Applied Soft Computing 10 (2010) 318–331

Contents lists available at ScienceDirect

Applied Soft Computing

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

Video sequence motion tracking by fuzzification techniques

Juan Moreno-Garcia [a], Luis Rodriguez-Benitez [b], Antonio Fernández-Caballero [c,*], María T. López [d]

[a] Escuela Universitaria de Ingeniería Técnica Industrial de Toledo, Universidad de Castilla-La Mancha, 45071 Toledo, Spain
[b] Escuela Universitaria Politécnica de Almadén, Universidad de Castilla-La Mancha, 13400 Almadién, Spain
[c] Instituto de Investigación en Informática de Albacete (I3A) & Escuela de Ingenieros Industriales de Albacete, Universidad de Castilla-La Mancha, Campus Universitario s/n, 02071 Albacete, Spain
[d] Instituto de Investigación en Informática de Albacete (I3A) & Escuela Superior de Ingeniería Informática, Universidad de Castilla-La Mancha, 02071 Albacete, Spain

ARTICLE INFO

Article history:
Received 26 April 2008
Received in revised form 6 April 2009
Accepted 2 August 2009
Available online 8 August 2009

Keywords:
Fuzzy sets
Permanency values
Motion analysis
Segmentation
Tracking

ABSTRACT

In this paper a method for moving objects segmentation and tracking from the so-called permanency matrix is introduced. Our motion-based algorithms enable to obtain the shapes of moving objects in video sequences starting from those image pixels where a change in their grey levels is detected between two consecutive frames by means of the permanency values. In the segmentation phase matching between objects along the image sequence is performed by using fuzzy bi-dimensional rectangular regions. The tracking phase performs the association between the various fuzzy regions in all the images through time. Finally, the analysis phase describes motion through a long video sequence. Segmentation, tracking an analysis phases are enhanced through the use of fuzzy logic techniques, which enable to work with the uncertainty of the permanency values due to image noise inherent to computer vision.

1. Introduction

Segmentation, tracking and later analysis of motion of objects in video sequences is a topic of growing interest in a great variety of applications [28]. This is why a great effort in research has been made so far in motion-based computer vision applications [18,1,38]. Firstly, a motion-based shape extraction from image sequences is performed (e.g. [23,30,31,5,42], among others). After that, the tracking of these shapes is done. Generally, these approaches are based on the fact that the image flow of a moving object varies spatially as well as temporarily. From most papers (e.g. [27]) one can guess the idea of coherently grouping spatially and temporally image pixels into regions based on a common set of features.

This work is based on fuzzy logic during its three phases. Firstly, we present some works that use fuzzy logic during the image segmentation. A fuzzy clustering algorithm named *Fuzzy Clustering/Segmentation Algorithm* (*FCSA*) is introduced [37]. A neuro-fuzzy image segmentation algorithm complemented by a path detection architecture is presented in [7]. [32] introduces a segmentation method that directly works on MPEG video format without decompression. Moreover, there are also some fuzzy methods for color image segmentation. This is the case of [12] using the T-LAMDA algorithm, or [22] intended to impulse noise filtering for color images. An important application field of fuzzy logic is in medical magnetic resonance image (MRI) segmentation [41,29,2,34]. Very recently, a new method called fuzzy membership C-means (FMCMs) for segmentation of MRI, and an efficient program implementation of it to the segmentation of MRI has been described [19]. According to [43], when noisy image segmentation is required, fuzzy C-means (FCMs) should be modified such that it can be less sensitive to noise in an image. In this correspondence, a robust fuzzy clustering based segmentation method for noisy images is developed by their authors. In [9] a motion estimation method in video sequences which uses fuzzy representation of the grey levels is presented. Motion is characterized by a series of parameters such as horizontal translation, rotation, and so on.

Respect to image tracking, a fuzzy tracking system for rotation-invariant image tracking is presented in [39]. This work is suitable for the implementation of real-time operation. A dual-template strategy is proposed; only two parallel matched filters and simple fuzzy logic are employed to construct the fuzzy tracking system. Another work line is presented in [35]. This paper presents an image tracking system and its applications for traffic monitoring and accident detection at road intersections. This work uses the active contour model approach, Kalman filtering techniques to

* Corresponding author. Tel.: +34 967 599200; fax: +34 967 599224.
E-mail addresses: Juan.Moreno@uclm.es (J. Moreno-Garcia), Luis.Rodriguez@uclm.es (L. Rodriguez-Benitez), caballer@dsi.uclm.es (A. Fernández-Caballero), mlopez@dsi.uclm.es (M.T. López).

1568-4946/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2009.08.002

track individual vehicle motion and a contour initialization method based on the concept of contour growing. Another image tracking work is presented in [15]. A fuzzy system is developed to ponder update decisions both for the trajectories and shapes estimated for targets from the image regions extracted in the images. An automatic procedure, based on neuro-fuzzy techniques, is applied to obtain a set of rules from representative examples. Lastly, the work by Rodriguez-Benitez [33] introduces a tracking algorithm that works by using as input an MPEG compressed video. Their algorithm makes use of fuzzy logic during the tracking phase. They test its method to track vehicles in complex traffic sequences obtaining good results in difficult traffic situations. Also, [40] proposes a new method to detect the moving objects using the orthogonal Gaussian-Hermite moments. For the segmentation of moving objects in the moment images, the authors propose a fuzzy relaxation method and 3D morphological relaxation method. Finally, the trajectories, speeds and accelerations of moving objects are analyzed to determine their moving direction.

Firstly, this paper introduces our proper method for spatially temporally segmenting and tracking rigid and/or non-rigid objects taking advantage of the inherent motion present in image sequences [13]. As a novelty, this paper also introduces the first results of applying fuzzy logic to enhance the analysis of the moving objects in video sequences. Binary logic does not permit to express all restrictions that enable to compose an object from others in an efficient manner [3]. The logic decision rules enable a reduction of the complexity and a grouping of the spatial objects into coherent classes [11]. Therefore, fuzzy logic will provide the management of uncertain data, necessary to correct the possible errors that appear during the segmentation and tracking phases. On the other hand, fuzzy logic incorporates the necessary characteristics for the definition of a linguistic description of the movement of the objects [44–46].

The method proposed consists of three phases that coincide with the traditional steps used in computer vision: segmentation, tracking and analysis, which are described in the following sections. The following section explains the segmentation phase of the proposed method. Section 3 introduces the tracking phase, and the analysis phase is shown in Section 4. The paper ends by applying the method to two video scenes and then some discussion and conclusions are offered.

2. Segmentation phase

The first phase is the image sequence *Segmentation phase*. Here, the motion vectors of an image [24,25] are used for a later processing of these vectors to obtain the positions of the moving objects present in the images of the scene. The position of each object is represented by means of a fuzzy region that tells *where the object is located*. Thus, for each input image a list of n components is obtained, where n is the number of moving objects detected within the image. Additionally, the rectangular area of each object of the scene is calculated (represented by means of a fuzzy region, as it was mentioned earlier). The following subsections describe the processes belonging to the segmentation phase.

2.1. Obtaining the permanency matrix

The first step of the *Segmentation phase* aims to obtain a matrix of permanency values, which will be called P, for each image of a sequence by applying the method described in [13], and that is reproduced in part by means of Algorithm 1. Notice that N and M are the number of rows and columns, respectively, of the input image. Also consider that the permanency matrix P starts with all its values charged at the maximum allowed value of 255. The minimum value for a cell of the permanency matrix is 0.

Algorithm 1. Obtaining the permanency matrix

```
for i = 1 to N do
  for j = 1 to M do
    if I[i, j] = 0 then
      P[i, j] ← max(255, P[i, j] + SUM)
    else
      P[i, j] ← 0
    end if
  end for
end for
```

The algorithm has to be understood as explained next. If no motion is detected at a given pixel $I[i, j]$ between two consecutive input images, that is to say, when $I[i, j] = 0$, the permanency matrix increments its value at $[i, j]$ by a constant value *SUM* up to the maximum of 255; while, if motion is detected, the permanency value is set to 0. In the first case, if no motion persists through time, the permanency matrix goes gradually incrementing its charge. Now, when motion is detected, there is an instant discharge [14]. Fig. 1(b) shows the permanency matrix P resulting from the detection of motion of the moving ball (Fig. 1(a)) in form of the trail as traced.

2.2. Fuzzification of the permanency matrix

In this second step the permanency matrix P of each image is fuzzified. For this purpose, previously the fuzzy sets that indicate *how much time has elapsed since there was motion on that pixel* are

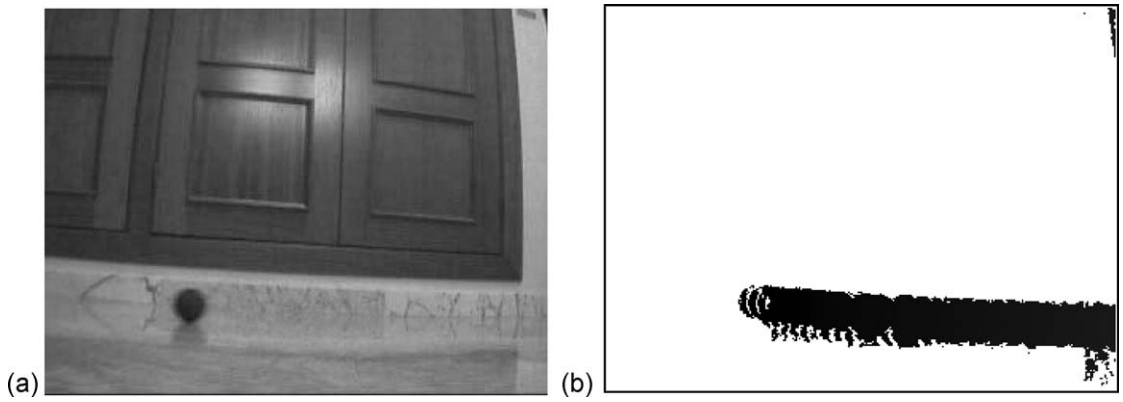

Fig. 1. Example of the permanency matrix. (a) Image 60 of a video sequence and (b) permanency matrix showing the motion of the ball.

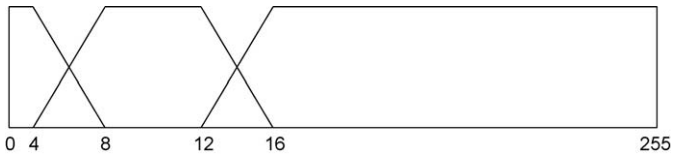

Fig. 2. Initial fuzzy sets.

previously defined. Fuzzy sets are used in this work due to the existence of a large uncertainty inherent to the permanency values approach, and also as input images carry a lot of noise. In our approach images are captured by low-resolution surveillance cameras, and, in general, with a relatively low frame rate of 25 images per second. Fig. 2 shows an example of the kind of fuzzy sets used.

As you may note, the support is defined from 0 to 255, which are precisely the values that the permanency matrix cells may take. The values usually concentrate close to 0, as this is the zone of the support where the more recent motion is found. The support of these fuzzy sets is wider as the motion to be detected is quicker. The result of this process is gotten in the fuzzified matrix F (one per image).

Algorithm 2. Permanency matrix fuzzification

for $i = 1$ to $NRows$ **do**
  for $j = 1$ to $NColumns$ **do**
    $F[i,j] \leftarrow argmax_{A_k}\ \mu_{A_k}(P[i,j])$
  end for
end for

Algorithm 2 describes the process where the permanency matrix is completely covered and the fuzzified matrix F is calculated. Each cell of the fuzzified matrix is assigned the fuzzy set A_i, which obtains the maximum membership grade to the value found in the same cell of the permanency matrix P, if there are two labels with the same membership grade it is selected the smaller one. This matrix will be used in Section 2.4 to group and to obtain the fuzzy regions that represent the position of the object.

2.3. Calculation of the fuzzy sets for axes X and Y

The aim of this third step of the *Segmentation phase* is to calculate the image zones where there is motion, starting from the values of the fuzzified matrix F. Bi-dimensional rectangular fuzzy regions that represent the zone where a moving object is present are obtained for each image.

Matrix F contains a fuzzy set at each cell. A value close to 0 means that there was motion fewer moments ago. On the other hand, we will also consider that motion on a pixel is not only represented by this value, but that it is also influenced by *the motion of its neighbors*. For it, in order to calculate motion in a pixel taking into account its neighboring pixels, an N_x by N_y window is passed over axes X and Y, respectively. Two matrixes are obtained this way. The first one is called W_x and it contains the fuzzy sets obtained when passing the N_x-size window by rows; the other one contains the fuzzy sets obtained when passing the N_y-size window by columns (W_y). That is to say, for each position y of each row x the fuzzy sets which represent the mean of the fuzzy sets from $F[x, y - N_x]$ to $F[x, y + N_x]$ are calculated. And, for each position x of each column y the fuzzy set which represents the mean of the fuzzy sets from $F[x - N_y, y]$ to $F[x + N_y, y]$ are calculated (see Fig. 3). Eq. (1) represents the expression used to perform the calculation of the value corresponding to cell $W_x[x, y]$. In other words, row x is completely covered and the fuzzy set for position y of that row is calculated. For the calculation of the value of each pixel in column y at position x the very similar Eq. (2) is used.

$$W_x[x,y] = \frac{\sum_{i=y-N_x}^{y+N_x} F[x,i]}{(2{*}N_x)+1} \tag{1}$$

$$W_y[x,y] = \frac{\sum_{i=x-N_y}^{x+N_y} F[i,y]}{(2{*}N_y)+1} \tag{2}$$

As you may notice, the sum of the trapezoidal fuzzy sets that belong to the previously defined window is performed, and the result is divided by the number of elements that form that window, namely $(2{*}N_x)+1$. Let $A = [a_0, a_1, a_2, a_3]$ and $B = [b_0, b_1, b_2, b_3]$ be two trapezoidal fuzzy sets; then, their sum $C = [c_0, c_1, c_2, c_3]$ will be another trapezoidal fuzzy set where $c_0 = a_0 + b_0$, $c_1 = a_1 + b_1$, $c_2 = a_2 + b_2$ and $c_3 = a_3 + b_3$. This process is very similar to fuzzy numbers operations using the extension principle [36]. The division of a fuzzy set C by a crisp number n is performed by means of Eq. (3) which obtains as result the fuzzy set R.

$$R = \frac{C}{n} = \frac{[c_1, c_2, c_3, c_4]}{n} = \left[\frac{c_0}{n}, \frac{c_1}{n}, \frac{c_2}{n}, \frac{c_3}{n}\right] \tag{3}$$

Next, it is possible to detect motion as a calculus on a rows and columns basis. For it, a representative crisp value for motion at each row starting from matrix W_y and at each column starting from W_x is searched. This crisp value is named *motion indicator*. That is to say, we will synthesize the fuzzy sets of the columns of matrix W_x in a vector of crisp values of size M called V_x. The same applies to matrix W_y, obtaining the vector of size N of crisp values termed V_y. In order to calculate the value for each one of the vectors V_x and V_y the following considerations are pertinent:

(1) The closer a fuzzy set of W_x or W_y to 0, the more attention it should be paid to detect motion at this row. This objective may be performed by means of a defuzzification process of the fuzzy set [36], or by means of *ranking fuzzy sets* [6,10]. In this work we have preferred to use the Mean Of the Maximum (*MOM*) defuzzification method [21].
(2) A fuzzy set close to 0 is more relevant when its predecessors also hold a value close to 0. This is implemented by means of a weigh factor such that, if the predecessor fuzzy sets of a cell hold a value close to 0, the weigh factor takes a greater value.
(3) The fuzzy sets far away from 0 are of no importance.

Algorithm 3. Calculating the motion indicator for row K

weigh=0
inc=γ
output=0
for $y = 1$ to N **do**
  if $MOM(W_x[K,y]) < \delta$ **then**
    $output \leftarrow output + \left(\frac{weigh}{MOM(W_x[K,y])}\right)$
    weigh=weigh+inc
  else
    if $weigh > 0$ **then**
      $weigh \leftarrow weigh - inc$
    end if
  end if
end for

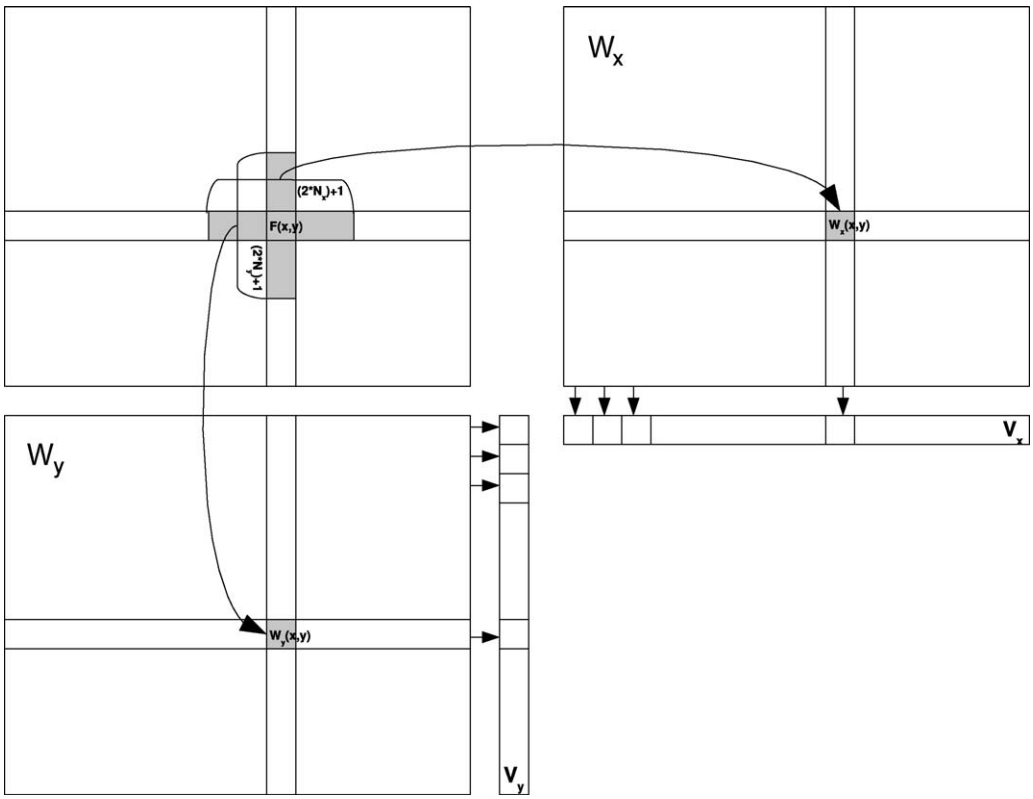

Fig. 3. Method to obtain W_x and W_y.

Algorithm 3 shows the process to calculate the motion indicator of a column K of matrix W_x. Variable *weigh* is used to calculate the weigh value used for each cell. The variable is initialized to 0, and it goes incrementing or decrementing depending on the fact that the *MOM* of the fuzzy set of a given cell ($W_x[K,y]$) is greater than or less than δ, which represents the desired closeness to 0. Variable *inc*, representing the quantity that increments or decrements variable *weigh*, takes the constant value γ. Variable *output* is the output crisp value, and each time that $MOM(W_x[K,y])$ is less than δ it is incremented in $(weigh/MOM(W_x[K,y]))$. This expression is in function of *weigh* and of the defuzzification value $W_x[K,y]$, considering that:

- The greater the value of *weigh*, the higher value it provides to *output*.
- The lower value $MOM(W_x[K,y])$ has, the higher value it provides to *output*. $MOM(W_x[K,y])$ is always less or equal to 1, as the fuzzy sets are normalized. Furthermore, the more motion a fuzzy set represents, the closer the *MOM* will be to 0.

The way of using matrix W_y to calculate the motion indicator in one of its rows is shown in Algorithm 3. Fig. 3 represents how to obtain vectors V_x and V_y. As you may grasp, in order to calculate the value, each column or row of W_x and W_y, respectively, is covered, by applying Algorithm 3. Fig. 4 shows a graphic of a real example of vectors V_x and V_y. We come to the following conclusions:

(1) In the zones of the domain of V_x or V_y where there is a moving object, the values of the motion indicators of the cells are very high in comparison with the ones where there are no moving objects.
(2) The maximum values of the motion indicators from one object to another one may vary considerably (see Fig. 4(a)). For instance, consider how from index 342 up to index 399 there is one object whose maximum is much lower than the maximum of the object defined from indexes 440 to 493. This really turns out to be a problem when calculating the fuzzy sets that represent the zone containing moving elements. Indeed, when normalizing values, some objects whose maximum values are low *are too low*.
(3) There are low output values with no continuity along a studied axis. This is due to the fact that in that zone there is no real motion. This occurs due to the noise present in the permanency values (see Fig. 4(a)). Hence, these values must be ignored.

Vectors V_x and V_y of each image enable to calculate the fuzzy sets that define the zones where there is motion in axes X and Y. Fig. 4 aims to represent the way we are performing. The idea consists in obtaining the fuzzy sets in the zones where there is continuity of values greater than 0. In order to simplify the process, again we shall use trapezoidal fuzzy sets. To obtain these sets, previously the following actions oriented to adequate the graphic to provide solutions to the mentioned problems are performed:

(1) There will be a cut in the graphic for all values superior to α. That is to say, values greater than α are reassigned the value α (Fig. 4(a)). Due to the layout of the graphic this cut has to take a great value. In the tests performed in this work the cut has been established to a 30%.
(2) Next, the graphic is normalized to values between 0 and 1 (Fig. 4(b)). As the superior cut has already been previously performed, there is less difference between the motion indicators of the different moving objects.
(3) In order to eliminate low values and those which have no continuity, an inferior cut is performed to value β. That is to say, values lower than β are made equal to 0.

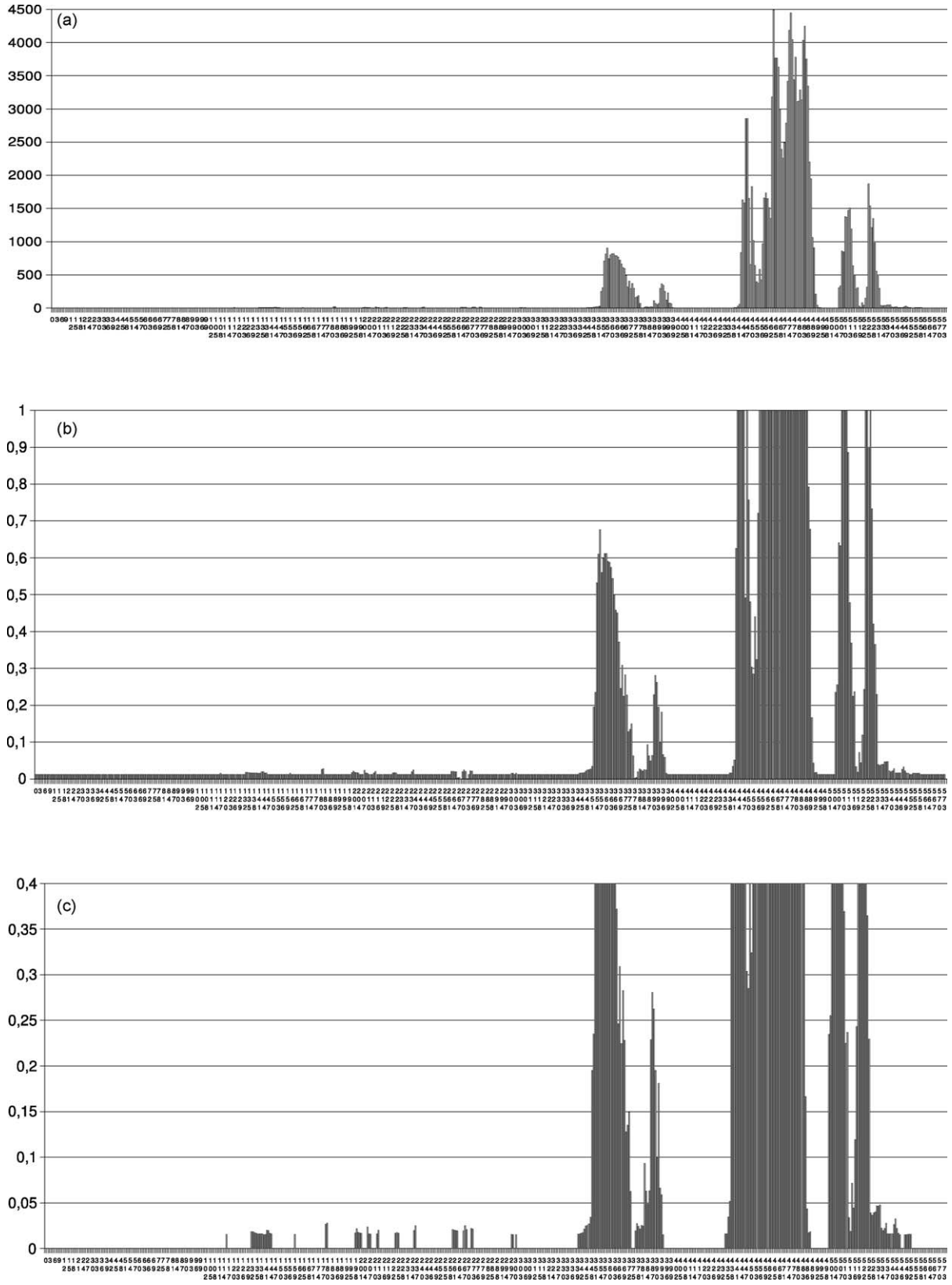

Fig. 4. Vector graphics. (a) Raw vectors output, (b) normalized vectors output and (c) final vectors output.

(4) Finally, a new superior cut to value λ is performed (Fig. 4(c)). This cut is used for the calculation of the trapezoidal fuzzy sets.

After applying the previous process on V_x and V_y, two vectors are obtained, CV_x and CV_y, which are cut and normalized according to the previous description. These new vectors are named *cut vectors* (for example, Fig. 4(c)). Algorithm 4 obtains the trapezoidal fuzzy sets present in vectors CV_x and CV_y. This algorithm takes as input a *cut vector* and gets as output the set of trapezoidal fuzzy sets *FS*. The algorithm is run twice, one with CV_x and another one

with CV_y, obtaining outputs FS_x and FS_y. The algorithm firstly groups consecutive intervals in CV that are different from 0 and show some more restrictions (Algorithm 4), obtaining the set of intervals I. Finally, it calculates the output set of fuzzy sets FS, which get a trapezoidal fuzzy set FS^j for each interval I^j of I assigning its four components as indicated by Algorithm 4.

Algorithm 4. Obtaining the trapezoidal fuzzy sets from a cut vector CV

4.1) Obtain intervals in CV that verify that their values are different from 0. These intervals may contain subintervals of zeroes of a maximum length $MaxLength$. Each obtained interval must contain at least a minimum length of $MinLength$ and must not contain a percentage of zeroes greater than the parameter $MaxZeroes$. A set of intervals $I = [I^0, I^1 \dots I^t]$ is obtained, where each interval j is represented as $I^j = [i_j^{FirstPixel}, i_j^{LastPixel}]$. For instance, in a domain ranging from 0 to 255, the set of intervals $I = [[20, 55], [102, 189]]$ indicates that there are two zones with motion in that axis, one that starts in pixel 20 and ends in pixel 55, and another one running from pixel 102 to pixel 189.

4.2) Starting from I a set of trapezoidal fuzzy sets called $FS = [FS^0, FS^1 \dots FS^t]$ is obtained, where each fuzzy set $FS^j = [fs_j^0, fs_j^1, fs_j^2, fs_j^3]$ is calculated from I^j of I. The four values that define each set FS_j are assigned to:

(1) $fs_j^0 = i_j^{FirstPixel}$.
(2) fs_j^1 takes the index where the maximum value of interval I^j is present, beginning the search from its first position and forward.
(3) fs_j^2 takes the index where the maximum value of interval I^j is present, beginning the search from its last position and backward.
(4) $fs_j^3 = i_j^{LastPixel}$.

2.4. Obtaining the fuzzy regions of each image

In the previous step FS_x and FS_y, which contain the trapezoidal fuzzy sets for axes X and Y, respectively, have been gotten. A fuzzy region where *there may be motion* is a region obtained by the fuzzy sets obtained for axis X with the fuzzy sets obtained for axis Y, that is, the obtained region is the intersection zone of the projection of the fuzzy sets of X-axis and Y-axis (FS_x and FS_y respectively). For example, Fig. 5 shows an image of a scene with two moving objects. Both objects cause that two projections appear in axes X ($Proy_x^0$ and $Proy_x^1$) and Y ($Proy_y^0$ and $Proy_y^1$). Then, in vectors FS_x and FS_y two fuzzy sets appear in each of them, namely $FS_x = \{FS_x^0, FS_x^1\}$ and $FS_y = \{FS_y^0, FS_y^1\}$. The domain of the projections $Proy_x^0$, $Proy_x^1$, $Proy_y^0$ and $Proy_y^1$ must be similar to the support of FS_x^0, FS_x^1, FS_y^0 and FS_y^1. This indicates that fuzzy regions that appear as crossing zones of the obtained fuzzy sets (A, B, C and D in the figure) represent the zones where there might be an object. As you may notice, not all of them are moving objects. For instance, in Fig. 5 consider that: (1) Only one of regions A or B represents an object. (2) Only one of regions C or D represents an object. (3) Only one of regions A or C represents an object. (4)Only one of regions B or D represents an object.

In other words, it is easy to interpret that moving objects are in fuzzy regions A and D, or in B and C. In order to calculate the fuzzy regions that really represent motion, all of them are estimated by means of an evaluation function that uses FS_x and FS_y, as well as the fuzzified matrix F. Each fuzzy region $FR_{a,b}$, obtained from fuzzy sets FS_x^a and FS_y^b is evaluated through Eq. (4). The membership grade of a fuzzy region $\mu_{FR_{a,b}}(x, y)$ is obtained by means of Eq. (5), where $\otimes$ is a t-norm.

$$\frac{\sum_{i=fs_a^0}^{fs_a^3} \sum_{j=fs_b^0}^{fs_b^3} MOM(F[i, j]) * \mu_{FR_{a,b}}(x, y)}{(fs_a^3 - fs_a^0) * (fs_b^3 - fs_b^0)} \quad (4)$$

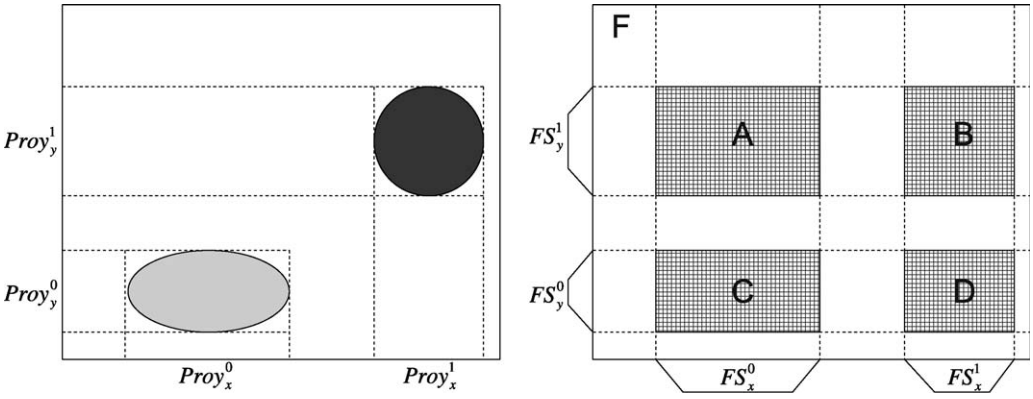

Fig. 5. Calculating the fuzzy regions from FS_x vector, FS_y vector and F matrix.

$$\mu_{FR_{a,b}}(x,y) = \otimes(\mu_{FS_x^a}(x), \mu_{FS_y^b}(y)) \quad (5)$$

As it is easy to notice, Eq. (4) is weighed using the *MOM* of a fuzzy set [21]. This way, the fuzzy sets with support close to 0 hold a weigh factor lower than those that have a support far from 0. For this reason, the fuzzy regions which fulfill the previous restrictions and which obtain a minimum value in the evaluation function of Eq. (4) are selected.

3. Tracking phase

During the *Tracking phase* each object of one image is matched with the objects of the next image. Some issues have been considered:

(1) In this phase we consider that, although the quality of the video may be not quite good (we usually work with 25 frames per second), an object in one image has to "*touch*" the same object in the following image. This is reflected as there must be an intersection between the rectangular area of one object in an image and the rectangular area of the same object in the next images.
(2) Furthermore, in some cases, some objects that have no correspondence are detected during the scene, gotten due to some noise in the values of the permanency matrix. When this is the case, as there is no continuity, the objects are just disregarded.
(3) Another possible effect is that there are objects that are not detected in some images of the video stream. This is not a real problem, as there are enough images per second; and, if an object is not detected in one image, it will be recovered in the following ones.

So, there is a selection of correct areas – those that have a correspondence through time – and an elimination of the object that are "*believed*" to be wrongly detected. Our method obtains a structure *OBJECTS* for each image. The objects of *OBJECTS* are substituted by their area and each object in *OBJECTS* is added sequentially in the sequence *seq*. Here, *seq* is a sequence that contains the objects of each image represented by means of their rectangular area. The idea underlying the process consists in selecting one object of the image of *seq*, and to recover the rest of the images in order to find the object in the images. Thus, a matching of the object is performed through the image sequence. The objects matched in the sequence are erased, and the process is repeated for each one of the objects present in the first image.

The next step is to select an object kept on the second image of *seq* and to perform the same process. There are two possible cases: (1) There are no more objects in the second image, as all the objects of the first image have been matched to the objects of the second one. Now, the process continues with the third image of *seq*. (2) There are still some objects; so, the process is repeated. The same process is repeated for all images of *seq*. As a result, we obtain a list of all the matched objects along the images, which are represented as rectangular areas.

Algorithm 5. Tracking phase

$\Theta \leftarrow \emptyset$
while *seq* not empty **do**
  if $seq[1]$ is empty **then**
    Delete $seq[1]$ from *seq*
  else
    $area \leftarrow ObtainArea(seq[1])$ {Obtains an area from $seq[1]$ that is assigned to *area* and deleted from $seq[1]$}
    $L \leftarrow \emptyset$
    $L \leftarrow L + area$
    for $i = 2$ to $|seq|$ **do**
      while $(|seq| > j)$ and (not $found$) **do**
        if intersection of *area* and $seq[i][j]$ more than *common* **then**
          $L \leftarrow L + seq[i][j]$
          $area \leftarrow seq[i][j]$
          Delete $seq[i][j]$ from *seq*
          $found \leftarrow TRUE$
        else
          $j \leftarrow j + 1$
        end if
      end while
    end for
    $\Theta \leftarrow \Theta + L$
  end if
end while
for $i = 1$ to $|\Theta|$ **do**
  if $duration(\Theta[i]) < duration$ **then**
    $Delete(\Theta[i] \in \Theta)$
  end if
end for

Algorithm 5 shows in deep detail the tracking phase process. The variable *objects* is a list containing the detected objects. Initially it is empty ($objects \leftarrow \theta$). The first *while* is used to cover *seq*, and with the first *if* sentence we check if there is an object in the first image of the sequence ($seq[1]$) of *seq*. If it is empty, the first sequence is eliminated and by means of the previous *while* the process goes on with the next image. If there is any object, it is gotten from the first image and it is matched with the rest of the images of the sequence. *L* is a list that contains the objects detected along time; initially it is empty and the first object is added, that is the one with the variable *area*. Next, *seq* is covered by means of the *for* to look for the equivalent object of the variable *area* along *seq*. The next *while* loop is used to search the equivalent object in the image corresponding to the *for* loop ($seq[i][j]$). This means that within image *i* object *j* is accessed. If the area common to both objects, *area* and $seq[i][j]$, is greater than constant *common*, we assume that there is only one same object. In this case, the object is added to *L* ($L \leftarrow L + seq[i][j]$). Value *area* is assigned to $seq[i][j]$ in order to compare the area of the last detected object during the next iteration ($area \leftarrow seq[i][j]$). Now, $seq[i][j]$ is eliminated from *seq* as this object has just been matched (Eliminate $seq[i][j]$ from *seq*) and the variable that stops the iterations of the *w* hile loop is set to *TRUE* ($found \leftarrow TRUE$). If the previous condition is not fulfilled the search continues in the objects of *seq*.

The last action of this phase consists in eliminating the objects from *objects* that have not been detected during a minimum time interval. This is performed by means of a parameter *duration*, which controls the minimum duration of the object in the video. If it lasts less time, it is eliminated through the last *for* loop.

4. Analysis phase

The aim of *Analysis phase* is to describe the movements of the objects throughout the video sequence. In this paper a first approach for this phase is introduced, as we are still actively working on it. The previous phase obtained a list Θ containing the positions of the moving objects on each image in form of rectangular and bi-dimensional fuzzy regions. In this last phase, trapezoidal fuzzy sets are also used to describe the movement over each dimension (' 'how much the object has moved''), and a *sign* that indicates *"the direction of the movement"*.

Firstly, the subtraction of fuzzy sets of each axis between two consecutive images is used to calculate *"how much the object has moved"*. Fig. 6 shows how this subtraction is performed. FS_i and FS_{i+1} are trapezoidal fuzzy sets of an axis obtained at time instants *i* and $i+1$, respectively. FS_i is represented by continuous and heavy outlines, while discontinuous and heavy outlines are used for FS_{i+1}. There are four possible cases: (a) $FS_{i+1} < FS_i$ (Fig. 6(a)), (b) $FS_i < FS_{i+1}$ (Fig. 6(b)), (c) $FS_i \subset FS_{i+1}$ (Fig. 6(c)), and, (d) $FS_{i+1} \subset FS_i$ (Fig. 6(d)). For the three first cases, the substraction of FS_{i+1} support minus FS_i support is the FS_{i+1} support that does not belong to FS_i (the obtained fuzzy set $D = [d_0, d_1, d_2, d_3]$ is projected on the *X*-axis in Fig. 6), while for the last case, the substraction is the empty set (FS_{i+1} is included in FS_i).

To calculate *"the direction of the movement"* in the *X*-axis, it is considered that the smallest domain values indicate that the object is more to the left, and the greater domain values show that the object is more to the right. The symbol$^{'-'}$ is used to represent a movement from right to left, whereas the symbol$^{'+'}$ means a movement from left to right (Fig. 7). Also, in the *Y*-axis, a movement from top to down is indicated by using the symbol$^{'+'}$, and a movement from down to top is represented by using the symbol$^{'-'}$. The objects found are ignored when there is no displacement, that is to say, when the substraction of two fuzzy sets is empty ($FS_{i+1} \subset FS_i$). In the third case (Fig. 6(c)), the obtained support is divided into two parts; two fuzzy sets are obtained. Then we select the left fuzzy set if

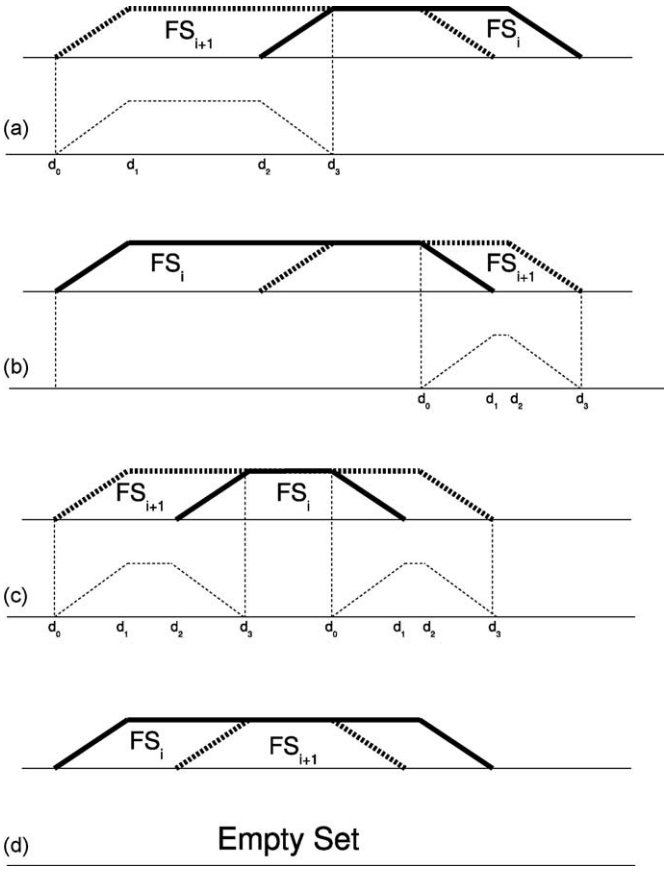

Fig. 6. Substraction between fuzzy sets.

the direction is$^{'-'}$, or the right one if the direction is$^{'+'}$. For each axis *E* (being *E* equal to *X* or *Y*), and for each image *i*, a tuple $S_i^E = < sign, D_i^E >$ that represents how much the object *i* has displaced (by using the fuzzy set D_i^E) and the direction of the displacement by using the *sign* is obtained.

Algorithm 6. Analysis phase

displacements=[ ]
for $i = 0$ to $|\Theta|$ **do**
  disp=[ ]
  for $j = 1$ to $|\Theta_i|$ **do**
    $S_i^X = Movement(\Theta_i^{j-1}(X) \circ \Theta_i^j(Y))$
    $S_i^Y = Movement(\Theta_i^{j-1}(X) \circ \Theta_i^j(Y))$
    disp=disp+[j,S_i^X,S_i^X]
  end for
  displacements=displacements+dist
end for

Algorithm 6 shows the behavior of the *A* nalysis phase. For each object – element of Θ – its displacement is calculated from an image to the following one by using the substraction of fuzzy sets. For it, the algorithm uses function *Movement* that obtains the tuple S_i^E for *X* and *Y* axes, respectively, that is (S_i^X and S_i^Y). The sign is assigned to *in* when $FS_{i+1} \subset FS_i$ (Fig. 6(d)). The sentences $S_i^X = Mo(\Theta_i^{j-1}(X), \Theta_i^j(X))$ and $S_i^Y = Mo(\Theta_i^{j-1}(Y), \Theta_i^j(Y))$ are used to calculate the displacements for both axes, where $\Theta_i^j(dim)$ is the fuzzy region *j* for image *i* and *dim* is the dimension (*X* or *Y*). The result is a list with as many elements as objects there are present in

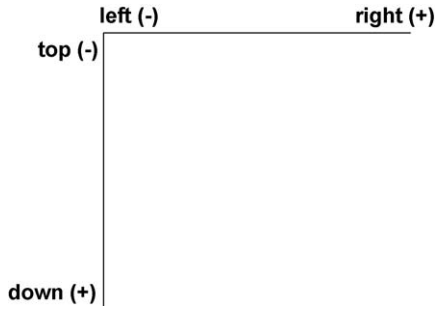

Fig. 7. Direction of the movement.

the scene. The displacement of each object in an image is represented as two tuples S_i^X and S_i^Y.

The results of this phase allow us to study how the objects have moved. The objects' speeds can also be calculated by means of the distance between the fuzzy sets D_i^E and D_{i+1}^E. But this issue is out of the scope of this paper.

5. Data and results

Two examples are used to test the validity of the method proposed, these examples have not occlusions, occlusions will be studied in future works. The first one uses a video sequence that shows a ball rolling from the right the left of the scene (remember Fig. 1(a)). The quality of the video is acceptable; thus, the permanency matrix does not contain much noise (Fig. 1(b)). There is a trail at the top right corner and a small shade at the bottom right corner when the ball is initiating its movement. The parameters used for this example are the following ones: $N_x = 5$, $N_y = 5$, $\alpha = 0.3$, $\beta = 0.05$, $\gamma = 0.15$, $MaxZeroes = 8$ and $MinLenght = 5$.

The *Segmentation phase* performs correctly, as the moving ball is properly detected. In addition, there are no erroneously detected objects in the scene. Fig. 8 shows the fuzzy regions as found in four images of the video sequence for this first example. As you may notice, the ball is located in the first image when it enters into the scene, and it is situated in a central position in the second image (image 49). Next, the ball is approaching to the left of the image, and, finally, it is leaving the scene. The ball is correctly detected throughout the whole video sequence.

Table 1
Ball displacements in the video scene.

Image	Sign	D_i^Y	Sign	D_i^X
36	–	[190, 191, 192, 192]	–	[304, 305, 314, 314]
37	–	[189, 190, 190, 190]	–+	[319, 319, 319, 319]
39		*in*	–	[271, 272, 281, 281]
40	–	[189, 190, 190, 190]	–	[261, 262, 271, 271]
41	–	[188, 189, 189, 189]	–	[252, 254, 261, 261]
42		*in*	–	[242, 243, 252, 252]
43	–	[187, 188, 188, 188]	–	[231, 232, 242, 242]
44		*in*	–	[222, 224, 231, 231]
45	–	[186, 187, 187, 187]	–	[214, 216, 222, 222]
46		*in*	–	[204, 207, 214, 214]
47		*in*	–	[195, 197, 204, 204]
48	–	[185, 186, 186, 186]	–	[180, 182, 195, 195]
50		*in*	–	[166, 167, 181, 181]
51	–	[184, 185, 185, 185]	–	[155, 156, 166, 166]
52	–	[184, 184, 184, 184]	–	[141, 142, 155, 155]
53	–	[182, 183, 184, 184]	–	[140, 141, 141, 141]
54	–	[182, 182, 182, 182]	–	[131, 134, 140, 140]
55	–	[181, 182, 182, 182]	–	[126, 127, 131, 131]

During the *Analysis phase* the results shown in Table 1 are obtained. For the purpose of simplification, this table only shows the outputs obtained for images 36–55. All displacements D_i^Y and D_i^X are correctly distributed throughout their axes. The direction gotten for both axes is –, since the ball rolls from right to left (from $[271, 272, 281, 281]$ to $[126, 127, 131, 131]$), and the ball slightly ascends (from $[271, 272, 281, 281]$ to $[126, 127, 131, 131]$).

The second example is a video scene obtained from the web page "http://www.cvg.cs.rdg.ac.uk/datasets/index.html". This is a

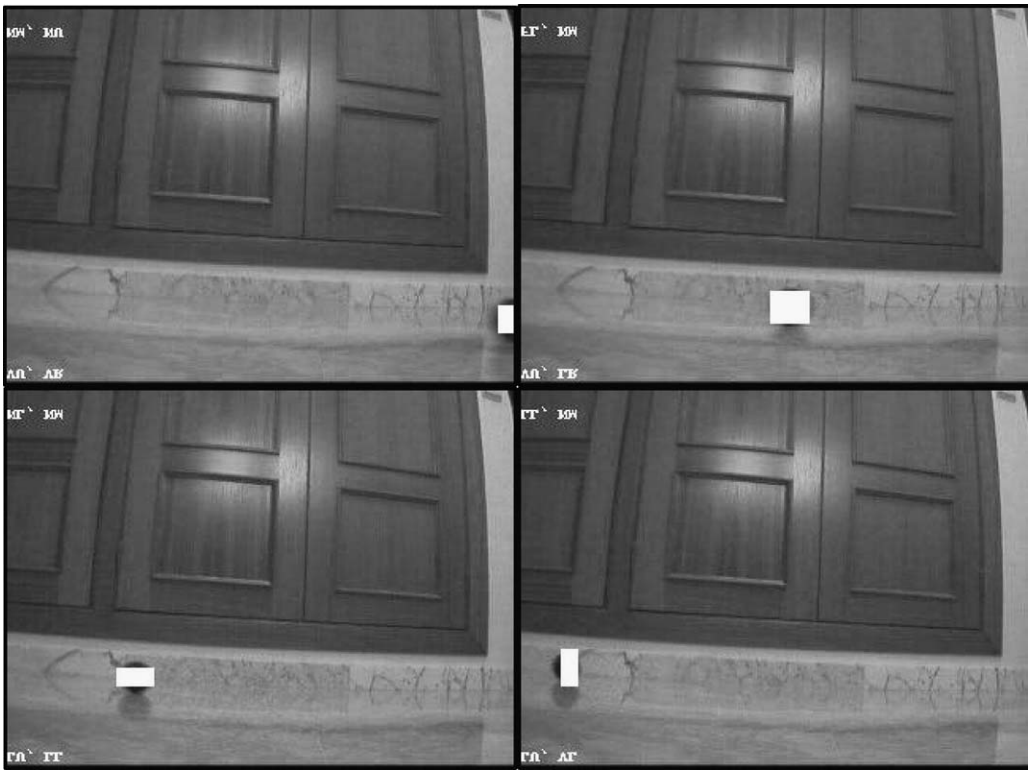

Fig. 8. Fuzzy regions for images 35, 49, 63 and 74 of the first example.

Fig. 9. Input images 61, 73, 85 and 98 of the second example.

surveillance video sequence where a car appears at the right of the image sequence and a pedestrian walks from left to right (Fig. 9). The video quality is rather poor, and this causes that the permanency matrix holds a lot of noise (Fig. 10). More precisely, the contour of non-moving objects (e.g. buildings) is also segmented. The real movement is represented by means of two black trails; the central one is due to the pedestrian's motion, and the right inferior corner trail is caused by the car entering into the

Fig. 10. Permanency matrix for images 61, 73, 85 and 98 of the second example.

Fig. 11. Car and pedestrian fuzzy regions for images 61, 73, 85 and 98 of the second example.

scene. The parameters used in this case are the following ones: $N_x = 5$, $N_y = 5$, $\alpha = 0.3$, $\beta = 0.0015$, $\gamma = 0.40$, *MaxZeroes* = 3 and *MinLenght* = 10.

Again, the performance of the *Segmentation phase* is rather correct, as the two moving objects of the scene (car and pedestrian) are correctly detected in most images. Nevertheless, nonexistent motion is detected in the images as a result of the noise in the permanency matrix. Also, in some images the fuzzy regions do not hold the appropriate size. Nevertheless, these problems are solved in the following phases. Fig. 11 shows the positions of the fuzzy regions obtained in four video images for the car and the pedestrian by using a black box. As you can see, our method locates the position of the car in each image, and the pedestrian is also found. *Tracking phase* matches correctly the objects throughout the images of the video sequence. The fuzzy regions that represent non-real motion are perfectly deleted.

Table 2 (pedestrian) and Table 3 (car) show the output obtained at the *Analysis phase*. Again, we only offer the obtained output for

Table 2
Pedestrian displacements detected.

Image	Sign	D_i^X	Sign	D_i^Y
61	+	[379, 384, 380, 386]	−+	[340, 340, 340, 346]
62		*in*		*in*
64			−+	[340, 340, 340, 346]
65		*in*	+	[398, 400, 398, 402]
66		*in*	+	[398, 402, 402, 403]
67	+	[381, 384, 384, 386]		*in*
68	+	[382, 386, 386, 389]		*in*
69			+	[398, 403, 398, 450]
72		*in*	+	[398, 400, 400, 426]
73			+	[398, 426, 426, 430]
76			+	[396, 402, 402, 420]
77	+	[387, 389, 399, 405]		*in*
78			+	[396, 420, 398, 424]
81		*in*	+	[391, 419, 398, 424]
82	+	[401, 403, 403, 404]		*in*
85	+	[403, 404, 409, 411]	+	[398, 407, 398, 431]
87	+	[409, 411, 413, 415]	+	[398, 431, 398, 433]
90	+	[413, 415, 423, 429]		*in*
91				*in*
93	+	[423, 429, 429, 434]	+	[398, 425, 425, 427]

Table 3
Car displacements detected.

Image	Sign	D_i^X	Sign	D_i^Y
61	−+	[711, 714, 715, 720]	−+	[485, 486, 487, 488]
62	−	[708, 709, 711, 714]	−	[482, 485, 485, 486]
63	−	[705, 708, 708, 709]		*in*
64	−	[702, 705, 705, 708]	−	[473, 482, 482, 484]
65	−	[697, 699, 702, 706]	−	[473, 473, 473, 482]
66	−	[692, 695, 697, 699]	−	[471, 474, 473, 481]
67	−	[690, 692, 692, 695]	+	[512, 519, 558, 575]
69	−	[683, 686, 690, 693]	−	[469, 478, 473, 480]
70	−	[681, 683, 683, 686]	+	[507, 515, 508, 516]
71	−	[678, 681, 681, 684]	+	[508, 516, 516, 525]
72	−	[675, 677, 678, 681]	−	[458, 460, 469, 475]
74	−	[669, 672, 675, 677]	−	[457, 458, 458, 460]
75	−	[667, 669, 669, 672]	−	[455, 458, 457, 458]
76	−	[664, 667, 667, 669]	−	[454, 455, 455, 458]
77	−	[661, 664, 664, 734]	−	[453, 455, 454, 455]
78	−	[659, 661, 661, 709]	−	[451, 454, 453, 455]
79	−	[657, 659, 659, 662]	−	[451, 451, 451, 454]
80	−	[655, 657, 657, 662]		*in*
81	−	[652, 655, 655, 720]	−	[450, 451, 451, 452]
82	−	[650, 652, 652, 714]	−	[449, 450, 450, 452]
83	−	[647, 650, 650, 715]	−	[447, 449, 449, 450]
84	−	[646, 647, 647, 650]		*in*
85	−	[644, 646, 646, 649]	−	[446, 447, 447, 449]
89	−	[635, 639, 644, 707]	−	[442, 445, 446, 447]
90	−	[633, 635, 635, 639]	−	[441, 442, 442, 445]
91	+	[758, 764, 758, 767]		*in*
92		*in*	−	[440, 441, 441, 443]
93	+	[758, 767, 767, 767]	−	[438, 441, 440, 441]
94	−	[638, 685, 758, 767]		*in*
95	−	[638, 684, 758, 767]	−	[344, 355, 438, 441]

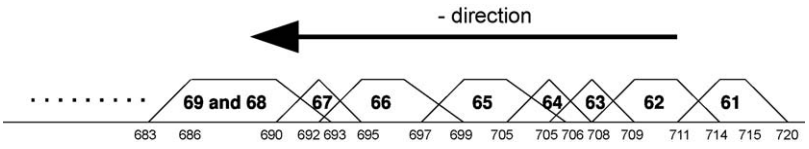

Fig. 12. Car displacements for images 61–69.

images 61–95. In the case of the pedestrian (Table 2), the direction (*sign*) in the *Y*-axis is alternated between *in*, + and −. This is because the pedestrian has no vertical displacement; he is walking parallel to the *X*-axis. The alteration in the values is based on the D_i^X obtained support. With respect to the *X*-axis, the obtained output indicates that the pedestrian is walking from left to right (sign +), and how at each time instant he is approaching the right side (see the support of D_i^Y). In relation to the car (Table 3), along the *X*-axis, the sign takes value −. This occurs as the displacement is from right to left (see the support of fuzzy set D_i^X and Fig. 12), just as the car is entering into the scene from the right. For the *Y*-axis, the obtained output shows how the car is moving upwards.

6. Discussion

A new fuzzy logic method in order to build an artificial vision system has been presented. The proposed technique can be considered full fuzzy logic based since fuzzy logic is used during the complete procedure and this fact is justified considering the existence of a large amount of uncertainty and noise associated to permanency values and test images. Similarly to Zhou and Zhang [47], in this work a fully automatic method is developed. In the initial stage motion vectors are obtained from the input sequence and after that segmentation, tracking and analysis phases are performed.

The motion vectors of an image are used as input to the artificial vision system and this data is considered a 2D approach to motion estimation. In this kind of approaches, motion estimation is obtained from motion of brightness or intensity patterns in the image and there exists a general restriction: the intensity of the image along the motion trajectory must be constant. In other words, changes through time in the intensity of a pixel are only due to motion. Nevertheless, this restriction does not affect our model because the proposed algorithms are prepared to work in those situations.

To establish a comparison with other motion field estimation techniques (the main approach in motion detection) gradient-based techniques are studied. These methods (e.g. [16]) are computationally efficient and satisfactory motion estimation is obtained. But, unfortunately, the common disadvantage of these approaches is the change in illumination. Therefore, the robustness of these methods is usually improved by using regions instead of pixels. In general, these methods are less sensitive to noise than gradient-based methods. Our particular approach takes advantage of this fact and uses all available neighbourhood state information as well as the proper motion information. On the other hand, our method is not affected by the greatest disadvantage of region-based methods. Our model does not depend on the pattern of translation motion. In effect, in region-based methods (e.g. [17]), regions have to remain quite small so that the translation pattern remains valid.

Once the necessary input data is available in the segmentation phase, motion vectors are fuzzified and the fuzzy regions that represent the objects are obtained. If *N* and *M* represent the number of rows and columns of the input image, respectively, the authors consider this stage has an $O(N \times M)$ complexity and that is because the permanency matrix (Section 2.1) is obtained with computational complexity of $O(N \times M)$, the fuzzification takes the same time complexity (Section 2.2) and this is also the case in the calculation process of the fuzzy sets (Section 2.3). Nevertheless, other techniques are less efficient because they require more processing steps. For example, Zhou and Zhang [47] present a method that needs spatial segmentation and motion segmentation. The spatial segmentation is based on probabilistic Fuzzy C-means clustering and Gibbs sampling. The segmented mask obtained is then refined by taking into account motion information. More concretely, the motion vectors are calculated using a block matching method based on phase correlation and also the texture feature is utilized during motion segmentation. The motion features and their spatial relationships are used to associate the segmented regions to form video objects.

With respect to the tracking process defined, it must be said that each object in an image is correctly matched with an object in the next image. Besides, boundaries are well approximated while this is a typical problem in other methods as described in [47]. We directly establish correspondences without any training phase, thus obtaining a faster approach than others like the one presented in [47], where time is consumed in a learning phase to build a membership matrix for each frame, reaching good results if the segmentation is correct.

Bhattacharyya et al. [4] propose a variation of this method for high-speed tracking of objects through the computation of optical flow fields. Optical flow assumes that intensity variations are solely due to image motions and this fact causes limitations of the optical flow field technique [16]. The authors' approach improves the time efficiency of the flow computation procedure by incorporating fuzzy set theoretic concepts. The main problem of this process is the determination of the optimum size of the search window around the point of interest, such that the tracked point does not fall outside the moving object. Besides this method there are other known methods where efficiency depends on the number of objects present in the scene. Our method is able to track several objects without increasing the complexity and without reduction of the quality of results, and, the increase of the complexity only occurs when the number of objects is quite high. On the contrary, there are some methods that are useful for tracking isolated targets in videos. For instance, in a paper by Kima and Park [20], the authors propose a novel object detection algorithm for tracking a moving object in an image sequence.

With respect to the motion analysis, we propose the use of fuzzy logic, allowing the selected representation method to be descriptive and qualitative. In this first approach, the displacement of the objects is obtained and the object's velocity in a qualitative mode is also gotten. Besides, fuzzy logic allows managing a great number of operations that can be incorporated in our analysis to improve it and to obtain more information. In this same direction, Cavallaro [8] introduces a content-based video encoding framework, which is based on semantic analysis, obtaining high-level descriptors, without using fuzzy logic.

7. Conclusions

In this paper a new method for locating moving objects in a video sequence has been introduced. The input data are the permanency values of an image. The process consists of three phases (segmentation, tracking and analysis), all of them based on fuzzy logic.

The *Segmentation phase* is performed by calculating rectangular fuzzy regions that indicate the zones containing motion. The use of fuzzy logic in the segmentation process helps to deal with the noise inherent to images captured as grey levels. The initial fuzzy sets are defined a priori (Fig. 2). This allows us to define them according to the speed of the objects being detected. The greater the speed of the objects, the greater the fuzzy sets support; especially the sets close to zero. In order to detect motion, it is accumulated by rows and columns in two crisp vectors, one vector per dimension. After that, the fuzzy sets for each axis are calculated, then the fuzzy regions that represent the moving objects are gotten.

The *Tracking phase* is based on the fuzzy regions intersection (objects intersection), as we suppose that an object in one image has to "*touch*" the same object in the following image. In this phase the "*nonexistent objects*" detected by our method are disregarded.

Finally, the fuzzy sets named *displacement* for each image axis are calculated in the *Analysis phase*. Each set is associated a sign that indicates the direction of the movement in the axis, and represents the displacement in this image between two consecutive images. This structure allows us to calculate the object's velocity. The use of fuzzy sets makes this phase more descriptive, although it is precisely the phase which we have developed less so far.

The proposed method works correctly in videos with a certain quality, as the absence of noise enables a better detection of the moving objects in the image. In video sequences with much noise, the *S* egmentation phase locates nonexistent objects and the objects size is not always the correct one. This is solved in the tracking and analysis phases. The method introduced allows to describe automatically the motion of the objects in the video sequence starting from the permanency values [26].

As future work, our intention is to improve the *Analysis phase* in order to make it much more descriptive. The *Tracking phase* will be modified to do a local search from the fuzzy regions obtained in the previous images, and we will create a method to fit the fuzzy regions. Another research line is to develop a method for automatically setting up the parameters of the algorithm.

Acknowledgements

This work is supported in part by the Spanish Ministerio de Ciencia e Innovación TIN2007-67586-C02-02 and TIN2009-14538-C02-02 grants, and the Junta de Comunidades de Castilla-La Mancha PII2I09-0052-3440, PII2I09-0069-0994, PII2I09-0071-3947 and PEII09-0054-9581 grants.

References

[1] J.K. Aggarwal, N. Nandhakumar, On the computation of motion from sequences of images—a review, Proceedings of the IEEE 76 (8) (1988) 917–935.
[2] M.N. Ahmed, S.M. Yamany, N. Mohamed, A.A. Farag, T. Moriarty, A modified fuzzy C-means algorithm for bias field estimation and segmentation of MRI data, IEEE Transactions on Medical Imaging 21 (3) (2002) 193–199.
[3] H. Bandemer, S. Gottwald, Fuzzy Logic, Fuzzy Methods with Applications, Wiley & Sons, 1996.
[4] S. Bhattacharyya, U. Maulik, P. Dutta, High-speed target tracking by fuzzy hostility-induced segmentation of optical flow field, Applied Soft Computing 9 (1) (2009) 126–134.
[5] A.F. Bobick, J.W. Davis, An appearance-based representation of action, in: Proceedings of the 13th International Conference on Pattern Recognition, Vienna, Austria, (1996), pp. 307–312.
[6] G. Bortolan, R. Degani, A review of some methods for ranking fuzzy numbers, Fuzzy Sets and Systems 15 (1985) 1–19.
[7] V. Boskovitz, H. Guterman, An adaptive neuro-fuzzy system for automatic image segmentation and edge detection, IEEE Trasactions on Fuzzy Systems 10 (2) (2002) 247–262.
[8] A. Cavallaro, O. Steiger, T. Ebrahimi, Semantic video analysis for adaptive content delivery and automatic description, IEEE Transactions on Circuits and Systems for Video Technology 15 (19) (2005) 1200–1209.
[9] F. Comby, O. Strauss, Using quasi-continuous histograms for fuzzy main motion estimation in video sequence, Fuzzy Sets and Systems 158 (5) (2007) 475–495.
[10] M. Delgado, J.L. Verdegay, M.A. Vila, A procedure for ranking fuzzy numbers, Fuzzy Sets and Systems 26 (1988) 49–62.
[11] R. de Kok, T. Schneider, U. Ammer, Object-based classification and applications in the Alpine forest environment, International Archives of Photogrammetry and Remote Sensing 32 (part 7-4-3 W6) (1999).
[12] A. Doncescu, J. Aguilar-Martin, J.-C. Atine, Image color segmentation using the fuzzy tree algorithm T-LAMDA, Fuzzy Sets and Systems (2006).
[13] M.A. Fernández, A. Fernández-Caballero, M.T. López, J. Mira, Length-speed ratio (LSR) as a characteristic for moving elements real-time classification, Real-Time Imaging 9 (2003) 49–59.
[14] A. Fernández-Caballero, M.A. Fernández, J. Mira, A.E. Delgado, Spatio-temporal shape building from image sequences using lateral interaction in accumulative computation, Pattern Recognition 36 (5) (2003) 1131–1142.
[15] J. García, J. Molina, J. Besada, J. Portillo, A multitarget tracking video system based on fuzzy and neuro-fuzzy techniques, Journal on Applied Signal Processing 2005 (14) (2005) 2341–2358.
[16] B.K.P. Horn, B.G. Schunck, Determining optical flow, Artificial Intelligence 17 (1981) 185–204.
[17] R.M. Horowitz, T. Pavlidis, Picture segmentation by a tree traversal algorithm, Journal of the ACM 23 (1976) 368–388.
[18] T.S. Huang, Image Sequence Analysis, Springer-Verlag, 1983.
[19] S.R. Kannan, A new segmentation system for brain MR images based on fuzzy techniques, Applied Soft Computing (2008).
[20] B. Kima, D. Park, Novel target segmentation and tracking based on fuzzy membership distribution for vision-based target tracking system, Image and Vision Computing 24 (12) (2006) 1319–1331.
[21] W.V. Leekwijck, E.E. Kerre, Defuzzification: criteria and classification, Fuzzy Sets and Systems 108 (2) (1999) 159–178.
[22] Y. Li, F.-L. Chung, S. Wang, A robust neuro-fuzzy network approach to impulse noise filtering for color images, Applied Soft Computing 8 (2) (2008) 872–884.
[23] J.J. Little, J.E. Boyd, Recognizing people by their gait: The shape of motion, Videre: Journal of Computer Vision Research 1 (2) (1998) 2–32.
[24] M.T. López, A. Fernández-Caballero, M.A. Fernández, J. Mira, A.E. Delgado, Motion features to enhance scene segmentation in active visual attention, Pattern Recognition Letters 27 (5) (2006) 429–438.
[25] J. Mira, A.E. Delgado, A. Fernández-Caballero, M.A. Fernández, Knowledge modelling for the motion detection task: The algorithmic lateral inhibition method, Expert Systems with Applications 27 (2) (2004) 169–185.
[26] J. Moreno-Garcia, J.J. Castro-Schez, L. Jimenez, A fuzzy inductive algorithm for modeling dynamical systems in a comprehensible way, IEEE Transactions on Fuzzy Systems, to be published in 2007.
[27] T. Olson, F. Brill, Moving object detection and event recognition algorithms for smart cameras, in: Proceedings of the DARPA Image Understanding Workshop, New Orleans, Louisiana, (1997), pp. 159–175.
[28] N. Pérez de la Blanca, J.M. Fuertes, M. Lucena, Deformable object matching based on multi-scale local histograms, Lecture Notes in Computer Science 3179 (2004) 154–162.
[29] D.L. Pham, J.L. Prince, Adaptive fuzzy segmentation of magnetic resonance images, IEEE Transactions on Medical Imaging 18 (9) (1999) 737–752.
[30] R. Polana, R. Nelson, Detecting activities, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, New York City, NY, (1993), pp. 2–7.
[31] R. Polana, R. Nelson, Recognition of nonrigid motion, in: Proceedings DARPA Image Understanding Workshop, Monterey, California, (1994), pp. 1219–1224.
[32] L. Rodriguez-Benitez, J. Moreno-Garcia, J.J. Castro-Schez, L. Jimenez, S. Garcia, Linguistic motion description for an object on Mpeg compressed domain, in: Proceedings of the Eleventh International Fuzzy Systems Association World Congress II, Beijing, China, II, (2005), pp. 1064–1070.
[33] L. Rodriguez-Benitez, J. Moreno-Garcia, J.J. Castro-Schez, L. Jimenez, Fuzzy logic to track objects from MPEG video sequences, in: 12th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU'08, (2008), 675–681.
[34] S. Shen, W. Sandham, M. Granat, M. Sterr, MRI fuzzy segmentation of brain tissue using neighborhood attraction with neural-network optimization, IEEE Transactions on Information Technology in Biomedicine 9 (3) (2005) 459–467.
[35] J. Tai, S. Tseng, C. Lin, K. Song, Real-time image tracking for automatic traffic monitoring and enforcement applications, Image and Vision Computing 22 (6) (2004) 485–501.
[36] K. Tanaka, An Introduction to Fuzzy Logic for Practical Applications, Springer-Verlag, 1998.
[37] Y.A. Tolias, S.M. Pannas, Image segmentation by a fuzzy clustering algorithm using adaptive spatially constrained membership function, IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 28 (3) (1998) 359–369.
[38] J. Wang, T.S. Huang, N. Ahuja, Motion and Structure from Image Sequences, Springer-Verlag, 1993.
[39] W. Wang, W. Lie, Y Chen, A fuzzy-computing method for rotation-invariant image tracking, Proceedings of Image Processing, ICIP'94, (1994), vol 1, pp. 540–544.
[40] Y. Wu, J. Shen, M. Dai, Traffic object detections and its action analysis, Pattern Recognition Letters 26 (13) (2005) 1963–1984.
[41] C. Xu, D.L. Pham, M.E. Rettmann, D.N. Yu, J.L. Prince, Reconstruction of the human cerebral cortex from magnetic resonance images, IEEE Transactions on Medical Imaging 18 (6) (1999) 467–480.

[42] M.-H. Yang, N. Ahuja, Extracting gestural motion trajectories, in: Proceedings of the 3rd. International Conference on Automatic Face and Gesture Recognition, Nara, Japan, (1998), pp. 10–15.
[43] Z. Yang, F.-L. Chung, W. Shitong, Robust fuzzy clustering based image segmentation, Applied Soft Computing (2008).
[44] L.A. Zadeh, The concept of a linguistic variable and its applications to approximate reasoning. I, Information Science 8 (1975) 199–249.
[45] L.A. Zadeh, The concept of a linguistic variable and its applications to approximate reasoning. II, Information Science 8 (1975) 301–357.
[46] L.A. Zadeh, The concept of a linguistic variable and its applications to approximate reasoning. III, Information Science 9 (1975) 43–80.
[47] J. Zhou, X.P. Zhang, Video object segmentation and tracking using probabilistic fuzzy c-means, IEEE Workshop on Machine Learning for Signal Processing (2005) 201–206.