Chapter 14
Real-Time Event Detection in Field Sport Videos

Rafal Kapela, Kevin McGuinness, Aleksandra Swietlicka and Noel E. O'Connor

Abstract This chapter describes a real-time system for event detection in sports broadcasts. The approach presented is applicable to a wide range of field sports. Using two independent event detection approaches that work simultaneously, the system is capable of accurately detecting scores, near misses, and other exciting parts of a game that do not result in a score. The results obtained across a diverse dataset of different field sports are promising, demonstrating over 90 % accuracy for a feature-based event detector and 100 % accuracy for a scoreboard-based detector detecting only scores.

14.1 Event Detection in Sports

Sport has always been one of the most popular of television broadcasts [1, 2]. Sports broadcast viewing figures are consistently high, particularly for events like the Olympics and national or regional finals of nationally popular games. Given such wide appeal and popularity, there has been significant interest in algorithms for automatic event detection in sports broadcasts [3]. This is motivated by potential applications such as automatic highlight generation for summarization for emerging second screen applications, indexing for search, and retrieval in large archives, mobile content delivery either offline or as an added value in-stadium user experience.

R. Kapela (✉) · A. Swietlicka
Faculty of Computing, ul. Piotrowo 3A, Poznan, Poland
e-mail: rafal.kapela@put.poznan.pl

A. Swietlicka
e-mail: aleksandra.swietlicka@put.poznan.pl

N.E. O'Connor · K. McGuinness
Insight Centre for Data Analytics, Dublin City University,
Glasnevin, Dublin 9, Ireland
e-mail: noel.oconnor@dcu.ie

K. McGuinness
e-mail: kevin.mcguinness@dcu.ie

T.B. Moeslund et al. (eds.), *Computer Vision in Sports*,
Advances in Computer Vision and Pattern Recognition,
DOI 10.1007/978-3-319-09396-3_14

Many of these applications either require or would benefit significantly from real-time event detection, however, this aspect has been largely overlooked to date in the relevant literature.

14.1.1 Sport Event Detection Approaches

A range of event detection algorithms have been presented in recent years, taking into account the broad diversity of different sports. It has been shown in [4] that about 97 % of interesting moments during a game are followed by a close-up shot presenting a player who scored or who caused some interesting action. In addition, features like end of a pitch, audio activity, or crowd shot detection have been shown to be very useful in event detection [4]. The system present in [4] was proven to work with different sports such as soccer, rugby, field hockey, hurling, and Gaelic football. A Support Vector Machine (SVM) was used as a event classifier. However, the algorithm was too computationally expensive for real-time implementation, mainly due to the use of the Hough transform.

A very similar approach was presented in [5]. To detect an event the authors declare so-called "plays" where mainly a color histogram is calculated and some heuristics are applied about the regions of histogram detection. An event is categorized using Hidden Markov Models (HMM) based on the sequence of camera shots. In this work events were detected in baseball, American football, and Japanese sumo wrestling. Another example of this kind of approach was presented in [6] where, based on simple visual features like pitch orientation and close-up detection, the authors achieved good accuracy. However, computation time is not evaluated in the paper and there is a large drop in accuracy when the SVM is trained on the samples that do not belong to the same game.

It is worth noting that the three approaches described above [4–6] are capable of extracting not only goals but also other exciting moments like penalties and near misses. In [7], very simple features like pixel/histogram change ratio between two consecutive frames, grass ratios, and background mean and variation in addition to time and frequency domain audio features were used to detect events in soccer games. The authors report high-accuracy using simple features, but again do not discuss computation time performance.

Although the acceptance of the MPEG-7 standard in the community has been rather low, there are approaches based on MPEG-7 descriptors. In [8] a goal detection system based only on MPEG-7 audio descriptors was proposed. Some implementations of the proposed feature extraction process are very fast and make the system applicable in real-time scenarios. However, the system is designed for soccer games alone, does not detect anything other than goals, and was tested on very few test samples (only 8 goal scenarios).

One event detection technique extensively investigated for broadcast videos is replay detection, since replays already contain interesting moments selected by a director. Some examples of these techniques for either slow or normal motion replay

detections are presented in [9–11]. Although all of these feature very good precision and recall (from about 60 to 100 %), event detection based on replay extraction techniques are not really appropriate for real-time applications (i.e., they occur after the real event and a specific one may occur several times during a game so this approach requires building some sort of indexing system to store and match following scenes).

Finally, in [12, 13], very different approaches are taken. The former utilizes the information produced by people during a game and tweeted by the popular Twitter website to detect events in different games (soccer and rugby were tested). The latter approach uses web-casted text for the same purpose. These are, at first sight, universal approaches, however they can suffer from quite large false positive detection rates, need constant connection to the Internet and introduce some ambiguity in the form of delay between detected and real events making the detection of event boundaries more difficult.

14.1.2 Field Sports as a Specific Genre of Team Sports

In general, from a content analysis point of view, sport broadcasts can be classified into two main categories [3]: a single/dominant camera or multipoint/view camera capture broadcasts. In sports like tennis or badminton, there is typically one camera view that dominates the broadcast (e.g., the full court in tennis) and contains almost all the semantically meaningful elements required to analyze the game. Yet in sports like football, hurling, soccer, baseball, hockey, basketball, there is no such well-defined camera view that dominates the course of play. This is mainly due to the fact that the playing field is too large to present all the necessary details with a single camera broadcast.

Most existing event detection algorithms focus on a particular type of sport (e.g., tennis, soccer, cricket) and are not robust for other types of sports, thereby limiting their applicability. This reflects the fact that different sports present different characteristics either in the rules for that sport or the manner in which they are captured for broadcast. So far there have been very few attempts to make event detection more general so that the same system/algorithm could be used for more than just a single sport. Although this may seem to be complex and challenging it could be very convenient from the perspective of a sports broadcaster, avoiding the need for development and deployment of multiple different algorithms and systems. For this reason, in this chapter we focus on a generic subset of all sports that can be designated as "field sports", a term originally introduced in [4] to refer to any sport played on a grass pitch (soccer, rugby, field hockey, etc.) featuring two teams competing for territorial advantage. In this work, however, we extend this definition to include other sports that exhibit similar characteristics but that are not necessarily played on a grass pitch. Specifically, we extend the definition of field sports to include sports played in a playing arena that features some kind of scoring posts (e.g., goal post in soccer or basket in basketball), whereby the overall objective is territorial advancement with a view to

obtaining a score. Since most field sports have similar characteristics, broadcasters use the same scene compilation techniques to present the action on the field to the viewer. Thus, once a robust and accurate algorithm is developed, it can be applied to different kinds of field sports without any modification.

14.2 The Architecture of the Event Detection System

The top-view architecture of the system is presented in Fig. 14.1. After video decoding the stream analysis is split into two independent threads. The first, termed the *scene thread*, works on extracted video frames and audio packets. The second, the *scoreboard thread* needs only video frames. The common data is stored in a shared buffer in order to simplify the processing flow. The two threads are described in more detail in the following sections. All their features are presented in the bottom-up approach—we start with a description of the low-level modules that gather the data extracted by feature detectors. All these data are then sent to the mid-level modules where, depending on the processing thread, it is either used to detect the text (scoreboard) region or predict what kind of scene is presented to the viewer. The top module gathers the events from both threads, summarizing the description of the system. The inputs (i.e., recognized score change or specific—event related sequence of shots) to this module are treated separately which means that there are two kinds of event markers at the system output. In general the primary design constraint was the latency introduced by the modules working at all levels throughout the system. This is reflected in the following sections, where time efficiency is the main parameter investigated in each module. The two threads are described in detail below.

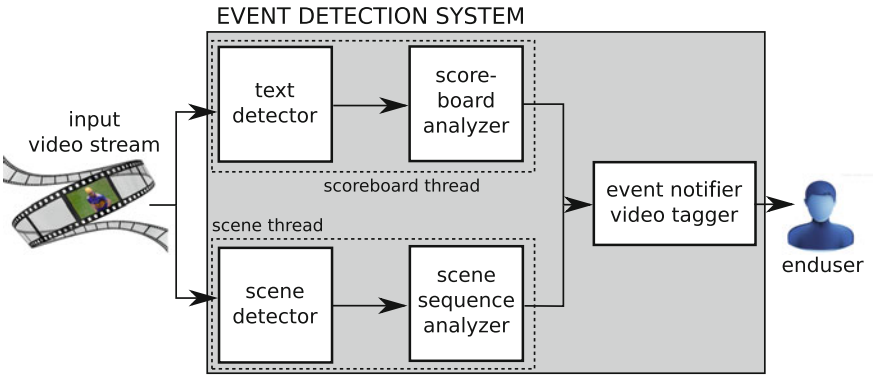

Fig. 14.1 The block schematic of the system—the two parallel threads are responsible for the analysis of the specific aspects of the video stream in order to determine if the particular part of the video is interesting from the user's point of view

14.3 Feature-Based Event Detection System

The scene analysis thread is responsible for detection of events that can subsequently be observed on the scoreboard (i.e., goals) but also situations like close-misses or any other actions that can be classified as interesting from the viewer's point of view. It belongs to the class of the algorithms that analyze the sequence of shots presented. Earlier works on event detection state that 97 % of the interesting moments are followed by close-up shots (i.e., shots that present a player) [4]. If we add to this the information about where the action has taken place (e.g., end of pitch/court—close to the goal-post/basket) before the close-up shot occurred, we can more precisely distinguish between usual and unusual events.

The scene analysis thread utilizes extracted video frames and audio packets in order to detect the events. Most of the information needed for this task is stored in visual data. For this reason we have chosen 14 different classes that represent most of the situations presented on the field/court during a match. In order to evaluate the generalization properties of the algorithm some of the classes were split into three subclasses: shot with simple background (a dominant color easily distinguishable), shot with complex background (no dominant color in the background) and a mixture of the two—Table 14.1.

However, we have also included simple audio analysis for better robustness of the algorithm. The reason for this was that we have observed how usually the interesting moment in the game is followed with increased audio activity intervals (e.g., round of applause or excitation of the commentator). From a real-time perspective audio track analysis does not introduce significant overhead in processing time since the decoder does audio extraction in parallel to the analysis of the video stream and moreover, we are only calculating a temporal energy of the audio signal which is a relatively simple operation.

Table 14.1 The abbreviations of descriptors used in the system

Abbreviation	Descriptor focus
CLHS	Close up shot head with simple background
CLHC	Close up shot head complex background
CLH	Close up shot head mixture background
CLWUS	Close up shot waist up simple background
CLWUC	Close up shot waist up complex background
CLWU	Close up shot waist up mixture background
SPS	Short distance shot presenting player(s) simple background
SPC	Short distance shot presenting player(s) complex background
SP	Short distance shot presenting player(s) mixture background
SS	Short distance shot presenting spectators
LC	Long distance shot presenting center of the field
LR	Long distance shot presenting right side of the field
LL	Long distance shot presenting left side of the field
LS	Long distance shot presenting spectators

14.3.1 Feature Detection

As mentioned in the introduction, to date there have been many features proposed to detect different scenes in a game. However, the majority of these approaches are either not suitable for real-time analysis of the game or, taking into account that we deal with not only one type of the sport, too simple to distinguish robustly between the type of the scenes. For these reasons we had to propose our own scene detection techniques [14] or simplify the existing ones to meet the real-time requirement.

In [14] we compared state-of-the-art image description techniques and proposed a dataset of images especially chosen for this task. The mentioned dataset meets two very important requirements in order to provide a tool for unique categorization of scene detection algorithms:

- it contains variety of field sports (specifically, soccer, rugby, Gaelic football, hurling, basketball, and cricket);
- provides a representative variety of images that can be assigned to particular classes (i.e., 14 types of scenes ranging from head close-up shots to shots with long perceptive distance to the players).

Our work in [14] shows that local feature detection algorithms like Scale Invariant Feature Transform (SIFT) [15] and Histogram of Oriented Gradients (HoG) [16] algorithms, that are characterized with the highest accuracy are too slow to be part of a system that has to work under real-time constraints. More recent local feature descriptors have been proposed, similar to SIFT and Speeded-Up Robust Features (SURF), which are much faster to compute as they produce a binary output (e.g., to produce the response, the algorithm only has to compare a fixed number of region intensities around the key-point). These include: Binary Robust Independent Elementary Features (BRIEF) [17], Fast Retina Keypoint (FREAK) [18], Binary Robust Invariant Scalable Keypoints (BRISK) [19] and An Efficient Dense Descriptor Applied to Wide Baseline Stereo (DAISY) [20]. Depending on the descriptor length and local region filtering they could be from several up to several tens of times faster than the SURF local descriptor [21, 22]. The fastest is BRIEF (up to sixty times faster than SURF) ensuring at the same time an acceptable recognition rate [22]. Note, that a scene description algorithm for sport videos does not need to provide scale and rotation invariance for two reasons:

- the broadcasted videos always present non-rotated scenes;
- we want to distinguish between long and close-up shots of the players and the field/court, thus the scale invariance has to be restrained to some, albeit relatively small, range.

As the BRIEF descriptor satisfies the above considerations and is computationally efficient, it was chosen for this application scenario.

14.3.2 Feature Aggregation

Once a set of binary local descriptors have been extracted for a frame, a method is needed to aggregate these into a single fixed-length descriptor that can be used for classification. The most widely known technique is the bag of visual words method, in which a histogram of local descriptors is created from a pre-trained codebook where each bin represents a cluster of similar local descriptors. This very simple and effective algorithm is also suitable for binary data and outperforms, in terms of computational complexity, other state-of-the-art techniques, for example, Fisher kernels [23] or VLAD [24].

Bag of visual word aggregation is usually implemented using k-means to generate the code book. However, clustering binary data with k-means algorithm is not straight forward as descriptors should be compared using Hamming distances rather than Euclidean, and substituting Euclidean distances may lead to unexpected results [25]. We therefore took a different approach for cluster generation. The algorithm is as follows:

1. Calculate local descriptors for all images from the training set.
2. For each of these, find N nearest neighbors based on Hamming distance (based on a number of experiments we noticed that N should be equal to 2–5 % of the number of descriptors, in our case $N = 150$). The sum of all the distances to a particular node is an estimate of a local density around it.
3. Sort the list by the local density estimate.
4. Choose a vector from this list, either from the top, bottom, or randomly, and store it as a cluster.
5. Remove the vector that produced the cluster and all its neighbors.
6. Repeat points 4–5 until all clusters are selected.

Intuitively there would be tendency to pick only clusters from the top, but experiments show that also selecting random vectors and vectors from the bottom has a positive effect on accuracy. Accuracy is improved because the candidates with less neighbors simply represent the regions of lower densities. Experimental results are presented in the Sect. 14.5.

We applied a moving window approach where the average of the SVM responses over a fixed time period is calculated when detecting the 14 classes defined in Sect. 14.3. Then these 14 values and the audio intensity descriptor are sent to the event recognition module.

14.3.3 Event Recognition

Figure 14.2 shows an example of the behavior of the description vectors (outputs from scene recognition SVMs). For a close-up shot one can observe that all the descriptors prefixed with CL (close-up) are dominant, whereas in a long-distance

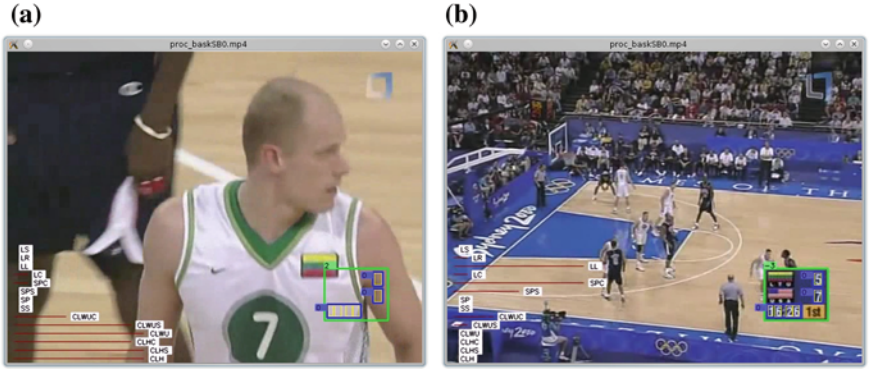

Fig. 14.2 Example of the behavior of visual descriptors (averaged responses from SVMs) **a** a close-up shot; **b** long-distance shot

shots the LL (long-left) descriptor is dominant with the additional feedback that some players/spectators detected could be at a short perceived distance to the camera. Note, that the difference between short and long type of shots is quite fuzzy (e.g., in basketball long type of shot presents players at a scale analogous to short distance shots in soccer, Gaelic football, or hurling). However, this does not negatively impact on effective event recognition.

Intuitively, an initial approach would be to train a deterministic classifier like an SVM to take all the responses of scene classifiers and detect if the shot presented belongs to an event class. Our experiments, however, show that we also need an additional input about previous SVMs responses and also responses from the event classifier.

14.3.3.1 State Machine

A state machine can be one of the simplest ways of singling out all of the important scenes from the given game. Tha main difficulty in this case is however the optimal choice of the number of states. After analysis of the correlation between the values of the descriptors and the ground-truth probes we have chosen four states:

A—an interesting event has been detected, e.g. there was a goal;
B—an interesting event just happened;
C—an idle state, when all the descriptors have values less than 0.5;
D—all other situations.

We assumed that the most interesting moments of the game take place at the end of the arena, so state A occurs when the descriptor of the end of the field or court is higher than 0.8, plus very often after an event, there is a close-up shot. This is reflected by keeping track of an upward trend of the set of close-up descriptors.

After analyzing the amount of transitions between the proposed states it is obvious that the biggest probability of transition is between the same states. This unfortunately leads to insufficient accuracy of around 46 %. This is caused by the fact that the state machine usually stays in the same state, and breaking the loop is almost impossible. This is unacceptable since we can of course reduce the number of states to two and propose a random classifier with an accuracy better than 46 %. As a conclusion of this analysis, these results forced us to look for other solutions and to consider dependency between descriptors at a higher level than just their values.

14.3.3.2 Decision Tree

Because of the real-time requirement on the system, using a decision tree for event detection was a natural choice to investigate. Also, going deeper into the analysis of values of each descriptor we decided to think about constructing a decision tree that would give the information about the significance of a single frame. We started with an analysis of the end-of-pitch/court descriptor since most of the events begin with this kind of scene and then they are followed with a close-up shot. Note, that from an event detection point of view we do not need to have the information about left or right side of a pitch/court. For this reason we shortened the number of elements in the description vector to nine values. The new one encompasses the following features: LLR equal to maximum of LL and LR; LC, LS, SP, SS, CLH, CLHS, CLHC, audio activity (AA).

Thus, for the end-of-pitch/court descriptor we can have two situations: the video sequence comprises or does not comprise an event. Table 14.2 shows both situations, where the first nine values refer to the audiovisual descriptors and the last one gives the information whether this frame belongs to an event or not. As can be seen, the first situation ensures that this scene is taking place at the end of the court but the value notifying about an event is false, which means that nothing interesting has happened. In the second case we have the opposite situation—a scene is not happening at the end of the arena but the frame belongs to an event. Accordingly, we can easily find two lines, where values of descriptors are exactly the same but can be distinguished by the event flag. This was a reason to create two separate decision trees. The first one assumes nine descriptors as values that classify each frame independently and the second one eighteen descriptors, including also the previous time instant.

Constructing the tree manually would result in a tree that has $11(N-1) \cdot 101$ nodes, where N stands for the length of the input vector. We would start with the first descriptor and consider all possible eleven values (i.e., the average of ten previous

Table 14.2 The disambiguation between scene descriptors

LLR	LC	LS	SP	SS	CLH	CLHS	CLHC	AA	EVENT
1.0	0.0	0.0	0.6	0.0	0.0	0.0	0.0	1.0	False
0.0	0.0	0.9	0.2	0.9	0.0	0.0	0.4	0.4	True

SVM 0/1 responses), each going to the next node representing the next descriptor which would also has eleven possible branches, etc. Then the last tree level would have 101 branches since this is the number of all possible values for audio descriptor. To avoid this and create an efficient decision tree we used the algorithm described in [26], based on the Gini diversity index, which measures the node impurity and is given with a formula:

$$g(n) = 1 - \sum_i \frac{n_i^2}{N^2} \tag{14.1}$$

where $n = (n_1, n_2, \ldots, n_k)$ is a vector of non-negative real numbers representing the number of examples of each class and

$$N = \sum_i n_i \tag{14.2}$$

represents the total number of examples at a given node. A node with just one class (a pure node) has the Gini index zero, otherwise the Gini index is positive. The algorithm is also available in MATLAB [26].

Depending on the amount of data and on the number of descriptors (9 or 19) the constructed tree has 2,000–2,500 nodes. In the case, with 9 descriptors we reached 94 % accuracy and with 18–99 % (see the Sect. 14.5 for details).

14.3.3.3 Artificial Neural Networks

An artificial neural network, thanks to its auto-associative memory capabilities, also seems to be an appropriate solution for the detection of interesting events since they can track regularities that occur in the sequences of values of descriptors, which are practically impossible to do with the naked eye. Optimizing this kind of classification method, since many local minima exist, is a non-convex problem but thanks to choosing a fixed but random starting point it gives good results and can be treated as a deterministic method.

We naturally considered at first a feedforward structure, where we simply used values of nine descriptors on the input and the network was trained to recognize if a given frame belongs to the event segment of a video. The structure of this neural network assumed 18 neurons and linear activation function in the hidden layer and a sigmoid activation function in the output layer. We used the Levenberg-Marquardt algorithm [27, 28] for training and the gradient of error function reached a minima after 10 iterations. Fast convergence indicates that the neural network was able to learn the classification task. We can assume this because fast convergence implies that the network was able to recognize quickly all the learning data not causing a divergence in the validation error at the same time. This is also a clue based on which we can say that the positive and negative training datasets are not overlapping each other. The accuracy reached around 65 % on average for all sports. More details on

how we split the dataset to train and test subsets is given in Sect. 14.5. We therefore experimented with a more sophisticated recursive network structure.

The natural choice was the Elman neural network [29], which assumes the recursion from a chosen number of neurons in the hidden layer. In this case we decided to consider two structures of the neural network:

- based only on nine values of descriptors at a certain moment of time;
- based on nine values of descriptors at a certain moment of time, values of descriptors at a moment preceding the considered moment of time and the information about whether the previous moment/frame was considered to be an event or not.

In the hidden layer we used 20 neurons where ten had a feedback to the input layer. Linear and sigmoid activation functions were used in hidden and output layer respectively. The Levenberg-Marquardt algorithm was used also for training, which was able to train the whole network in only 30 iterations. After a number of simulations, the architecture with 19 inputs appeared to be the most effective. The accuracy of this network was above 90 % for all sports in our dataset.

14.4 Scoreboard-Based Event Detection

The detection of an event in sport video broadcasts may seem to be an easy task. Very often there is a scoreboard in one of the corners on the screen to notify the viewers that an event occurred. So naturally a first approach would be to analyze this region of the screen to detect any interesting moments in a game. However, this is not an easy task from a computer vision point of view, taking into account the variety of shapes, fonts, and behaviors of these regions. For sports with relatively simple scoring systems, like soccer, Gaelic football, hurling, and rugby, the scoreboards tend to share some characteristics. There is usually information about the teams, points that they have scored, and the remaining/elapsed time. However, even in this case there is significant diversity in the possible layouts. Figure 14.3 shows some examples of the scoreboards in our dataset.

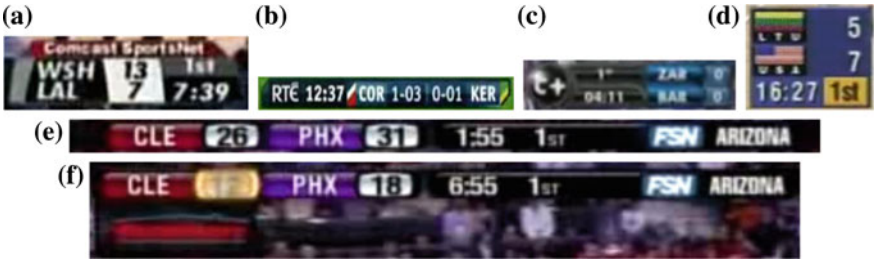

Fig. 14.3 Sample scoreboards: **a** *top-bottom* scoreboard with highlighted score region; **b** Gaelic football scoreboard with big/small points; **c** low bitrate soccer scoreboard; **d** Olympic basketball scoreboard; **e** side-by-side scoreboard with highlighted score regions; **f** the same scoreboard during a transition effect

These are just representatives of the vast range of scoreboards possible. Examples Fig. 14.3a shows the top-bottom layout of the scoreboard. Note, that the gap between the two scores and digits in the same score on the scoreboard Fig. 14.3a is very small, which significantly affects the digit segmentation/extraction process. Since the score region can be highlighted, the process that finds the digits needs to be invariant to different backgrounds and complex objects like country flags in the region where text is detected Fig. 14.3d. Scoreboard Fig. 14.3b represents a Gaelic football scoreboard where the score for both teams can be further split into big and small points. Another issue in text detection and recognition in the images is the quality of the image itself. Example Fig. 14.3c represents a scoreboard region extracted from the video of very low bitrate or resolution. Note, that even if the text region is correctly recognized the score itself is almost invisible, or at least very difficult to extract. These aspects that make the extraction of the score difficult are exacerbated by the fact that almost every broadcaster has implemented a different style of the scoreboard behavior that can be changed at short notice. There are two major features that can vary from broadcaster to broadcaster:

- the amount of time the scoreboard is visible on the screen (even when the score is recognized, due to the fact that in replays usually this information is invisible, it is not possible to simply assume that it will be there every time we need it);
- transition effects cause changes such as color change or some additional graphical effects Fig. 14.3e, f.

Clearly, the implementation of the scoreboard-based event detection system is not a straightforward task.

14.4.1 Scoreboard Detection

The scoreboard/text region detection process is run independently on each of the three channels of the RGB color space, and then merged into a single text region image representation in a logical *AND* manner presented with Eq. (14.3).

$$I^{\text{text}} = I^R \wedge I^G \wedge I^B \tag{14.3}$$

Note, that in order to perform logical *AND*, I^i where $i = R, G, B$ has to be a binary image.

Every channel is a logical *OR* combination of present and past video frames where text features are calculated with respect to the Eq. (14.4):

$$I^i = I^i_{t,t-\tau} \vee I^i_{t,t-2\tau} \vee I^i_{t-\tau,t-2\tau} \tag{14.4}$$

where $i = \{R, G, B\}$, t is an actual timestamp and τ is a fixed delay. In our experiments τ is equal to one second which is a tradeoff between the delay needed to detect a change by the algorithm and delay needed for the scoreboard to apply the

text transition effect. As it can be seen, we take three different pairs resulting from all different combinations of an actual frame and two frames from the past delayed by the same amount of the time between each of them. For every pair we calculate following formula (note, that for clarity we drop the i index) (14.5):

$$I_{t0,t1} = I^{\text{col}}_{t0,t1} I^{\text{var}} I^{\text{edg}}_{t0,t1} \tag{14.5}$$

Equation (14.5) shows the three features extracted for text region detection:

$$\begin{aligned}
I^{\text{col}}_{t0,t1} &= \overline{|I^{\text{col}}_{t0} - I^{\text{col}}_{t1}|} \\
I^{\text{var}} &= \frac{1}{L} \sum_{p=1}^{L} \left(p_{x,y} - \bar{I}_{x,y}\right)^2 \\
\bar{I}_{x,y} &= \frac{1}{L} \sum_{p=1}^{L} p_{x,y} \\
I^{\text{edg}}_{t0,t1} &= \overline{|I^{\text{edg}}_{t0} - I^{\text{edg}}_{t1}|} \\
I^{\text{edg}}_{t} &= \max\left(I^{\text{vertical}}_{t}, I^{\text{horizontal}}_{t}, I^{\text{diagonal}}_{t}\right)
\end{aligned}$$

where $p_{x,y}$ is a pixel of x, y coordinates, L is the number of pixels in the local variance mask and I^{vertical}_{t}, $I^{\text{horizontal}}_{t}$, I^{diagonal}_{t} are vertical, horizontal, and diagonal gradients of a frame t. These kind of gradients are proven to work well in text detection process [30].

Taking all the above into account a text region is a region where:

1. The color distribution does not change (even if it is transparent to some extent) and the font color does not change.
2. Text is a region of a high edge density that also usually does not change between the two frames.
3. Local variance of a text region is big.

This is always true for all three RGB channels. Figure 14.4 shows a sample soccer video frame and a response from text region identification algorithm. It can be seen that the presented algorithm works for multiple regions that have different characteristics (i.e., different colors, partially transparent background, etc.).

The next step after finding the text candidate regions is to create consistent text regions on which the digit identification algorithm can run. This is done by a project and split algorithm that works in the following manner:

1. Project the text candidate images horizontally onto the y axis (projection axis number 1 in Fig. 14.5);
2. Based on the projection curve detect consistent regions of values above a threshold, equal to 5 in our experiments—a good tradeoff between noise removal and small text region extraction (regions 1 and 2 in Fig. 14.5);

Fig. 14.4 Sample video frame from a basketball game (**a**) and its text region candidates (**b**)

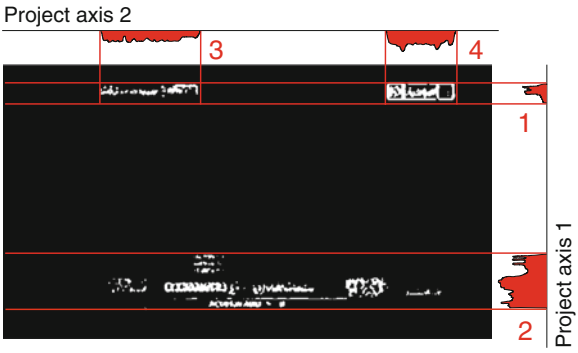

Fig. 14.5 Project and split algorithm

3. Store all the region candidates in memory;
4. Take region candidate from the queue;
5. Project region in the opposite direction (projection axis number 2 in Fig. 14.5);
6. Based on the projection curve detect consistent regions of values above threshold (regions 3 and 4 in Fig. 14.5);
7. If the detected region width/height is equal to the original width/height before projection store ready text region in the memory;
8. Go to step 4.

The above procedure can be implemented using recursion. The first few steps are shown in Fig. 14.5.

14.4.2 Event Detection

Figure 14.6 shows the processing flow in the scoreboard-based event detection task. The input to the algorithm are the extracted text regions described in the previous section. The following stages are described herein.

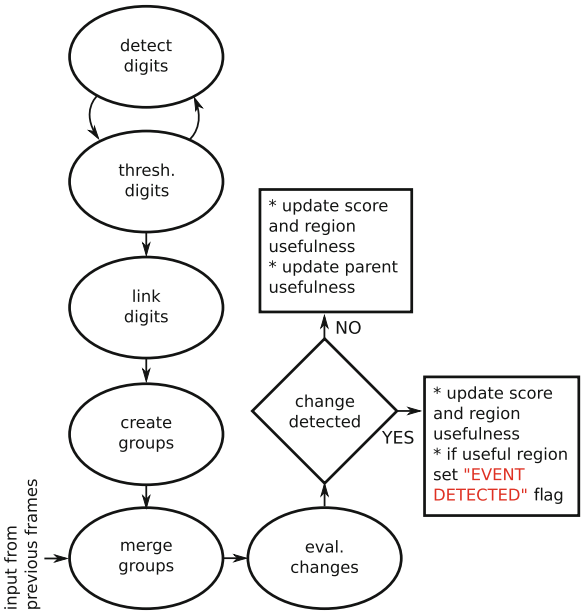

Fig. 14.6 The flow of scoreboard-based event detection

14.4.2.1 Digit Detection

Since the scoreboard region is typically designed for better visibility with high contrast we assume that either the text color is very dark or very bright. This approach allows us to deal with the challenge of different font colors (Fig. 14.3a). Based on this assumption, the detection phase is performed twice: the first time it seeks black digits on white background and second, white digits on black background. The digit identification algorithm finds foreground objects, extracts their features, and applies a cascade classifier to detect if the region contains a digit and, if so, what digit. The levels of the classifier validate the following features:

- region height;
- region width/height ratio;
- region no. foreground pixels to no. background pixels ratio;
- a Support Vector Machine (SVM) response.

An SVM is trained to recognize digit/non-digit image blocks based on concatenated horizontal and vertical projections of this block [31]. A feed-forward artificial neural network trained on the same input as in the SVM is then used to distinguish between digits. A dataset of figures containing differently shaped digits and non-digits was created to train both an SVM and ANN. Figure 14.7 shows some exemplary images from this dataset. It contains 5,000 images distributed equally between different digit and non-digit classes. We have chosen an ANN for

Fig. 14.7 Samples of digit images used to train SVM and ANN in digit detection and recognition tasks: **a** exemplary digits; **b** exemplary non-digits

multi-class digit recognition since thanks to its auto-associative characteristics it performed better than multi-class SVMs trained on the same dataset. Both classifiers obtained an accuracy of around 98 %.

14.4.2.2 Digit Thresholding

The digit identification algorithm works for every foreground object and is repeated for varying thresholds. This is due to the fact that the video compression process softens the edges, thus, making the digits' shapes less sharp, connected to other digits or the background. Introducing a loop where the input region is thresholded with increasing threshold value assures that in some range of this value the extracted digit will be recognized correctly. After a set of experiments we decided to implement ten thresholds with spacing equal to ten (0–100 for black digits, 150–250 for white). Based on experiments we performed, this is a more robust and efficient solution than thresholding with adaptive threshold value which is not very robust to quantization noise or performing image segmentation based on a Gaussian mixture model where the number of Gaussians is unknown. For every iteration of the loop the digit recognition result is remembered and a voting is performed afterwards so that the output of this algorithm is a digit with the highest number of votes.

14.4.2.3 Digit Linking and Digit Group Creation

The first of these two processes looks for digits that are in close proximity and links them together. The idea behind this process is to link the digits that comprise a single score or the time information in the scoreboard. Note, that the distance between two digits to be linked has to be adaptive and is a function of a region height and font characteristics. The second process takes all the links into consideration and performs digit grouping, i.e., a single group consists of the digits that were linked together. Note, that a particular digit in a digit group does not have links with all the members in the group (transitional links are allowed). Each group is assigned a specific number called the *usefulness* which stands for the number of digit changes within the group.

14.4.2.4 Digit Group Merging and Changes Evaluation

If at least one digit group was detected in the text region candidate the procedure assigns a scoreboard candidate flag to this region and tries to find any equivalent results (scoreboard candidates and digit groups) from the previously processed frames. The features taken into account are the dimensions, location, and number of digit groups in the scoreboard candidates. If the first two match then the scoreboard candidates are merged together. During the merge process the recognized digits and usefulness scores are updated. If there was a digit change this information is sent to the following step where changes evaluation takes place. During this process the usefulness of the scoreboard candidate and digit group where the change was detected are taken into account. First, since there was a change, digit group usefulness is decreased. If the usefulness of a digit group and its parent (scoreboard candidate) are sufficiently high an *event detected* flag is set. If not, the algorithm updates the usefulness of a scoreboard region.

Figure 14.8 shows an example scoreboard recognized by our approach. After the digits are detected (yellow bounding boxes) and grouped into digit groups (blue bounding boxes) a text region becomes a scoreboard region. This is visualized with a green bounding box. Scoreboard region and digit groups are given usefulness parameters visualized with numbers at the upper left corners. The pictured situation was captured after few seconds from the beginning of the video and it shows that the regions that change often (e.g., time region) are marked with negative usefulness whereas score digit groups and their parent (scoreboard region) have all high usefulness numbers.

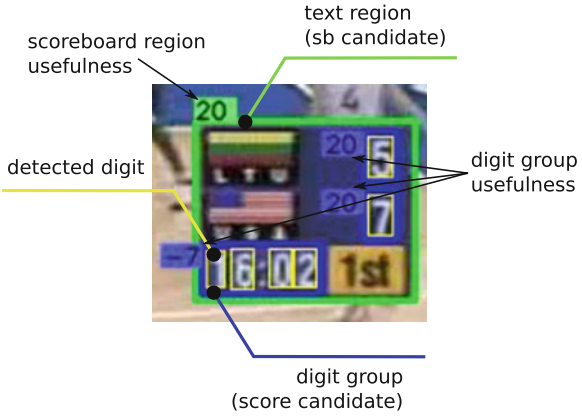

Fig. 14.8 Exemplary scoreboard region (*green* bounding box) with detected digits (*yellow* bounding boxes), digit groups (*blue* bounding boxes)

14.5 Event Detection Results

This section presents results obtained from testing our event detection algorithms on datasets proposed especially for this task. We discuss the accuracy and the time performance separately. Results obtained from both investigations allow us to conclude that the system is capable of extracting the interesting events in a real-time.

14.5.1 Accuracy

The dataset comprises 40 h of basketball, soccer, rugby, cricket, Gaelic football, and hurling videos. The duration of the videos in each sport group is balanced equally in order to avoid biasing the obtained results. Also accuracy, precision, and recall are calculated for each sport to show that exactness/quality and completeness/quantity of results are consistent. For every sport the ground-truth dataset is split into submodules in order to detect three types of events:

1. Goal event—an event that ends with a change of the game score;
2. Exciting event—an event that could be extremely exciting to the viewer, e.g., skillful dribble before the goal or a close-miss;
3. General event—type of an event that includes both previous events, plus additional, less exciting moments.

In Tables 14.3, 14.4 and 14.5 we present calculated values of accuracy, precision and recall for the three possible scenarios presented above. All the abbreviations used in the above tables are as follows:

SM—state machine;
DT—decision tree;
MLP—multi-layer perceptron neural network;
ENET—Elman neural network.

Table 14.3 Accuracy, precision, and recall calculated for six different games for the classification of the general event scenario

	Accuracy				Precision				Recall			
	SM	DT	MLP	ENET	SM	DT	MLP	ENET	SM	DT	MLP	ENET
Rugby	0.66	0.92	0.74	0.99	0.63	0.66	0.17	0.98	0.36	0.97	0.04	0.98
Soccer	0.58	0.95	0.79	0.98	0.7	0.77	0.21	0.95	0.51	0.97	0.04	0.96
Basketball	0.74	0.99	0.85	0.99	0.62	0.95	0.17	0.92	0.32	0.91	0.19	0.92
Cricket	0.42	0.95	0.9	0.99	0.7	0.67	0.1	0.93	0.26	0.73	0.04	0.97
Gaelic football	0.59	0.98	0.92	0.99	0.3	0.81	0.18	0.97	0.39	0.92	0.05	0.97
Hurling	0.44	0.95	0.8	0.99	0.79	0.77	0.36	0.99	0.73	0.98	0.08	0.99

Table 14.4 Accuracy, precision, and recall calculated for six different games for the classification of the goal scenario

	Accuracy				Precision				Recall			
	SM	DT	MLP	ENET	SM	DT	MLP	ENET	SM	DT	MLP	ENET
Rugby	0.55	0.96	0.94	0.99	0.22	0.61	0.01	0.88	0.4	0.73	0.00	0.99
Soccer	0.56	0.95	0.83	0.96	0.72	0.75	0.23	0.81	0.5	0.92	0.07	0.96
Basketball	0.53	0.99	0.67	0.99	0.62	0.95	0.02	0.82	0.47	0.98	0.22	0.93
Cricket	0.8	0.96	0.96	0.94	0.33	0.59	0.07	0.57	0.42	0.79	0.03	0.97
Gaelic football	0.78	0.98	0.85	0.98	0.57	0.77	0.09	0.71	0.6	0.73	0.1	0.97
Hurling	0.59	0.94	0.68	0.99	0.46	0.59	0.55	0.92	0.46	0.96	0.05	0.99

Table 14.5 Accuracy, precision, and recall calculated for six different games for the classification of the exciting event scenario

	Accuracy				Precision				Recall			
	SM	DT	MLP	ENET	SM	DT	MLP	ENET	SM	DT	MLP	ENET
Rugby	0.44	0.97	0.65	0.94	0.52	0.48	0.83	0.88	0.72	0.99	0.28	0.74
Soccer	0.73	0.97	0.82	0.95	0.48	0.75	0.27	0.82	0.51	0.99	0.53	0.8
Basketball	0.58	0.99	0.93	0.99	0.32	0.97	0.52	0.71	0.72	0.99	0.17	0.95
Cricket	0.41	0.95	0.27	0.96	0.41	0.34	0.73	0.78	0.36	0.99	0.19	0.72
Gaelic football	0.46	0.99	0.26	0.98	0.77	0.49	0.14	0.22	0.39	0.76	0.75	0.78
Hurling	0.74	0.99	0.57	0.99	0.6	0.52	0.48	0.7	0.4	0.93	0.24	0.78

The training was performed with use of probes collected only from one game (in this case—basketball) and simulated on six different games: rugby, football, basketball, cricket, Gaelic football, and hurling. In order to figure out the optimal split between training and testing subclasses of the given data we ran a number of experiments. Naturally, we tried a training dataset that is equally distributed among all classes and sports but this did not give the most effective results. Surprisingly the optimal division seems to be in the case when we take most of the samples from one game and train the classifier and test it on the remaining data points. This is due to the fact that a large training dataset from one sport only will cover most of the situations that the system could deal with and are simply impossible or at least very difficult to analyze by a human. Concluding, as the training data we used 70 % of probes collected from the basketball game, which is around 19,600 frames.

It can be seen from the results that the system's accuracy in the case of a decision tree classifier is usually above 95 % and never lower than 92 %. The results for the Elman neural network are between 96 and 94 %. This proves, that both classifiers are well trained and are capable of recognizing event sequences from videos or even sports that they have never seen before.

Regarding the efficiency of the scoreboard-based event detection (which can only detect scores), once the scoreboard is detected correctly, as was presented in Sect. 14.4.2 it is just a matter of tracking the changes inside this region. If the change

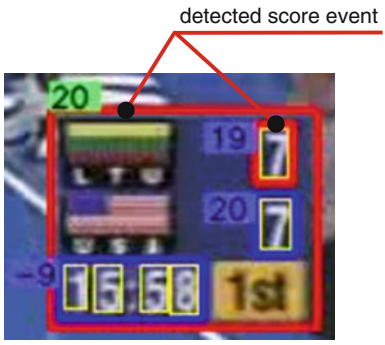

Fig. 14.9 Exemplary scoreboard region highlighting detected scoreboard-based event

appears in the digit region that does not change often and the digits present a number that is reasonably similar to the probable score of the game we say that there was a goal/event. Note, that the system is not constrained to count a score from zero since otherwise it would have to analyze every video from the beginning of the game. Figure 14.9 shows an exemplary visualization performed by the system when the change in the score is detected.

The only feature of the video that can affect this process is the perceptual quality of the scoreboard which is dependent on video bitrate and resolution. For example in our tests the system could easily recognize scoreboards in videos of resolution 640×480 whereas this was not the case for videos of resolution 768×76 due to a specific font used However, for videos where scoreboard regions can be clearly recognized by subjective viewing (about 80 % in our dataset) its effectiveness was always 100 %.

14.5.2 Time Performance

Given the classification methods presented in Sects. 14.3.3 and 14.4.2 it is important to highlight the time performance of the proposed detection techniques. Given the low accuracy of the state machine we do not take it into consideration.

As mentioned, the accuracy of the decision tree and Elman neural network are always higher than 92 % which makes them good candidates for the final event detector. Looking more closely at the two classification algorithms, the main differences that make them the best candidate for this task would be a train/predict time performance and the nature of the algorithm itself. The decision tree algorithm needs less than a second to create a tree (either in the case of nine or eighteen descriptors) while the Elman neural network needs more than a minute to perform the training. The prediction time is very similar for both cases: 0.0219 s for the decision tree and 0.0269 s for the neural network. Since training can be done offline it is not particularly crucial unless the system is required to change its behavior or gain some additional knowledge about events in interactive mode training time is very important. In addition,

the great disadvantage of the neural network is that as a non-deterministic algorithm it will always give different results for every training, while it always starts with different values of weights. Thus, it is very unpredictable and each has to be validated separately. Moreover, the output of the neural network is a value between 0 and 1, and setting up the threshold for event detection can be a difficult task.

Figure 14.10 shows the breakdown of the time performance of the feature-based event detection algorithm. This is the time needed to prepare data for the final classifier (decision tree or Elman neural network). As can be seen, keypoint detection, feature extraction and matching (i.e., assigning all the descriptors to particular nodes) consume the most time, whereas creation of bag-of-words histogram and SVM prediction can almost be disregarded.

The time performance of the scoreboard-based event detection significantly depends on the behavior of the scoreboard region. In general it can be split, according to Fig. 14.6 into the following stages: preprocessing, digit detection, digit linking, digit grouping, group merging, evaluation/event detection. All stages except the last one are done only a few times at the beginning of each game unless the behavior of the scoreboard prevents the algorithm from detecting a stable scoreboard region (e.g., when scoreboard appears only for a few seconds after the score changes). The times for particular stages are as follows:

- preprocessing—30 ms;
- digit detection—135 ms;
- digit linking—6 μs;
- digit grouping—30 μs;
- group merging—250 μs;
- evaluation/event detection—55 μs.

So after the scoreboard is detected and digit groups are recognized the scoreboard-based event detection algorithm only needs 55 μs to detect an event. Finally, it should be noted, that the given times for scoreboard-based and feature-based event detection are not cumulative since both techniques work in parallel with no interdependency.

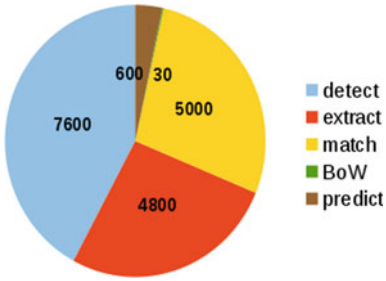

Fig. 14.10 Breakdown of processing time in the various stages of the feature-based event detection algorithm; all times are given in μs

14.6 Summary

This chapter describes a real-time field sports video event detection system that is capable of detecting goals/scores, close-misses, and also other exciting moments with high accuracy across multiple sports. Unlike previous work in sports scene classification, we pay particular attention to computational requirements, deriving an algorithm that is comparable with the state of the art in terms of accuracy, but which requires significantly less computational resources. For concreteness, our experimental evaluation focuses on field sports video as this covers multiple individual sports such as basketball, soccer, rugby, football, all genres of hockey, etc.; however, as our descriptor avoids using sports video specific mid-level features, like grass ratios, it is sufficiently general to be applicable in other contexts. The system is composed of two event detection modules that work in parallel performing complementary event detection tasks.

The first one is based on audio-visual low-level features and can be trained for different events such as goals, or very interesting moments such as a close miss. A feature of the approach is minimal ambiguity, as its response is considered in terms of event probability rather than a deterministic response. We also validated different event classification techniques and present results (accuracy, precision and recall) for each of them with respect to different sport videos processed by the system. The simplest approach is to use a state machine. However, the results showed that this solution does not give satisfactory results. To overcome this, we evaluated more sophisticated methods like deterministic decision trees or non-deterministic artificial neural networks. Both these solutions give good results, the difference is mainly in training time. It is important to notice that training either the decision tree or the Elman neural network based on the probes from one game only does not have negative influence on accuracy for any other game, which is always higher than 92 %. Using mixed values from different games could result in overfitting since it is very hard to judge if the game events encoded with low-level features are similar or not. For this reason we think it is better to use most of one game to train the event classifier since we can be sure it covers most of the possible cases on the field/court. The second, independent thread complements the first mentioned and is capable of detecting scoring events with 100 % accuracy with high resolution footage. The detection process is based on detection of changes in the score presented on the scoreboard on the screen. In this case, time performance is not constant during the video analysis and takes more time at the beginning where the scoreboard area has to be detected in order to track changes in the scores of the teams played.

References

1. Revoir P, The Most Watched TV Shows of All Time. http://www.dailymail.co.uk/tvshowbiz/article-1071394/The-watched-TV-shows-time-old-programmes.html
2. The 50 Most-watched Sporting Events of 2012. http://www.sportsmediawatch.com/2013/01/2012-numbers-game-the-most-watched-sporting-events-of-the-year/

3. Kokaram A, Rea N, Dahyot R, Tekalp AM, Bouthemy P, Gros P, Sezan I (2006) Browsing sports video. IEEE Signal Process Mag 23(2):47–58
4. Sadlier DA, O'Connor NE (2005) Event detection in field sports video using audio-visual features and a support vector machine. IEEE Trans Circuits Syst Video Technol 15:1225–1233
5. Li Baoxin S, Ibrahim M (2001) Event detection and summarization in sports video. In: Proceedings of the IEEE workshop on content-based access of image and video libraries (CBAIVL'01). IEEE Computer Society, Washington
6. Ye Q, Qingming H, Wen G, Shuqiang J (2005) Exciting event detection in broadcast Soccer video with mid-level description and incremental learning. In: Proceedings of the 13th annual ACM international conference on multimedia. Hilton, Singapore, pp 455–458. doi:10.1145/1101149.1101250
7. Chen SC, Shyu ML, Zhang C, Luo L, Chen M (2003) Detection of soccer goal shots using joint multimedia features and classification rules. In: Proceedings of the fourth international workshop on multimedia data mining (MDM/KDD2003), pp 36–44
8. Kim HG, Roeber S, Samour A, Sikora T (2005) Detection of goal events in Soccer videos. In: Proceedings of storage and retrieval methods and applications for multimedia, vol 5682
9. Wang L, Liu X, Lin S, Xu G, Shum H (2004) Generic slow-motion replay detection in sports video. In: ICIP, pp 1585–1588
10. Jinjun W, Engsiong C, Changsheng X (2005) Soccer replay detection using scene transition structure analysis. In: Acoustics, speech, and signal processing, proceedings (ICASSP'05)
11. Zhao Z, Shuqiang J, Qingming H, Guangyu Z (2006) Highlight summarization in sports video based on replay detection. In: 2006 IEEE international conference on multimedia and expo, pp 1613–1616. doi:10.1109/ICME.2006.262855
12. Lanagan J, Smeaton FA (2011) Using Twitter to detect and tag important events in live sports. In: Proceedings of the fifth international AAAI conference on weblogs and social media, pp 542–545
13. Changsheng X, Zhang Y, Guangyu Z, Rui Y, Hanqing L, Qingming H (2008) Using webcast text for semantic event detection in broadcast sports video. IEEE Trans Multimed 10(7):1342–1355. doi:10.1109/TMM.2008.2004912
14. Kapela R, McGuinness K, O'Connor NE, Real-time field sports scene classification using colour and frequency space decompositions. J Real-Time Image Process (In review)
15. Lowe DG (1999) Object recognition from local scale-invariant features. In: IEEE international conference on computer vision, pp 1150–1157
16. Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. In: Proceedings of the 2005 IEEE conference on computer vision and pattern recognition (CVPR 2005), vol 1, pp 886–893
17. Calonder M, Lepetit V, Strecha C, Fua P (2010) BRIEF: binary Robust independent elementary features. Lecture Notes in Computer Science, pp 778–792
18. Alahi A, Ortiz R, Vandergheynst P (2012) In: IEEE conference on computer vision and pattern recognition, Rhode Island, Providence, USA, 16–21 June
19. Leutenegger S, Chli M, Siegwart RY (2011) BRISK: binary robust invariant scalable keypoints. In: 2011 IEEE international conference on Computer vision (ICCV), pp 2548–2555. doi:10.1109/ICCV.2011.6126542
20. Tola E, Lepetit V, Fua P (2010) An efficient dense descriptor applied to wide baseline stereo. IEEE Trans Pattern Anal Mach Intell 32(5):815–830
21. Mikolajczyk K, Schmid C (2005) A performance evaluation of local descriptors. Pattern Anal Mach Intell IEEE Trans 27(10):1615–1630. doi:10.1109/TPAMI.2005.188
22. Khvedchenia I, A battle of three descriptors: SURF, FREAK and BRISK. http://computer-vision-talks.com/2012/08/a-battle-of-three-descriptors-surf-freak-and-brisk/
23. Perronnin F, Dance C (2007) Fisher Kernels on visual vocabularies for image categorization. In: Proceedings of the IEEE conference on computer vision and pattern recognition, vol 1–8
24. Hervè J, Douze M, Schmid C, Perez P (2010) Aggregating local descriptors into a compact image representation. In: IEEE conference on computer vision and pattern recognition, pp 3304–3311

25. Ordonez C (2003) Clustering binary data streams with K-means. In: Proceedings of the 8th ACM SIGMOD workshop on research issues in data mining and knowledge discovery DMKD'03, pp 12–19
26. Coppersmith D, Hong SJ, Hosking JRM (1999) Partitioning nominal attributes in decision trees. Data Min Knowl Discov 3:197–217
27. Levenberg K (1944) A method for the solution of certain problems in least squares. Q Appl Math 5:164–168
28. Marquardt D (1963) An algorithm for least-squares estimation of nonlinear parameters. SIAM J Appl Math 11(2):431–441
29. Hammer B (1997) Generalization of Elman networks. In: Artificial neural networks—ICANN'97, pp 409–414
30. Shivakumara P, Weihua H, Tan CL (2008) Efficient video text detection using edge features. In: 19th International conference on pattern recognition, 2008. ICPR 2008. December 2008, vol 1, no 4, pp 8–11
31. Baggio DL, Emami S, Escriva DM, Khvedchenia I, Mahmood N, Saragih J, Shilkrot R (2012) Mastering OpenCV with practical computer vision projects, Pact Publishing
32. Binary decision tree for classification, MathWorks Documentation. http://www.mathworks.com/help/stats/classificationtreeclass.html