than the proposed method: about 300–700 ms per frame in a CIF (352×288) sequence and about 50–80 ms in a SIF (352×240) sequence with not much movement. In the second approach [13], the processing time is about 32 ms in CIF sequences, however this method uses spatiotemporal information, i.e., it uses last frames data while processing a frame. With respect to the quality of the results, the average precision and recall are 92.9% and 96.6% for Zeng approach [30] in *Erik* sequence and 92.2% and 98.3% for Liu approach [13]. In the presented approach, the average precision is 91.4% and the recall is 96.8%. Fig. 16 shows results obtained in *Erik* sequence.

The hardware used in the experiments is an Intel Pentium IV Core 2 Duo T7100 1.8 GHz with 2 GB RAM memory. Table 8 shows average processing time depending on the frame resolution. Since a 320×240 frame is processed in around 20 ms and a 640×480 one in around 38 ms, it can be ensured that the segmentation works in real-time, i.e., at least 25 fps.

7. Conclusions

This work presents a real-time segmentation approach of moving objects with the next features. First of all, the algorithm is designed for H.264/AVC compressed domain due to its excellent compression efficiency and its widespread field of multimedia applications. The algorithm can work in real-time and requires very little of data because it uses only the motion vectors field and the decision modes to carry out the segmentation. The proposed approach is based on fuzzy logic to detect the moving objects; fuzzy logic allows to avoid the noise inherent to the encoding process and to obtain conceptual representations that describe the regions detected in a comprehensive way. Finally, the segmentation performance is robust because it can adapt itself to the application scenario in a dynamic way updating in real-time the values of the fuzzy sets. This feature allows to improve the merge and split rates.

As future work it is planned to improve the performance of the segmentation algorithm over changing situations. The initial version of the algorithm was strongly dependent of the expert knowledge: the values of the configuration parameters

and the design of the fuzzy sets had to be established by an expert before the execution of the segmentation process. In this work, an improvement to update dynamically the values of the fuzzy sets is presented and tested. So one of the future work should be to get the configuration parameters updated in real-time. In this way, a matching learning algorithm could be developed to identify the different types of moving objects and the usual trajectories of each of them. Finally, another research line is the management of the velocity linguistic variables and the corresponding velocity of the detected moving objects. The horizontal and vertical linguistic intervals of a blob show the fuzzy values of the motion vector directions, i.e., the displacement associated with each macroblock (or macroblock partition) due to the encoding process. However, it would be more interesting to obtain the real displacement of the moving objects in function of their position respect to the camera location.

Acknowledgments

This work was supported by the Ministry of Science and Technology of Spain under CONSOLIDER Project CSD2006-46, CICYT Project TIN2006-15516-C04-02 and CENIT Project HESPERIA and the Council of Science and Technology of Castilla-La Mancha under Projects PEII09-0037-2328 and PII2I09-0045-9916.

References

[1] J.A. Albusac, J.J. Castro-Schez, D. Vallejo. Learning maximal structure rules with pruning based on distances between fuzzy sets, in: Proceedings of the Information Processing and Management of Uncertainty in Knowledge-based Systems, IPMU'08, 2008, pp. 441–447.
[2] J.L. Barron, J.D. Fleet, S.S. Beauchemin, Performance of optical flow techniques, International Journal of Computer Vision 12 (1) (1994) 43–77.
[3] E. Cox (Ed.), The Fuzzy Systems Handbook, 2nd ed., AP Professional, 1998.
[4] C.D. Creusere, G. Dahman, Object detection and localization in compressed video, in: Asilomar Conference on Signals, Systems and Computers, vol. 1, Pacific Grove, California, USA, 2001, pp. 93–97.
[5] E.D. Gelasca, T. Ebrahimi, M. Karaman, T. Sikora, A framework for evaluating video object segmentation algorithms, in: Computer Vision and Pattern Recognition Workshop, CVPRW'06, 2006, pp. 198–210.
[6] H.L. Eng, K.K. Ma, Spatiotemporal segmentation of moving video objects over MPEG compressed domain, in: Proceedings of the IEEE International Conference on Multimedia and Expo, vol. 3, New York, USA, 2000, pp. 1531–1534.
[7] Y.W. Huang, C.H. Chen, C.H. Tsai, C.F. Shen, L.G. Chen, Survey on block matching motion estimation algorithms and architectures with new results, Journal of VLSI Signal Processing Systems 42 (3) (2006) 297–320.
[8] R.T. Iqbal, C. Barbu, F. Petry, Fuzzy component based object detection, International Journal of Approximate Reasoning 45 (3) (2007) 546–563.
[9] M.L. Jamrozik, M.H. Hayes, A compressed domain video object segmentation system, in: Proceedings of the IEEE International Conference on Image Processing, vol. 1, New York, USA, 2002, pp. 113–116.
[10] X. Ji, Z. Wei, Y. Feng, Effective vehicle detection technique for traffic surveillance systems, Journal of Visual Communication and Image Representation 17 (3) (2006) 647–658.
[11] Joint Video Team (JVT), Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification, ITUT Recommendation H.264 and ISO/IEC 14496/10 AVC, 2003.
[12] Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, Reference Software to Committee Draft. JVT-F100 JM14.2, 2008.
[13] Z. Liu, Y. Lu, Z. Zhang, Real-time spatiotemporal segmentation of video objects in the H.264 compressed domain, Journal of Visual Communication and Image Representation (18) (2007) 275–290.
[14] P. Matsakis, J.M. Keller, L. Wendling, J. Marjamaa, O. Sajhputera, Linguistic description of relative positions in images, IEEE Transactions on Systems Man and Cybernetics 31 (4) (2001) 573–588.
[15] K. McKoen, R. Navarro-Prieto, B. Duc, E. Durucan, F. Ziliani, T. Ebrahimi, Evaluation of video segmentation methods for surveillance applications, in: Proceedings of the European Signal Processing Conference, EUPSICO'00, 2000, pp. 1045–1048.
[16] V. Mezaris, I. Kompatsiaris, N.V. Boulgouris, M.G. Strintzis, Real-time compressed-domain spatiotemporal segmentation and ontologies for video indexing and retrieval, IEEE Transactions on Circuits and Systems for Video Technologies (14) (2004) 606–621.
[17] R. Munoz-Salinas, R. Medina-Carnicer, F.J. Madrid-Cuevas, A. Carmona-Poyato, Multi-camera people tracking using evidential filters, International Journal of Approximate Reasoning 50 (5) (2009) 732–749.
[18] M. Nachtegael, E. Kerre, S. Damas, D. Van der Weken, Special issue on recent advances in soft computing in image processing, International Journal of Approximate Reasoning 50 (1) (2009) 1–2.
[19] PETS Data Sets, IEEE International Workshop on Performance Evaluation of Tracking and Surveillance, Available in <ftp://ftp.pets.rdg.ac.uk/pub/>, 2002 (last access: 03.02.09).
[20] I.E.G. Richardson, H.264 and MPEG-4 Video Compression, John Wiley & Sons Ltd., New Jersey, 2003.
[21] L. Rodriguez-Benitez, J. Moreno, J.J. Castro-Schez, J. Albusac, L. Jimenez, An approximate reasoning technique for segmentation on compressed MPEG video, in: 2nd International Conference on Computer Vision Theory and Applications, VISAPP'07, Barcelona, Spain, 2007, pp. 184–191.
[22] L. Rodriguez-Benitez, J. Moreno, J.J. Castro-Schez, J. Albusac, L. Jimenez, Automatic objects behaviour recognition from compressed video domain, Image and Vision Computing 27 (6) (2009) 648–657.
[23] O. Sajhputera, P. Matsakis, J.M. Keller, R. Bondougula, Linguistic descriptions for an object in motion, in: Proceedings of the NAFIPS, New Orleans, Louisiana, USA, 2002, pp. 243–248.
[24] D. Schonfeld, D. Lelescu, VORTEX: video retrieval and tracking from compressed multimedia databases-multiple object tracking from MPEG-2 bitstream, Journal of Visual Communication and Image Representation (11) (2000) 154–182.
[25] R.E.O. Schultz, T.M. Centeno, G. Selleron, M.R. Delgado, A soft computing-based approach to spatio-temporal prediction, International Journal of Approximate Reasoning 50 (1) (2009) 3–20.
[26] R. Soodamani, Z.Q. Liu, GA-based learning for a model-based object recognition system, International Journal of Approximate Reasoning 23 (2) (2000) 85–109.
[27] O. Sukmarg, K.R. Rao, Fast object detection and segmentation in MPEG compressed domain, in: Proceedings of the TENCON, vol. 3, Kuala Lumpur, Malaysia, 2000, pp. 364–368.
[28] M. Valera, S.A. Velastin, Intelligent distributed surveillance systems: a review, IEEE Proceedings on Vision, Image and Signal Processing 152 (2005) 192–204.
[29] L.A. Zadeh, The concept of linguistic variable and its application to approximate reasoning, Information Sciences 8 (1975) 199–249.
[30] W. Zeng, J. Du, W. Gao, Q. Huang, Robust moving object segmentation on H.264/AVC compressed video using the block-based MRF model, Real-Time Imaging (11) (2005) 290–299.
[31] H. Zhang, J.E. Fritts, S.A. Goldman, Image segmentation evaluation: a survey of unsupervised methods, Computer Vision and Image Understanding (110) (2008) 260–280.