Abstract

In this paper we present a people tracking algorithm which is able to detect and track soccer players in complex situations with varying light conditions, high frame rate, and real time processing. Object segmentation is performed by means of an algorithm based on background subtraction. In order to cope with presence of moving objects and light changes during the background modeling phase, an approach based on the evaluation of pixels energy content has been developed. Detected objects are then classified by means of an unsupervised clustering algorithm that allows the solution of blobs splitting and merging problems. For people tracking purpose we propose a stochastic approach based on the evaluation of the maximum a posteriori probability (MAP). First of all the algorithm evaluates geometrical information on the blob overlapping and then applies a color feature classification to track players and solve blob merging situations. Experimental tests have been carried out on long soccer image sequences in different weather and light conditions.

1. Introduction

In the last decade the scientific community has been greatly involved in developing visual systems for surveillance applications such as behavior analysis, activity recognition, event detection. These systems generally consist of several steps: recognition of interesting objects in the image, tracking of these objects in the scene along time, analysis of temporal variations for high level processing. A fundamental module of visual systems for soccer analysis is the players tracking algorithm that has to be robust in order to allow any further processing. The ability of tracking of multiple people before, during and after occlusions is vital for many applications and this subject has been, extensively, studied in the literature. McKenna [8] has used colour information alone to track object in complex scenes. Haritaoglu [4] used a silhouette to match objects. Isolated objects or a small number of objects having transient occlusions have been reliably tracked [14]. Whilst these authors claim to be able to track multiple objects, none of them seem to claim that they are able to deal correctly with occlusion. Few works, however, present some ideas to face up occlusions that is reidentifying correctly people when a group of them separates. However, tracking in more crowded situations, where a large number of people with persistent occlusions is present, remains a challenging topic in the computer vision community [17], [7], [2], [3].

A correct people segmentation step is essential to track players during the soccer matches. The most used algorithms for moving objects detection are based on background subtraction: the foreground objects are extracted by subtracting the current image from a reference background model. Good reviews on these approaches can be found in [5] and [13]. The main characteristics of sport applications brought the researchers to use motion detection approaches based on the segmentation of an almost homogeneous region (the field) from the different-colored ones (the players and the ball). In this direction, relevant works has been proposed in [11]: the authors iteratively segment the whole image into smaller regions by thresholding color histograms. Other interesting approaches are based on mathematical morphology: in particular in [10] an interesting segmentation algorithm based on the watershed transform is proposed. Authors manually initialize the algorithm in order to distinguish athletes’ uniforms from turf areas.

In [9] two distinct modules for player segmentation have been implemented as plug-ins in a more complex architecture; in [15] information about the principal color in the image (the field) and the topology of the field are integrated to model the background by means of a mixture of Gaussians in the HSI color space. The main drawback of this
approach is its dependence from several thresholds manually defined. Thresholds are also critical for the work proposed in [16]: authors have manually defined some values for H, S and I components to distinguish the field from the players. A different approach is proposed in [6]: here the goal of the work is the tracking of players, and authors try to make their algorithm insensible to the motion detection phase: it can be used even in presence of moving cameras, the only requirement is the manual initialization of the regions (players) to be tracked. The main goal of this work is to develop both a robust foreground detection algorithm and a tracking process for soccer scenes analysis which are able to detect, classify and track players in complex situations with varying light conditions, high frame rate, and real time processing. The first part of the paper deals with an approach based on background subtraction for the segmentation of players. It is presented a background modeling algorithm able to face all the crucial situations typical of a motion detection system with an unsupervised approach; no assumptions about the presence/absence of foreground objects and changes in light conditions was required. So it is particularly suitable for sport scene analysis, where the constraint of absence of actors during the initialization phase is practically unachievable. The main idea is to exploit the pixels energy information in order to distinguish static points from moving ones. The proposed approach seems to be a good trade-off between efficiently and computational requirements. We started from a well known approach, and we have generalized it in order to create an algorithm able (i) to create a reliable model of the background even in presence of moving objects; (ii) work in real time. At this moment the proposed approach is integrated in a system for realtime offside detection, and at the present time, the optimized frame rate is 21 fps (just a little bit less then the required 25 fps). The second part of this paper faces up to the players tracking issues. In soccer scene analysis extracted object features are neither uniform nor static and multiple occlusions make recognition difficult. All the approaches based on initial object segmentation and temporal following fail when a single connected region contains multiple objects or when a single object is segmented in more regions. It is proposed a stochastic approach for foreground people tracking based on the evaluation of the maximum a posteriori probability (MAP). In order to solve the split events it is also necessary to know the team of each player. For this reason we have implemented an unsupervised classification step which uses color information of each segmented blob. The classes are created by means of a modified version of the BSAS algorithm [12] (Basic Sequential Algorithmic Scheme). This player team information is used by the tracking algorithm to resolve each singular blob splitting events.

The remainder of this paper is organized as follows. Section 2 describes the moving object segmentation algorithm. The soccer team classification is reported in Section 3. In Section 4 the tracking algorithm is outlined. Finally Experimental results and Conclusions are reported in Section 5 and Section 6.

2. Moving Object Segmentation

The most used approach for motion detection is background subtraction. It is based on the comparison between a reference image and the current one. So the background creation and updating procedure becomes a crucial step in the whole player classification procedure. We have integrated our idea with a very common and useful background subtraction approach [5]: a pixel \((x, y)\) is considered as a moving one if it differs from the background model \(B(x, y)\) more than two times the standard deviation \(V(x, y)\). The implemented background modeling algorithm uses the energy information of each image point, evaluated in a small sliding temporal window, to distinguish static points from moving ones. In this way we are able to obtain a statistical background model with only the contribution of background points, without the effects of foreground objects.

The basic idea is to analyze in a small temporal window the energy information for each point: the statistical values relative to slow energy points (static points) are used for the background model, while they are discarded for high energy points (moving points). A coarse-to-fine approach for the background modeling is applied in a sliding window of size \(W\) (number of frames). The first image of each window is the coarse background model \(B_C(x, y)\). In order to have an algorithm able to create at runtime the required model, instead of building the model at the end of a training period, as proposed in many works, the mean and standard deviation are evaluated at each frame; then, the energy content of each point is evaluated over the whole sliding window, to distinguish real background points from the other ones. Formally, for each frame the algorithm evaluates mean and standard deviation:

\[
\overline{\mu} = \alpha \mu + (1-\alpha)\overline{\mu}^{t-1} \tag{1}
\]

\[
\sigma^t = \alpha |\mu - \overline{\mu}| + (1-\alpha)\sigma^{t-1} \tag{2}
\]

only if the intensity value of that point is substantially unchanged with respect to the coarse background model, that is:

\[
|I^t(x, y) - B_C(x, y)| < th \tag{3}
\]

where \(th\) is a threshold experimentally selected. In this way, at the end of the analysis of the first \(W\) frames, for each point the algorithm evaluates the energy content as follows:

\[
E(x, y) = \int_{t \in W} |I^t(x, y) - B_C(x, y)|^2 \tag{4}
\]
The first fine model of the background $B_F$ is generated, as

$$B_F(x, y) = \begin{cases} (\mu(x, y), \sigma(x, y)) & \text{if } E(x, y) < th(W), \\ \phi & \text{if } E(x, y) > th(W). \end{cases}$$

(5)

where $\phi$ is the empty set, and $th(W)$ is a threshold that depends on the size $W$ of the sliding window. The whole procedure is iterated on another sequence of $W$ frames, starting from the frame $W+1$. The coarse model of the background is now the frame $W+1$, and the new statistical values Eq. 1 and 2 are evaluated for each point, like as the new energy content 4. The relevant difference with 5 is that now the new statistical parameters are averaged with the previous values, if they are present; otherwise, they become the new statistical model values. So, the new formulation of 5 becomes:

$$B_F(x, y) = \begin{cases} (\mu(x, y), \sigma(x, y)) & \text{if } E(x, y) < th(W) \land B_F(x, y) = \phi \\ \beta \ast B_F(x, y) + (1 - \beta)(\mu(x, y), \sigma(x, y)) & \text{if } E(x, y) < th(W) \land B_F(x, y) \neq \phi \\ \phi & \text{if } E(x, y) > th(W). \end{cases}$$

(6)

The parameter $\beta$ is the classic updating parameter introduced in several works on background subtraction [5,6]. It allows to update the existent background model values to the new light conditions in the scene. The whole procedure is iterated $N$ times, where $N$ could be a predefined value experimentally selected to ensure the complete coverage of all pixels. Otherwise, to make the system less dependent from any a-priori assumption, a dynamic termination criteria is introduced and easily verified; the modeling procedure stops when a great number of background points have meaningful values:

$$\#(B_F(x, y) = \phi) \approx 0$$

(7)

After the creation of a reliable model, the segmentation algorithm is able to detect moving objects; the whole procedure above described is then iterated for the maintenance of the background model for all time. Weather conditions, as well as light changes, have been coped with an appropriate value of the updating window $W$. In figure 1 an image of the output of the segmentation procedure is shown. A connectivity analysis has been implemented on the segmented images, removing items with small areas.

3. Team Player Detection

The classification procedure is composed of two steps: first of all during an initial training phase, the classes are created by means of a modified version of the BSAS algorithm [12] (Basic Sequential Algorithmic Scheme); it is an unsupervised approach substantially independent of human interaction. Then, at runtime, each foreground object is assigned to one of the classes previously extracted. The first step requires the definition of proper features for each segmented region, and the definition of a distance measurement on these features. In particular the features must be able to both discriminate different classes of interest and keep down the computational costs. Normalized Histograms in the RGB color space are a good trade-off between these two requisites. Color histograms present the advantage of being independent of the posture of players and the normalization process guarantees more stable results with respect to scaling factors. Using normalized color histograms, images become points in the feature space and the class recognition becomes a clustering problem. The metric used for estimating the "proximity" of two histograms is the Manhattan distance which has the advantage of being simple and fast with respect to other more complex distance measures. For the cluster building phase, a training set is created by se-

Figure 1. The image with the blob information after the connectivity analysis.

Figure 2. The image with the classification results.
lecting randomly a certain number of foreground objects. These objects, represented by their normalized histograms, are provided to the BSAS algorithm that detects a number of separate clusters. Feature vectors are presented only once and the number of clusters is not known a priori. It is needed a similarity measure \( d(X, C) \) between the feature vector \( X \) and the cluster \( C \), a threshold \( \tau \) on this measure, and the number \( q \) of the maximum allowed clusters. The algorithm assigns every newly presented vector either to an existing cluster or creates a new one for this sample, depending on a similarity measure \( d \) and the number of clusters is not known a priori. It is needed more details about this algorithm can be found in \[12\]. In our work we have lightly modified the classic implementation of this algorithm. In particular, we have chosen to fix the threshold \( \tau \) to a small value, that is increased if the number of detected clusters exceeds the predefined value \( q \). In this way the algorithm converges to the correct cluster configuration with the best value of \( \tau \). Moreover, in order to be independent of the order in which the samples are presented a merge procedure is carried out on the output clusters, using \( \tau \) as a merge threshold: if the distance between two clusters is less than \( \tau \) they are merged, and the clustering procedure is started again. In case of an inadequate numbers of clusters, a new training set is built, and the whole training procedure is repeated. The last situation happens when the training set is composed only by players belonging to the same team. In the proposed applicative context, with the chosen cameras configuration, each processing unit (linked to a particular fixed camera that observes a part of the scene) can found a number of classes that can vary in a given range (in the middle of the field two or three classes can be found: two teams and, eventually, referee; on the other hand, the lateral cameras can observe 2, 3 or 4 classes). So we have decided to fix only the max number (5) of classes; the training set is automatically built in a few seconds without human interaction, so it is possible that some classes (i.e., referee) can be not represented in it. So a check on the output clusters, and a check on the final clusters distance, is necessary to validate them. At the end of this phase, a logical label is manually assigned by the human operator to each cluster, creating the correspondence between five classes and the two teams, the two goalkeepers, and the referees. At runtime, for each region extracted by the foreground segmentation procedure, the relative feature vector (normalized color histogram) is compared with the representative of each cluster (called prototype); the Manhattan distance is used again to select the winner cluster by means of a minimum distance criteria. Then, the winner prototype is updated in order to adapt itself to possible variations in light conditions; this updating is carried out by means of a weighted mean as follows:

\[
C_k = \frac{1}{n_k + 1}(n_k C_k + X)
\]  

(8)

where \( C_k \) is the prototype of the winner cluster \( k \), \( X \) is the feature vector of the examined objects, and \( n_k \) is the weight of the cluster \( k \), i.e. the number of objects classified as belonging to the cluster \( k \) before the updating. In figure 2 the output of the classification procedure is shown.

### 4. Player Tracking

The multiple players tracker uses the Maximum a Posteriori Probability to get the best fit of state and observation sequences. The state vector includes information about the location, velocity, acceleration and dimension of the single bounding box (BB). We denote with \( N_t \) the maximum number of predicted people in the image at time \( t \). Each BB is identified by the index \( i \) with \( i \in \{1...N_t\} \) and represented by a vector \( x^i_t = (p^i_{x_t}, v^i_{x_t}, a^i_{x_t}, d^i_{x_t}, l^i_{x_t}, c^i_{x_t}, s^i_{x_t}) \) where

- \( p^i_{x_t}, v^i_{x_t}, a^i_{x_t}, \) and \( d^i_{x_t} \) represent the BB position, velocity, acceleration and dimension respectively.
- \( c^i_{x_t} \) is the BB status. It assumes the values: 1 for a single blob, 2 for a merge blob, 3 for an exiting blob, 4 for a disappeared blob, 5 for solved blobs in a merge blob, 6 for incoming blobs.
- \( l^i_{x_t} \) is a single label if the blob is a single blob, or a set of labels if the blob is a merge blob.
- \( s^i_{x_t} \) is a single class number if the blob is a single blob or a set of class numbers if the blob is a merge blob.

We denote the multiple people configuration at time \( t \) with \( X_t = \{x^i_t | i = 1..N_t\} \) where \( N_t \) is described above. In the same way we describe all the blob state observation \( Z_t = \{z^j_t | j = 1..M_t\} \) where \( z^j_t \) are the observation instance vectors \( z^j_t = (p^j_{z_t}, c^j_{z_t}, d^j_{z_t}) \), \( M_t \) is the number of BB actually observed at time \( t \). For the trajectory tracking, a configuration sequence that maximizes the \( P(X_t|Z_t) \) is computed which is the a posterior probability distribution over states conditioned on observations.

The Bayes rule provides a tool for reasoning probabilistically about a given state sequence \( X \) and observation image sequence \( Z \):

\[
P(X_t|Z_t) = \frac{P(Z_t|X_t)P(X_t)}{P(Z_t)}
\]

(9)

The observations \( Z_t \) are the results of prior steps: segmentation and classification. They are independent of the previous instances, but to eliminate false blobs (after background updating many artifacts could be introduced in the segmented image) they have to be validated by consecutive observations. This probability is evaluated by a function that considers the maximal number of players allowed in the field, and the camera position (cameras situated on
the middle of the field have a greater probability of observing more players than lateral cameras), and the persistence of observations of the same blob with the identical class in successive frames. Furthermore, we have to consider several factors: player position changes, players who exit from the field of view, blob merging, splitting and occlusion. The prediction of people configuration at time \( t \) is usually evaluated as a function of blob position. The commonly used model for the blob estimation \( p_{x_t} \) is

\[
p_{x_t} = A p_{x_{t-1}} + \Sigma w_{t-1} \tag{10}
\]

where \( A \) is the state transition matrix, \( \Sigma \) is the disturbance matrix and \( w_{t-1} \) is the normal random noise with zero mean and covariance \( Q \). The values of \( A \) are set to define the possible movements of each blob considering both the camera frame rate and the maximum admissible blob velocity.

Assuming independent movement of each person, the state transition dynamics can be evaluated as follows:

\[
P(X_t) = \prod_{i=1}^{N_t} \prod_{i}^{(X_{t-i})} ((x_{t-i}, x_{t-i+1}, \ldots, x_{t-N_t}),
\]

\[
(x_{t-i}, x_{t-i+1}, \ldots, x_{t-N_t}), \ldots) \tag{11}
\]

Each term of this product is estimated using a prediction model for the blob position estimation and considering several factors:

- \( x_{t-1} \) could be a single track, if there is a previous blob \( x_{t-1} \) whose predicted position is in the actual frame image and we assign \( l_{x_t} = l_{x_{t-1}} \) and \( s_{x_t} = s_{x_{t-1}} \);
- \( x_{t-1} \) could be a merge, if there are two or more blobs \( (x_{t-1}, x_{t-1}, \ldots) \) whose predicted positions fall close in the actual image and we assign \( l_{x_t} = l_{x_{t-1}} \cup (l_{x_{t-2}}, \ldots) \), \( c_{x_t} = c_{x_{t-2}}, \ldots \) and \( s_{x_t} = 2 \);
- \( x_{t-1} \) could be an outgoing blob if there is a blob \( x_{t-1} \) whose predicted position is outside the actual image;
- \( x_{t-1} \) could be a resumed blob if there is not a correspondence with any blob in the previous frame, but is found with someone in the paste instances;
- \( x_{t-1} \) could be an incoming blob if there is a blob entering into the scene and its state is initialized to \( s_{x_t} = 6 \);
- \( x_{t-1} \) could be a new blob coming from a split event caused by a merge blob.

The last three situations cannot be predicted by the past blob states history, but have to be validated by observations.

The blob observation likelihood measures how well the calculated configuration fits the current observation. It can be computed as a matching score between the estimated blob configuration and the actual players location. We propose a likelihood function based on the evaluation of the blob distances as follows:

\[
P(Z_t | X_t) = \frac{\sum_{h=1}^{M_t} P(z_{t|h} | x_1^t, x_2^t, \ldots, x_{N_t}^t)}{M_t} \tag{12}
\]

where

\[
P(z_{t|h} | x_1^t, x_2^t, \ldots, x_{N_t}^t) = M \max_{i \in (1, N_t)} (e^{-|p_{x_t} - p_{x_{t-1}}|^2}) \tag{13}
\]

The configuration providing the Maximum a posteriori probability in (12) is the one selected as best fit between state observation and state prediction. At this point a further step for the prediction validation has to be carried out.

By comparing the matches between the observations \( z_{t|h} \) with \( h \in (1, M_t) \) and the predictions \( x_{t|h} \) with \( i \in (1, N_t) \), two possible situations may happen: there are some observations \( z_{t|h} \) that do not match any prediction and there are some predictions \( x_{t|i} \) that do not correspond to any observation. In the first case different situations may be occurred: 1) the observation \( z_{t|h} \) could be a new entry blob if \( p_{x_t} \) is on the image border, then a new \( x_{t,i} \) is generated with an incoming state \( s_{x_t} = 6 \); 2) the observation \( z_{t|h} \) could be a resumed blob if it is close to a prediction \( x_{t,i} \) with a disappeared state; 3) the observation \( z_{t|h} \) could be generated by noise, then a new entity \( x_t \) is created and observed along a temporal window until a decision on its persistency is taken. In the second case, if the prediction \( x_{t|i} \) has not a correspondent among the observations and it is not on the image border (it is not in a outgoing situation), it means that the foreground segmentation step was not detecting the blob and then the state vector is maintained setting \( s_{x_t} = 4 \) (disappeared blob). An additional analysis is required when merge event occurs between two or more blobs. We can predict that two or more blobs will merge (we set the status equal to 2), but since we need to maintain their vector status separated we have to split logically them in the corresponding observation. This splitting procedure could be difficult especially when two or more players are very close to each other and the occlusion is almost total. However when a merge blob is detected by the tracking algorithm, it also maintains the information about the class numbers of the grouped players and the labels identifying the single tracked blobs. Starting from this information, a logical splitting (inside the merge blob) procedure evaluates the blob area and searches for sub-regions having the same color features of those searched. The search starts from the positions predicted by the single blob vector status and it is enlarged as soon as the algorithm finds the best match. At the end of this step the state vector of each segmented blob is maintained by updating its positions in the merge blob and setting the status \( s_{x_t} = 5 \).
(solved blobs in a merge blob). The maintenance of state vectors for solved blobs in merge blobs allows us to recognize physical splitting situations. In fact when a physical split occurs (a single blob at time $t-1$ is divided in two or more blobs at time $t$) we have that two or more observations $z_l^t$, $z_l^t$, $z_l^t$ are matching both with a prediction $x_l^t$ with $s_{x_l}^t = 2$ (merge status) and with two or more predictions $x_l^m$, $x_l^n$, $x_l^t$ with $s_{x_l}^m = 5$, $s_{x_l}^n = 5$, $s_{x_l}^t = 5$ (solved blob). According with the number of objects in the merge blob and the number of observations we discriminate between a simple split or a complex split in single blobs and merge blobs. The decision is taken considering for each observation the blob dimension, the color features and the best correspondence with the prediction with a solved status. In this way we assign to the single blob $x_l^m$ the new status $s_{x_l}^m = 1$ (single track) and we remove or modify the merge blob $x_l^t$ with $s_{x_l}^t = 2$ reducing its number of internal objects.

5. Experimental results

The experiments were carried out on soccer sequences acquired by proprietary Color Cameras with $25fps$ and resolution of $1920 \times 1080$. Our algorithms run during a whole game. The first few minutes were used to build the background model for the foreground segmentation algorithm and to build the prototypes of the five clusters corresponding to the two soccer teams, the two goal-keepers, and the referee team. After this unsupervised learning phase, our classification and tracking algorithm was able to detect and follow players and referee moving in the pitch during all the match. A visual evaluation of the real time processing allows only a qualitative analysis of the effectiveness of our classification and tracking results. In order to have a quantitative evaluation it is necessary to establish a ground truth against which compare the results of the tracking algorithm. We have extracted an image sequence and off line, for each image, we have manually estimated the player positions and associated the corresponding classes. Along this sequence we have evaluated the ability of our tracking algorithm to separate players in merge situations, to manage new entering and exiting players, and above all the capability to overcome the problems due to mistakes of the foreground detection algorithm such as false moving areas or lacking/partial segmentation of players.

In table 1 the blob tracking performance of four players is reported. The players are always correctly tracked when they are seen as single blobs. The first two players remain for 86 frames in a merge blob, and the last two players for 24 frames. Even if the tracking algorithm maintains the status of merge blobs, the classification procedure is not always able to separate the blobs. According to the quantity of occlusion between players, some cases occur in which the players are not separated. In figure 3 we show the results of blob splitting using the color features. The yellow blobs represent the merge blobs with the status ”solved”. The interior blobs of different colors represent each segmented blobs.

An evaluation of the tracking algorithm in terms of false positive detection has also been provided. Due to the complex experimental context, outdoor environment with sudden and strong light variations, the foreground segmentation step is exposed to many false moving areas. Experimental evidence demonstrated that these noise blobs do not remain for long time in the image. They disappear as soon as the background model is updated, and only real moving areas persist in the image. The proposed tracking algorithm is able to evaluate the persistency of each blob and to discard false positive areas when their lifetimes are under a reasonable number of frames. In our test sequence a number of false moving blobs were erroneously generated by the segmentation step. In table 2 the number of tracked frames for each blob is reported. The longest tracked blob (20 frames) is the second one that however can be easily removed as a false positive candidate because, unless its position is on the border of the image, it cannot represent a player that enters and disappears in the field.

<table>
<thead>
<tr>
<th># tracked frames</th>
<th>Player 1</th>
<th>Player 2</th>
<th>Player 3</th>
<th>Player 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>86</td>
<td>86</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>2</td>
<td>71</td>
<td>75</td>
<td>20</td>
<td>18</td>
</tr>
</tbody>
</table>

Table 1. Number of tracked merge blobs for four players.

<table>
<thead>
<tr>
<th>Blob Number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td># tracked frames</td>
<td>4</td>
<td>20</td>
<td>12</td>
<td>11</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 2. False positive blobs: number of tracked frames.
A stochastic approach for player tracking based on the evaluation of the maximum a posteriori probability has been used in order to track players during matches. Experimental tests have been carried out on soccer image sequences acquired during the Italian “Serie A” soccer championship 2006/2007. The obtained results show that the algorithm is able to correctly detect foreground objects and separate the merging blob in the relative single players. As future works we are working in the direction of decreasing computational time in order to realize a real time system for sport players tracking.

<table>
<thead>
<tr>
<th># Frames</th>
<th># Segmented Blobs</th>
<th># Tracked Blobs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person 1</td>
<td>776</td>
<td>739</td>
</tr>
<tr>
<td>Person 2</td>
<td>188</td>
<td>164</td>
</tr>
</tbody>
</table>

Table 3. Percentage of blob tracking on a CAVIAR sequence with two people.

It is difficult to compare our method with previous ones due to variability in data sets and lack of access to the earlier methods’ code. In order to test our algorithm, we did several experiments also on public data sets. We evaluated our tracking algorithm on a CAVIAR sequence [1] which was captured with a stationary camera and a frame size of 384x288. It is an un-occluded sequence only (in this experiment we skip the classification step that was devised for the soccer context) because we want to show the tracking ability to maintain and follow the same objects even when the foreground detection algorithm makes mistakes. In table 3 the results are reported. The first column represents the ground truth extracted from the CAVIAR data set for the two people entering in the camera view. In the second column the results of the foreground segmentation process are reported. Due to the low image resolution of the sequence and the necessity of our segmentation algorithm to build a background models in the first frames, when the two people enter in the scene they are non correctly detected. Furthermore, the segmentation algorithm has also some problems in segmenting correctly the first person giving some frames in which there is an interruption in the continuity of blob detection. In the third column the results of the tracking algorithm are shown. The first person is correctly tracked, maintaining the same identification label, also when the foreground detection fails. In this case we had three jumps of 2,3 and 5 frames in which the person wasn’t segmented, but the tracking algorithm recognized the same object as soon as it appeared again in the image.

6. Conclusions

In this paper a detection and tracking algorithm applied to soccer players is presented. The proposed algorithm is able to work in complex situations with varying light conditions, high frame rate image sequences. The segmentation procedure is able to create a reliable statistical model even in presence of moving objects, typical situation of sport scene analysis. It is carried out by evaluating pixel energy contents in temporal sliding windows. The foreground objects are then classified according to the belonging team by means of an unsupervised clustering algorithm that is a modified version of standard BSAS. The features we have chosen to use are color histograms in RGB space. A stochastic approach for player tracking based on the evaluation of the maximum a posteriori probability has been used in order to track players during matches. Experimental tests have been carried out on soccer image sequences acquired during the Italian “Serie A” soccer championship 2006/2007. The obtained results show that the algorithm is able to correctly detect foreground objects and separate the merging blob in the relative single players. As future works we are working in the direction of decreasing computational time in order to realize a real time system for sport players tracking.

References

