OPTIMIZATION OF AUTOMATIC HIGHLIGHT DETECTION FOR FOOTBALL BROADCASTS

Thiago Frensch

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientadores: Sergio Lima Netto
Eduardo Antônio Barros Da Silva

Rio de Janeiro
Março de 2015

OPTIMIZATION OF AUTOMATIC HIGHLIGHT DETECTION FOR FOOTBALL BROADCASTS

Thiago Frensch

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

Prof. Sergio Lima Netto, Ph.D.

Prof. Wallace Alves Martins, D.Sc.

Prof. Lisandro Lovisolo, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2015

Frensch, Thiago

Optimization of automatic highlight detection for football broadcasts/Thiago Frensch. – Rio de Janeiro: UFRJ/COPPE, 2015.

XII, 60 p.: il.; 29, 7cm.

Orientadores: Sergio Lima Netto

Eduardo Antônio Barros Da Silva

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2015.

Referências Bibliográficas: p. 57 – 60.

1. Feature Selection. 2. Soccer. 3. Correlation. 4. Classifier. 5. Support Vector Machine. 6. AdaBoost. 7. Random Forest. I. Netto, Sergio Lima *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

Aos meus pais e esposa.

Agradecimentos

Gostaria muito de agradecer:

- aos professores Sergio Lima Netto e Eduardo Antônio Barros da Silva, pela oportunidade, orientação e paciência dada durante todo o período do mestrado;
- aos professores Wallace Alves Martins e Lisandro Lovisolo, por aceitarem o convite de participação na banca de examinação deste trabalho;
- ao Luiz Gabriel de Vasconcelos pela oportunidade e apoio necessário para o desenvolvimento desta dissertação
- à minha esposa, pelo seu amor, apoio e compreensão nos momentos de ausência;
- aos meus pais, irmã e toda família, pelo apoio incondicional;
- aos meus colegas de turma, pela amizade e suporte durante esses anos de mestrado;
- à Central Globo de Engenharia da TV Globo, pelo material fornecido necessário ao desenvolvimento deste projeto;
- a todos funcionários e professores do Programa de Engenharia Elétrica da COPPE, por todos os serviços prestados.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

OTIMIZAÇÃO DE DETECÇÃO AUTOMÁTICA DE MELHORES MOMENTOS DE TRANSMISSÕES TELEVISIVAS DE FUTEBOL

Thiago Frensch

Março/2015

Orientadores: Sergio Lima Netto
Eduardo Antônio Barros Da Silva

Programa: Engenharia Elétrica

Todo dia mais eventos de esportes são televisionados por todo o mundo, porém cada emissora não possui tempo hábil para a exibição de todo esse conteúdo. Contudo, as emissoras não podem perder essas informações, tudo de importante deve ser armazenado para futuras consultas e também deve ser resumido para utilizações compactas. A mão de obra necessária para seleção manual seria muito custosa e demorada. Para resolver este dilema, é necessária uma ferramenta confiável capaz de executar a seleção de forma eficiente. Uma ferramenta dessas foi desenvolvida em um trabalho anterior, tendo obtido bons resultados. Entretanto, a sua complexidade computacional e eficiência ainda deixavam a desejar. O objetivo deste trabalho é aprimorar essa ferramenta de modo que ela se torne menos complexa e mais eficiente sem comprometer seu desempenho. Isso foi obtido empregando-se inicialmente técnicas de seleção de características com o intuito de reduzir a massa de dados necessária inicial para o processo. Em seguida, através de experimentos com novos classificadores, se buscou aumentar o seu desempenho em termo de taxas de erro de classificação. Foi obtido uma redução de pelo menos 4 vezes do número de entradas usadas pelos classificadores, sem nenhuma perda de desempenho. No caso de se tolerar uma pequena redução de desempenho, alcançou-se uma redução de até 40 vezes no número de entradas originais.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

OPTIMIZATION OF AUTOMATIC HIGHLIGHT DETECTION FOR FOOTBALL BROADCASTS

Thiago Frensch

March/2015

Advisors: Sergio Lima Netto
Eduardo Antônio Barros Da Silva

Department: Electrical Engineering

Each day more sports events are broadcasted around the world, but each television broadcaster does not have enough time for the exhibition of all the content. However, TV broadcasters can not miss such information, every important content must be stored for further queries, and also they must be summarized for compact exhibition. The manpower necessary for manual selection would be too expensive, besides slow. To solve this dilemma, a reliable tool capable of executing this selection in an efficient way is necessary. One such tool has been developed in a previous work, and good results have been obtained. However, its computational complexity and efficiency are still not ideal. The objective of this work was to improve this tool in order to it become less complex and more efficient without compromising its performance. This was achieved initially by employing techniques of feature selection with the aim of reducing the amount of initial data required for the process. Then, by making experiments with different classifiers, we tried to increase the performance in terms of classification error rates. We have obtained a four-fold reduction in the number of inputs used by the classifiers with no loss of performance. In case a small reduction of performance is allowed, we have reached a reduction in the number of inputs of up to 40 times.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Most TV broadcasters have their own video logging system. In general, it consists of collecting every important event which occured in videos owned by the broadcasters is discriminated by defining its content and time (beginning and end) stamps. Such a system is a very powerful tool for which reporters and screenwriters can make queries, in case they need to find any specific video content, such as a desired event or person.

There are many ways to insert a video description into the logging system. The most common method is through a direct human interface, where someone sits in front of the video screen and watches the target video. Every time that person finds out an important moment, he/she inserts into the computer the label (some sort of brief description), the mark in and the mark out of the event in that particular stream. In this process, logging a given video excerpt takes at least the duration of the video, requires at least one professional, and carries a great deal of subjectivity.

During large sport events, for instance, such as the FIFA World Cup and Olympic Games, where many events of interest occur simultaneously, logging costs may increase substantially by employing more people to expedite the process. An alternative is to record all videos and have the logging process performed by fewer people in non-real time. For journalistic purposes, however, this cost-efficient solution may not be viable, as the broadcaster cannot waste any time to transmit news related to that event.

The logging process is still performed manually requiring full attention from the operator to avoid missing an important event. There are many studies, however, which attempt to perform the logging task automatically, with the assistance of a computer. In that case the main challenge lies in making the computers understand and evaluate the event importance similarly to a human being. Some recent studies have succeeded in finding important events in particular types of TV broadcasts, such as specific sport events, but still with a considerable error: either missing important events (false negative situation) or classifying a dull event as an important

one (false positive situation). The focus of this work is the automatic summarization of football transmissions, using both audio and video information, in an attempt to minimize both error situations.

1.1 Automatic Summarization

For humans, the task of identifying highlights in a football game is simple. If you are watching the game, you can easily tell which event you would call important or not. However, if you try to describe the fact that made that event important, you would probably say something like: the player shot at the goal. You will identify the player, the ball and the interaction between both to detect the event. The same happens to other events like goals, fouls and etc. The problem is that computers do not identify such entities easily, like we do. Thus, the simple task of detecting the game highlights requires performing several pre-detection tasks, which often require additional procedures, and so on, until we reach the task level that computers can do easily, like addition and multiplication operations.

There are many ways to teach a computer how to detect highlights, but every method has its advantages and also disadvantages. The method described above is the object tracking method. which is the easiest way to be understood. However, when we try to understand how to make computers working the same way, that method becomes one of the most complex methods to reach the desired target. In practice, when using computers to perform a human task, it is often easier to have the programmer to think like a computer (and take advantage from the computer's main capabilities) than to have the computer to think like a human being.

Figure 1.1 represents most of the approaches used for automatic highlight detection in some sport events [2]. Following that framework, most methods can be classified according to three different aspects, as discussed below.

The dependence on the video models can be divided into genre-specific and genre-independent. To be genre-specific means that the work is fully oriented to a specific sport, i.e., the algorithm can only be applied to football if it was designed for football. This approach normally leads to better results but has an inherent loss of generality. The genre-independent is a class of approaches which tries to keep the algorithm generic for most sport events.

With respect to the data dependence, there are also two separate groups: the external-source-based and the internal-source-based algorithms. In the external-source-based group, the work uses information from outside the original broadcasted signal, such as closed caption and web-casting text. The internal-source model extracts all required information from the standard transmitted signal. In this case, the three information sources, that is, video, audio, and textual data can be used

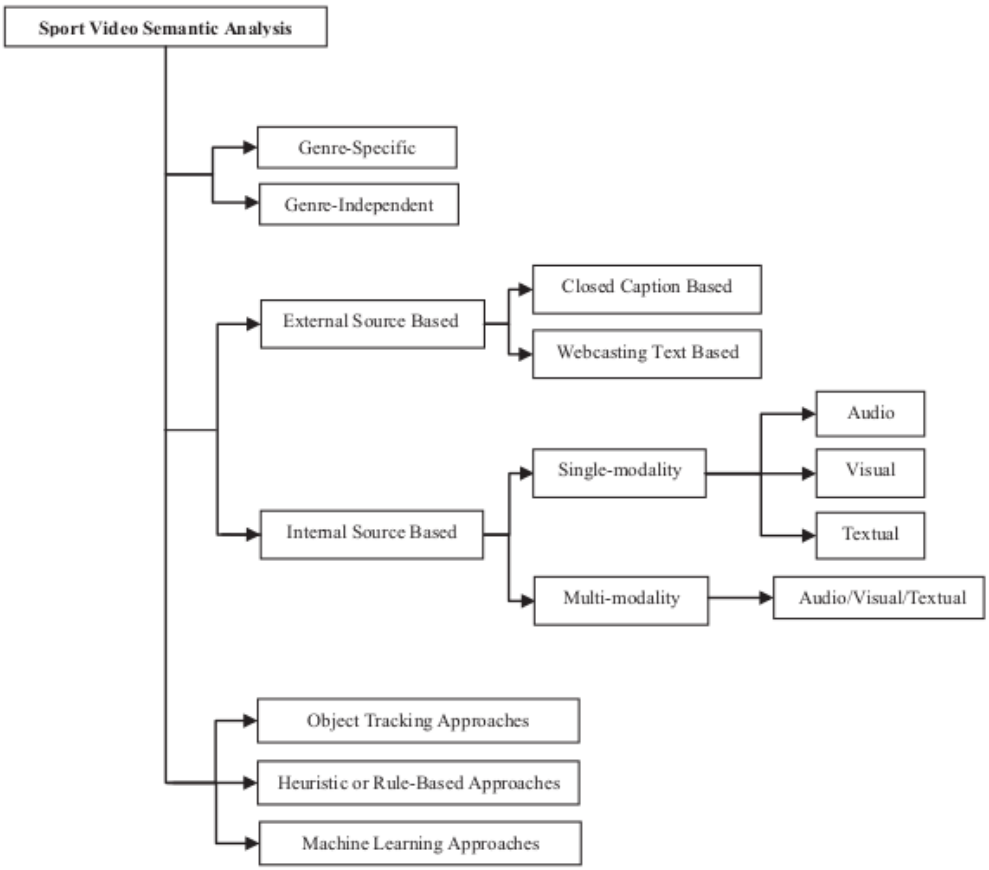


Figure 1.1: A taxonomy of different approaches proposed for sport video semantic analysis [2].

separately or in a multi-modal approach.

The third and final aspect is the algorithm pattern. The object-tracking technique, as mentioned before, is the most complex method, as it tries to identify and track every significant object in the video. The heuristic or rule-based approaches use rules which are created based on the observation of some event patterns in the video. For example, if a logo is displayed on the screen, an important event must have happened in the game. The last method is the machine-learning approach where a classifier is selected and trained using single- or multi-modality features. In such method, the classifier training creates some intrinsic model of the highlights, which is then used to identify the events of interest in an automatic manner.

The work presented in this dissertation is genre specific, as we focus on football summarization. By analyzing the data dependence, the proposed work may be classified as multi-modal internal-source based, as it is based on features extracted from the available audio and video signals. Also we follow the machine-learning approach, and investigate the use of different classifiers such as random forest and SVM.

This work achieved large reduction of the features used in the input of the classifier without any loss of precision. The initial inputs in the classifier were 976 input features, as obtained in [1]. After the feature selection the number decreased to 488, and later with the spacing between frames the number decreased to 248 without any loss of precision and a big increase of performance. Later, experimenting new classifiers, the Random Forest was discovered as a strong classifier this job. It achieved similar results to the ones obtained by [1] using AdaBoost.

1.2 Organization of the Dissertation

The organization of this master thesis as follows:

- The bibliographic review (Chapter 2) about the newest papers in the area, and also the new method, techniques and directions of the works of the area;
- The previous work(Chapter 3) which this work is taking as a start. Describing the techniques used by [1] and the results obtained by it.
- The feature selection (Chapter 4) applied over the [1]. Using correlation techniques to excluded unnecessary information used in the classifier. Improving the performance of the whole process.
- The classifiers (Chapter 5) search for some of the most relevant classifiers in the state of art. The background theory of each one of the selected classifiers

and also the main characteristics for the scenario which they are going to be applied.

- The experiments (Chapter 6) is where the empirical study is done. This Chapter is very important to conclude the affirmations claimed before in the work. And also to test new possible ways to direct the studies of this and later works.
- The conclusion (Chapter 7) is the Chapter where the whole conclusion of the project is combined, so the final conclusion can be done.

Chapter 2

Bibliographic Review

This Chapter discusses different interpretations of the summarization problem by several authors and compare their approaches and techniques for solving that problem. The main differences described in this review are:

- The type of events of the game to be selected by the algorithm;
- Different types of algorithms;
- The low-level, mid-level, and high-level features;
- Feature processing;
- Segmentation of the game;
- Most common features;
- Highlighting and semantic detection;
- Most common classifiers.

The associated literature has many different approaches for the summarization of sport videos and also different understandings. Some works like [4] summarize the video by looking for goals within the football game, whereas works such as [5] and [6] go further and classify events like yellow and red cards, corner kicks, free-kicks and a lot of other events.

The type of event summarized depends on the focus of the summarization. Some applications, such as the complete logging process, need the full semantic meaning of the event, while others only require the highlight event itself with no semantic meaning associated.

Also, summarization approaches can vary from using only the camera shot type for classifying the event, such as in [7], to doing a complete object tracking as proposed in [8]. The camera-shot classification although simple is very limited to

the producer method of operation, whereas the object-tracking method elevates the complexity of the classification as it attempts to catch the full meaning of the event like humans do.

Besides those techniques, other works such as [9] and [10] use a mid-level layer of information processing. The mid-level layer is an intermediate processing step, which attempts to achieve a good compromise between system complexity and generality. In that scheme, the process transforms the raw signal features into new variables that carry more semantics to the overall final context, splitting the classification job in, at least, two steps. For example, [6] uses line detection to detect straight lines in the field as low-level features. The identified lines are then processed to determine if the scene contains or not the penalty box (which, can be identified by some parallel lines close to each other). In this case, the penalty-box detection becomes the mid-level feature, which is nearer to a semantic event in a football game, as it allows one to deduce if the attacking team is close to the opponent's goal. The mid-level technique is also an important tool for reducing the number of system inputs, what sometimes hinders the use of some classifier families such as support vector machines (SVM).

Table 2.1 shows some features used in some of the main articles in the area. The most common low-level feature used in the summarization literature is the frame dominant color. There are many ways to extract this feature, like getting it from the hue histogram along the frame, as used in [2], [11], [12], or [13], which also cuts off the superior part of the frame before getting the dominant color. The motion vector is the second most widely used low-level feature [14], [15]. In [16], the motion vector is extracted directly from the coded stream as given by the H.264/MPEG-4 codecs or by calculating the discrete cosine transform (DCT) between consecutive frames. Both of these low-level features are used to get the camera-shot type, which is the most common used by mid-level feature. This feature tries to describe the camera view type of a given frame, like global panoramic, close up, and zoom shots. This mid-level classification employs the dominant-color percentage, the object sizes and the average pixel motion in a given frame to decide the camera-view type. A Table with the main mid-level features used in some articles are described in Table 2.2.

The play-break technique is also widely used in video segmentation works, such as [6], [13], and [14]. This technique, as studied in [18], uses the fact that every important event is followed by a break, which should be considered by the viewer to provide a complete understanding of the event itself. For example, the yellow and red cards in football games only happen during a break, after some foul play. This technique uses the dominant color as main low-level feature but can also use replay, slow-motion, and logo detection to improve the video segmentation, at the cost of increasing dependency on the video production framework. Video segmentation

Table 2.1: Summary of video low-level features employed in the summarization literature.

	[5]	[6]	[7]	[9]	[13]	[11]	[12]	[2]	[17]	[1]
Dominant color	X	X		X	X	X		X	X	X
Motion Vector	X	X		X		X	X			X
Player Size Detection		X								
Count White Pixels		X								
Line Detection	X	X		X				X	X	
Face Color	X			X	X			X		
Color Histogram				X					X	
Region Entropy				X						
Jersey Color								X		
Non Zero DCT for Crowd	X									
Shot Cut			X							

Table 2.2: Summary of video mid-level features employed in the summarization literature.

	[5]	[6]	[7]	[9]	[13]	[11]	[12]	[2]	[17]	[1]
Camera Shot type	X	X	X	X	X	X	X	X	X	
Penalty Box		X								
Break Duration		X								
Replay Duration		X								
Number of Close-Up		X								
Shots in Break		X								
Nb of Players in Penalty Box		X								
Players Detection					X					
Referee Detection		X							X	
Goal-Keeper		X								
Graphical Captions	X	X								
Slow Motion Detection					X				X	
Logo Detection		X			X			X		
Shot Boundary Detection	X								X	

helps separate the candidates for highlight events from the normal play, greatly reducing the number of times a classifier is activated.

Most of the articles in this area do the main work using only video features, completely disregarding the audio or other auxiliary signals. In fact, the related literature indicates that audio features do not give a semantic meaning to the event unless keywords are detected, which is a very computationally intensive process. However, the audio signal may be quite efficient in separating football highlights from the normal play, particularly for typical Brazilian broadcasts. Table 2.3 shows some of the main audio features used in the highlight identification process.

Currently, most works focus on the semantic of the event applied after a simple highlight or non-highlight selection. Works like [6] and [17], for instance, make logo, replay, or slow motion detection to filter the non-highlight events, which is a very

Table 2.3: Summary of audio low-level features employed in the summarization literature.

	[5]	[6]	[7]	[9]	[13]	[11]	[12]	[2]	[17]	this
Short Time Energy	X		X					X		X
Comb Sum Energy				X			X			X
ZCR				X			X			
STE							X			
Pitch	X						X			X
Brightness							X			
Shots in Break							X			
Bandwidth							X			
Spectrum flux				X			X			
Sub-band power				X			X			
LPCC							X			
MFCC				X			X			

poor approach considering that not all video productions follow the same sequence of scenes. For example, the Brazilian championship broadcast never uses logos after a highlight or before a replay and also almost never uses slow-motion takes. Therefore, those algorithms would not be effective in the Brazilian football broadcasts, which require a more sophisticated highlight detection.

Another important characteristic of a given highlight-detection system is the type of classifier employed in the system. Table 2.4 shows the most frequently employed classifier families in the related literature. In general, these classifiers differ in the corresponding computational complexity, capability of generalization (avoiding overtraining), number of input features, and misclassification rates.

Table 2.4: Summary of classifier families employed in the summarization literature.

	[5]	[6]	[7]	[9]	[13]	[11]	[12]	[2]	[17]	this
Support Machine Vector	X	X				X				
Hidden Markov model		X	X	X						
Bayesian Network		X				X				
Neural Network								X	X	
Unsupervised Clustering							X			
Ranked Classification							X			
Rule-Based					X					
AdaBoost										X

2.1 The [1] Approach

In the work, a genre-specific approach was considered, suited to highlight detection in football broadcasts. Some similarities of football with other sports (grass play

field, such as American football and rugby; goal oriented, as indoor football and handball; and so on) may make, however, the final system suitable to other sports as well.

It uses joint audio and video features, leading to a multi-modal approach. Overall, were verified that the audio features are the main features in the highlight classification, whereas the video features provide a slight increase in system accuracy and, most importantly, greater independence from the game narrator.

The main information to be extracted from the audio features was the narrator excitement, which can be detected mostly from the associated pitch and energy. In practice, one can say that the narrator uses these features to indicate the best moments of the game for the spectators. This was the same behavior wanted for the classifier using the narrator's voice features.

The video features were used to generalize the classifier performance to another narrator's voice or style. The dominant color, camera movement, and panoramic frames are going to be the video features considered here. The video features also help discard events that happen outside the playing field, like the ones associated to the fans. It can also be used to remove replays, logos, and closed-up takes.

One of the main contributions of the present dissertation is the analysis of importance of all audio and video features considered in this work. Such a survey is needed to reduce the number of classifier input parameters, which on its turn may lead to a faster feature-extraction stage, simpler training procedures, and more reliable classification rates (as noisy information is removed a prior).

Other contribution is the performance analysis of several classifier families on the problem at hand. In particular, in this dissertation, we consider the use of neural networks, AdaBoost, SVM, and random forest classifiers, investigating the advantages and disadvantages of each classifier type in several aspects.

Chapter 3

Previous Work

3.1 Introduction

This dissertation is closely related to reference [1]. To facilitate comparisons between both works, this Chapter summarizes the main aspects of [1], detailing its signal features, classifier, and database, as well as the results achieved by the corresponding system.

3.2 The features

3.2.1 Audio Features

Following [19], [20], [21], [22], and [23], one may conclude that the audio features such as pitch (fundamental frequency, for all practical purposes) and energy contribute significantly for the highlight detection in football games, as, in general, the narrator's voice tends to reflect the excitement of the game.

The pitch estimation was the first feature employed by [1]. Given the audio signal $x(n)$, the pitch estimation can be performed using the autocorrelation function as given by [24]:

$$R_{xx}(\tau) = \sum_{n=1}^{N} x(n)x(n-\tau) \quad , \tag{3.1}$$

where N indicates the number of signal samples considered in this process. For stationarity purposes, audio segments are usually in the range of 20 to 30 ms. In this work, we have adopted 33.3 ms duration to match the duration of an NTSC frame, since this is a multi-modal system. An example of the autocorrelation function of an audio signal is depicted in Figure 3.1, where it is possible to extract the period T, and consequently, its inverse, which is the desired fundamental frequency F_0 of the audio signal (Pitch).

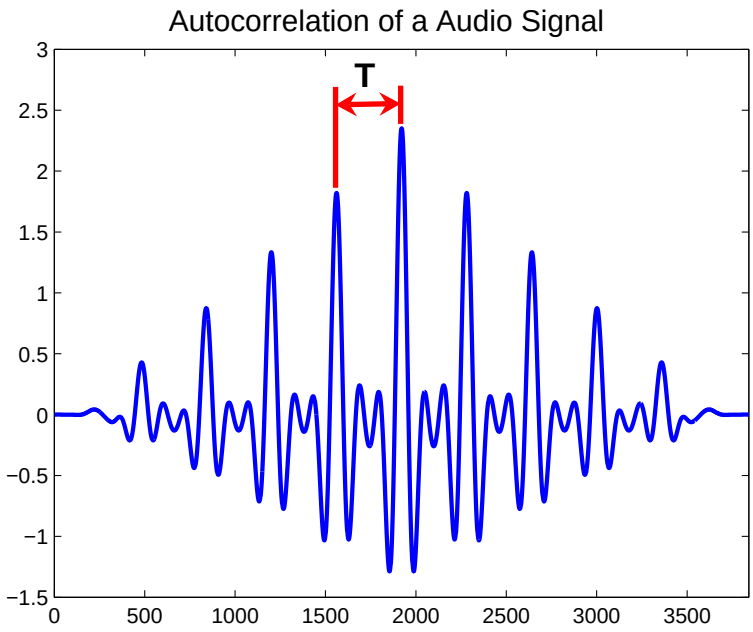


Figure 3.1: Example of fundamental period T of a given audio signal extracted from the corresponding autocorrelation function. (Source: [1])

An alternative pitch estimation method, which is less computationally intensive, uses the frequency domain to estimate the autocorrelation function, as given by [25]

$$R_{xx}(\tau) = IDFT\{|DFT[x(n)]|^2\} \quad , \tag{3.2}$$

where $DFT(\cdot)$ and $IDFT(\cdot)$ are the direct and inverse discrete Fourier transforms.

In [1], the method for calculating the pitch are similar to [3] but with two slight differences: following [26], pitch values lower than 50 Hz and above 500 Hz are discarded, the same happens for audio segments with low energy values. Such modifications lead to the differences between the performances of the two methods illustrated in Figure 3.2.

Another audio feature selected in [1] is the energy of the voice within a given segment, which can be determined as

$$e_{st}(t) = \sum_{n=t-\frac{N}{2}}^{t+\frac{N}{2}} x^2(n)h(n) \quad , \tag{3.3}$$

where $h(n)$ denotes the window function applied to the audio signal, which in this case was of the *Hamming* type.

To emphasize the fundamental frequency of a given audio signal and its corresponding harmonics, we may put the signal through a comb filter with frequency

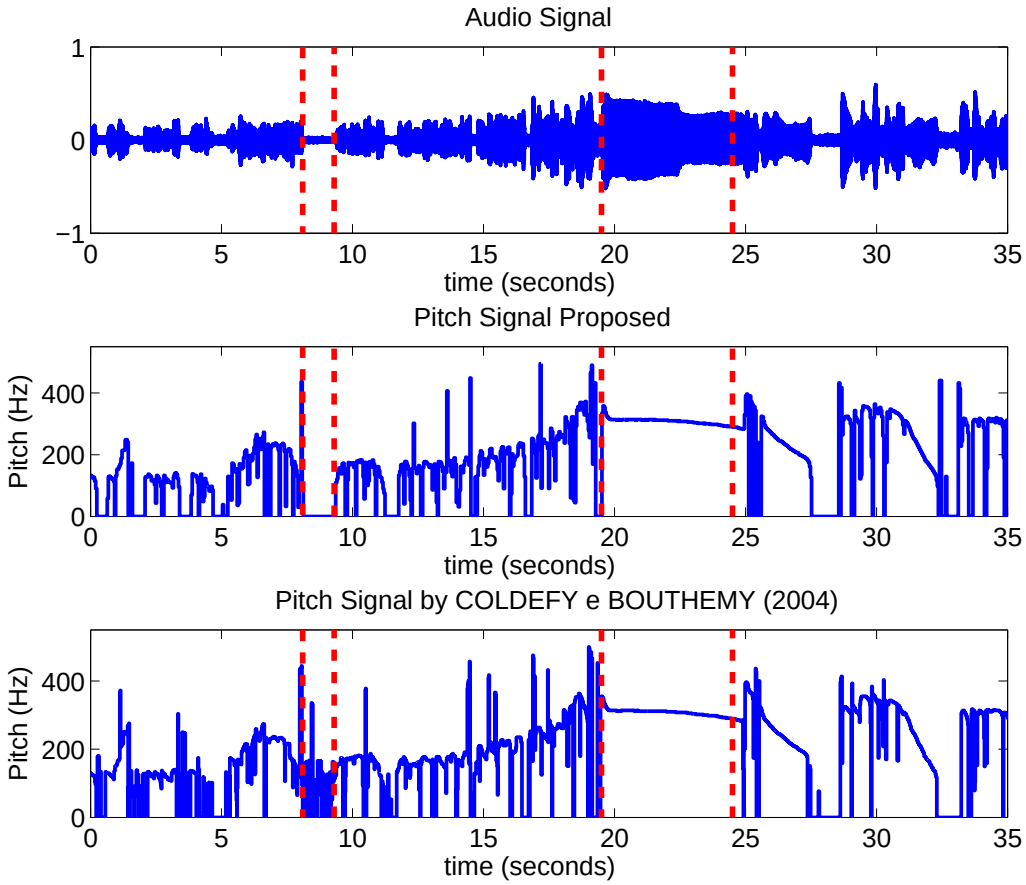


Figure 3.2: Pitch comparison between the methods of [1] and [3]. (Source: [1])

response of the form

$$C(f) = \sum_{k=1}^{d} \delta(f - kF_0) * \Pi(f) \quad , \tag{3.4}$$

where δ is the Dirac's impulse function, d is the number of harmonics which will have their energy emphasized, and Π is the window function associated to the comb filter. Figure 3.3 illustrates the role of the comb filter in emphasizing the voiced information of the given audio signal.

Another interesting audio feature attempts to evaluate the signal's dynamic evolution along time. The idea is to determine the moments where the energy signal increases significantly indicating a possible highlight of interest. This can be done by using a differentiator followed by a low-pass filter to smooth out the differentiator's output. Another possibility is to evaluate the local growth by using the auxiliary functions

$$\mu_a(t) = \sum_{n=t+1}^{t+N_a} f(n), \tag{3.5}$$

$$\mu_p(t) = \sum_{n=t-N_p}^{t-1} f(n), \tag{3.6}$$

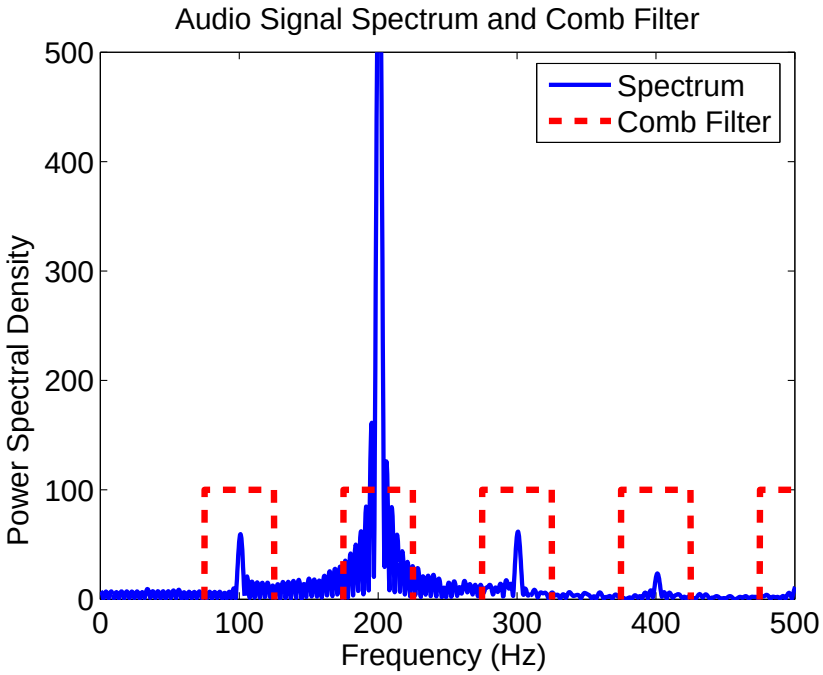


Figure 3.3: Use of a comb filter tuned to the signal pitch and its harmonics to estimate the audio signal energy. (Source: [1])

and then calculate the so-called local-growth estimate given by

$$\mu_s(t) = \begin{cases} 1, & \text{if } \mu_p(t) - \mu_a(t) \geq 0 \\ 0, & \text{if } \mu_p(t) - \mu_a(t) < 0 \end{cases}. \tag{3.7}$$

3.2.2 Video Features

The first video feature used in [1] is the dominant color of the frame. This feature indicates when the camera focuses the playing field (which should be dominantly green) or it has a panoramic take (close-ups may have other dominant colors), as illustrated in Figure 3.4.

When showing the field, a large proportion of the frame tends to be of some shade of green. In order to avoid dependency on illumination intensity, the dominant color algorithm should operate in a color space such as the hue-saturation-intensity (HSI) system, as opposed to the standard red-green-blue (RGB) domain. In the HSI system, the first component corresponds to the traditional concept of color [27]. Figure 3.5 shows the representations of those HSI and RGB systems, where one concludes how the HSI controls the shades of green with a single parameter, different of RGB system which the color depends on all three parameters.

Usually, the video signal is encoded in the RGB space. In order to transform the RGB domain into the HSI system, one must employ the following relationships:

Figure 3.4: Examples of video frames for dominant color computation. (Source: [1])

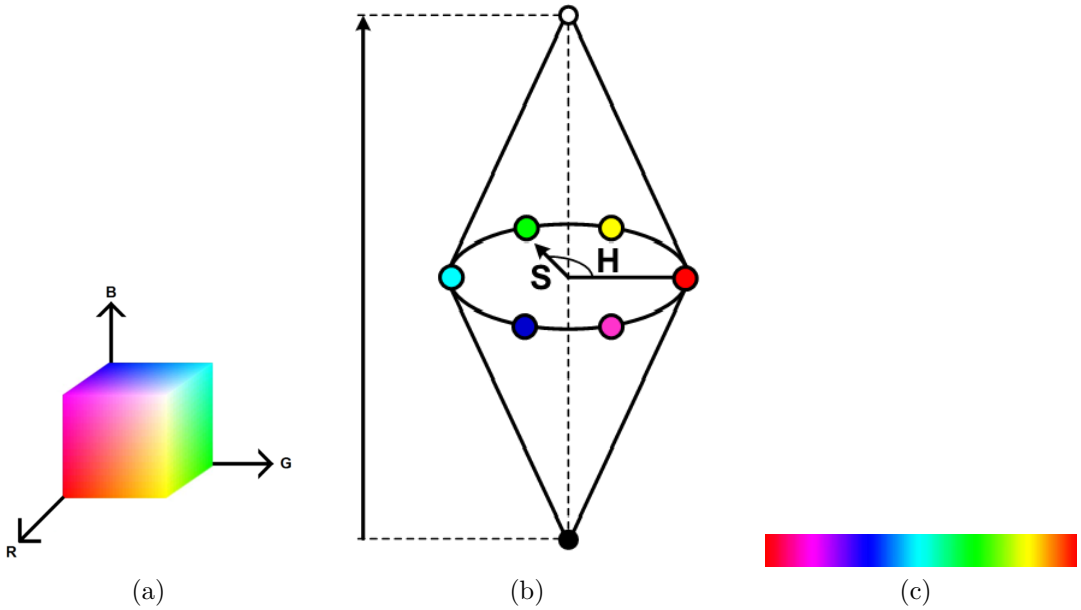
$$r = \frac{R}{R+G+B}, \tag{3.8}$$

$$g = \frac{G}{R+G+B}, \tag{3.9}$$

$$b = \frac{B}{R+G+B}, \tag{3.10}$$

$$A = \sqrt{\left(r-\frac{1}{3}\right)^2+\left(b-\frac{1}{3}\right)^2+\left(g-\frac{1}{3}\right)^2}, \tag{3.11}$$

$$B = \frac{2}{3}(r-\frac{1}{3})-\frac{1}{3}(b-\frac{1}{3})-\frac{1}{3}(g-\frac{1}{3}). \tag{3.12}$$



(a) (b) (c)

Figure 3.5: Video representation systems: (a) RGB; (b) HSI; (c) The range of the Hue value. (Source: [1])

$$\theta = \arccos\left(\frac{B}{A\sqrt{\frac{2}{3}}}\right)\frac{180}{\pi}, \tag{3.13}$$

$$H = \begin{cases} \theta & \text{, if } g \geq b; \\ 360^\circ - \theta & \text{, otherwise.} \end{cases}, \tag{3.14}$$

$$S = 1 - 3\min(r, g, b), \tag{3.15}$$

$$I = \frac{R + G + B}{3}. \tag{3.16}$$

After this transformation, the color of each pixel can be used to calculate the dominant color in the frame. Using a pixel-level hue histogram within a frame, the dominant color appears as the mode of such distribution. For high values of intensity (I) or saturation (S), the hue value (H) is not reliable, as these limits correspond to the white and black colors, respectively. Hence, the following equations are used

to calculate the dominant color:

$$H[i_{min} - 1] < K * H[i_{peak}] \leq H[i_{min}], \tag{3.17}$$

$$H[i_{max} + 1] < K * H[i_{peak}] \leq H[i_{max}], \tag{3.18}$$

$$i_{min} \leq i_{peak} \leq i_{max}, \tag{3.19}$$

$$DominantHue = \frac{\sum_{i=i_{min}}^{i_{max}} H[i] * i}{\sum_{i=i_{min}}^{i_{max}} H[i]}. \tag{3.20}$$

where $K = 0.2$ corresponds to a 20% safety margin for the respective variable.

During a football transmission, graphical insertions can interfere in the dominant-color computation, as they usually present some artificial color, which corresponds to spikes in the color histogram. To remove these undesirable peaks, a simple moving average filter is used to smooth out the histogram, before detecting the dominant color.

A distinct feature associated with the dominant color is the percentage of pixels associated with that color.

Another phenomenon which usually occurs during a highlight is an increase on the camera movement. For this reason, the information of the camera type and movement are also employed as video features. The algorithm for estimating the camera movement attempts to quantify the global movement of the image and its objects. A simple way to calculate the global movement is using the correlation between subsequent video frames. Following this reasoning, the phase correlation [28], represented by the vector (v_x, v_y), quantifies the translational movement of image I_1 with respect to image I_2:

$$I_2(x, y) = I_1(x - v_x, y - v_y). \tag{3.21}$$

Applying the Fourier transform, one gets

$$F_2(m, n) = F_1(m, n)e^{-\pi j(mv_x + nv_y)}, \tag{3.22}$$

where F represents the Fourier transform of the corresponding I function [29].

Hence, one has that

$$F_{1,2}(m, n) = F_1(m, n)F_2(m, n)^* = F_1(m, n)F_1(m, n)^* e^{2\pi j(mv_x + nv_y)}, \tag{3.23}$$

which leads to

$$C(x,y) = IDFT\left\{\frac{F_{1,2}(m,n)}{|F_1(m,n)||F_2(m,n)|}\right\} = \delta(x - v_x, y - v_y), \tag{3.24}$$

where the Dirac's impulse represents the shift between the images I_1 and I_2. In practice, however, $C(x,y)$ is not a perfect impulse function, as the relationship between these two images is not perfectly translational, as several objects present in the scene (ball, players, and so on) may have independent movements.

Using the polar coordinate system, one can represent the camera movement by its magnitude Δ and direction θ such that

$$\Delta = \sqrt{x^2 + y^2} \quad , \tag{3.25}$$

$$\theta = \text{arctg}\left(\frac{y}{x}\right) \quad . \tag{3.26}$$

, this parameters were added to the set of features too.

Another parameter of interest is the amplitude ρ of the peak generated by equation (3.24). Such value provides an indication of how reliable is the movement estimate, where higher ρ values correspond to more global movements. The variances of these values within a 15-frame interval (which correspond to about a half second) also provide some degree of confidence for such estimates.

3.2.3 Feature Collection

Table 3.1 summarizes all audio and video features employed by reference [1], as discussed in the previous subsections.

3.3 Database

The database employed for system training is probably as much important as the choice of the classifier and its features for system success. The database must represent the process in a reliable and compact manner. In our problem, the database must be diversified about the type of events being selected as highlights, the game time, the stadiums, the narrator, the teams involved, the broadcasting style and so on. Table 3.2 summarizes the employed database of 30 games representing the following aspects [1]:

- 5 different tournaments which show different production rules, different shots and effects;
- 30 different teams which bring different jersey colors;

Table 3.1: Low-level audio and video features employed in reference [1].

Label	Parameters	Abbreviation	Feature	Type
1	Hue Mean 3.18	Hue	Dominant Color	Video
2	Percent of pixels associated to Hue	PHue	Dominant Color	Video
3	Δ 3.25	Δ	Camera Movement	Video
4	θ 3.26	θ	Camera Movement	Video
5	ρ Peak of correlation	ρ	Camera Movement	Video
6	$VAR(\Delta)$	$VAR(\Delta)$	Camera Movement	Video
7	$VAR(\theta)$	$VAR(\theta)$	Camera Movement	Video
8	Pitch 3.2	Pitch	Pitch Estimation	Audio
9	Pitch Diff 3.5	PitchDiff	Pitch Estimation	Audio
10	Pitch Diff Ascending 3.7	PitchDiffA	Pitch Estimation	Audio
11	Short-time Energy 3.3	STE	Energy	Audio
12	Short-time Energy Diff 3.5	STEDiff	Energy	Audio
13	Short-time Energy Diff Ascending 3.7	STEDiffA	Energy	Audio
14	Comb Short-time Energy 3.4	CSTE	Energy	Audio
15	Comb Short-time Energy Diff 3.5	CSTEDiff	Energy	Audio
16	Comb Short-time Energy Diff Ascending 3.7	CSTEDiffA	Energy	Audio

- 15 different stadiums which resulted in different color of the grass and camera distance;
- Different time of the day which changes all the color set of the camera;
- 4 different narrators who have different voices and pitches;
- 86 goals and 421 goal attempts.

The game signals were captured by TV Globo in the SD system with 486 lines and 720 columns, with a frame rate of 29,97 frames per second encoded in MPEG-2 at 6 Mbps. The audio was captured with 48000 samples per second and encoded in PCM.

For the labeling, every goal attempt (like any shot in the direction of the goal, including the scored goals, or any moment inside the penalty box that the attacking team has the possession of the ball and has a chance to score) was considered as a "good moment".

3.4 The Classifier

The classifier employed in [1] was of the AdaBoost type, which aggregates several weak (with low discriminative rate) features to the system to form a strong overall classifier.

One of the initial AdaBoost drawbacks is its inability to deal with temporal evolution, as it considers each input sample independent to the others. This problem was solved using past and future samples as additional features to the system.

Table 3.2: Database composition. G = Number of goals. LP = Number of goal attemp. (Source: [1])

	Game	Tournament	Stadium	Narrator	Time of the day	G	LP
1	Argentina x Germany	World Cup 2010	Green Point	Luís Roberto	Afternoon	3	17
2	Argentina x Mexico	World Cup 2010	Soccer City	Cléber Machado	Night	4	13
3	Argentina x Nigeria	World Cup 2010	Ellis Park	Galvão Bueno	Afternoon/Night	1	17
4	Argentina x South Korea	World Cup 2010	Soccer City	Cléber Machado	Afternoon	5	11
5	Brazil x Chile	World Cup 2010	Ellis Park	Galvão Bueno	Night	3	14
6	Brazil x Netherlands	World Cup 2010	Port Elizabeth	Galvão Bueno	Afternoon/Night	3	12
7	Chile x Switzerland	World Cup 2010	Port Elizabeth	Rogério Pinheiro	Afternoon/Night	2	13
8	Denmark x Japan	World Cup 2010	Royal Bafokeng	Luís Roberto	Night	4	15
9	France x Mexico	World Cup 2010	Peter Mokaba	Galvão Bueno	Night	2	12
10	Germany x England	World Cup 2010	Free State	Luís Roberto	Afternoon/Night	5	20
11	Germany x Spain	World Cup 2010	Durban	Galvão Bueno	Night	1	13
12	Germany x Uruguay	World Cup 2010	Port Elizabeth	Cléber Machado	Night	5	13
13	Italy x Slovakia	World Cup 2010	Ellis Park	Cléber Machado	Afternoon/Night	6	11
14	Netherlands x Japan	World Cup 2010	Durban	Cléber Machado	Afternoon	1	11
15	Netherlands x Slovakia	World Cup 2010	Durban	Luís Roberto	Afternoon/Night	2	4
16	Portugal x North Korea	World Cup 2010	Green Point	Luís Roberto	Afternoon	7	15
17	Spain x Netherlands	World Cup 2010	Soccer City	Galvão Bueno	Night	0	18
18	Spain x Portugal	World Cup 2010	Green Point	Cléber Machado	Night	1	17
19	Spain x Switzerland	World Cup 2010	Durban	Luís Roberto	Afternoon/Night	1	18
20	Uruguay x Netherlands	World Cup 2010	Green Point	Cléber Machado	Night	5	13
21	Uruguay x South Korea	World Cup 2010	Port Elizabeth	Luís Roberto	Afternoon/Night	3	18
22	Brazil x Italy	Confederation Cup 2009	Loftus Versfeld	Galvão Bueno	Night	3	22
23	Spain x United States	Confederation Cup 2009	Free State	Cléber Machado	Night	2	16
24	Atlético Mineiro x Vasco	Campeonato Brasileiro 2010	Mineirão	Luís Roberto	Afternoon/Night	3	15
25	Corinthians x Fluminense	Campeonato Brasileiro 2010	Pacaembu	Luís Roberto	Afternoon/Night	1	15
26	Santos x Vasco	Campeonato Brasileiro 2010	Vila Belmiro	Rogério Pinheiro	Afternoon/Night	4	10
27	Barcelona x Internazionale	Champions League 2010	Camp Nou	Galvão Bueno	Night	2	9
28	Bayern MÃ$\frac{1}{4}$nich x Internazionale	Champions League 2010	Santiago Bernabéu	Galvão Bueno	Afternoon/Night	2	14
29	Cruzeiro x São Paulo	Copa Libertadores 2010	Mineirão	Cléber Machado	Night	3	10
30	São Paulo x Cruzeiro	Copa Libertadores 2010	Morumbi	Cléber Machado	Night	2	15

A post-processing step was added to the final system, using a median filter to smooth out the frame-based highlight classification. The implementation used is the GML AdaBoost Matlab Toolbox [30].

The AdaBoost has two main parameters to be configured. One is the depth of the internal tree, which was set to 3. The other parameter is the number of iterations which the classifier weights are adjusted in the training stage.

A k-fold cross-validation process was employed to evaluate the robustness of the resulting classifier performance. This process consists of selecting one fold to validate the process and using the others for training. The process is repeated selecting another fold to be the validation fold, and so on, until all folds are used as the validation fold, as illustrated in Figure 3.6 and summarized in Table 3.3.

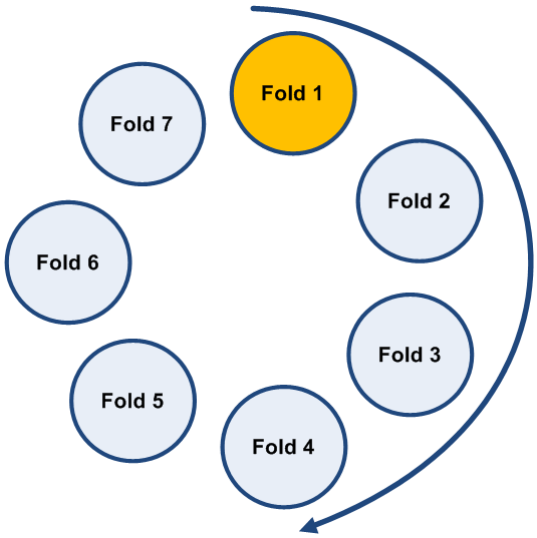


Figure 3.6: Representation of k-fold cross-validation technique to evaluate robustness of overall system performance. (Source: [1])

The system performance was evaluated using the following metrics: the precision rate (PR, percentage of detected highlights which were true highlights in the database annotation stage), the recall rate (RR, percentage of true highlights detected by the system), and the summarization rate (SR, game percentage classified as a highlight), defined as

$$\begin{aligned} PR(\%) &= \frac{\text{Good Moments Correctly Detected}}{\text{Good Moments Detected}} \quad , \\ RR(\%) &= \frac{\text{Good Moments Correctly Detected}}{\text{Good Moments Existing}} \quad , \\ SR(\%) &= \frac{\text{Duration Good Moments Detected}}{\text{Duration All Moments}} \quad , \end{aligned} \tag{3.27}$$

Table 3.3: Game division for the 8-fold cross-validation scheme. (Source: [1])

	Match	Narrator	Time of the day
Fold 1	Argentina x Germany	Luís Roberto	Afternoon
	Spain x Portugal	Cléber Machado	Night
	Corinthians x Fluminense	Luís Roberto	Afternoon/Night
	Barcelona x Internazionale	Galvão Bueno	Night
Fold 2	Denmark x Japan	Luís Roberto	Night
	Netherlands x Japan	Cléber Machado	Afternoon
	Uruguay x South Korea	Luís Roberto	Afternoon/Night
	Spain x EUA	Cléber Machado	Night
Fold 3	Germany x England	Luís Roberto	Afternoon/Night
	Italy x Slovakia	Cléber Machado	Afternoon/Night
	Spain x Netherlands	Galvão Bueno	Night
	Cruzeiro x São Paulo	Cléber Machado	Night
Fold 4	Brazil x Chile	Galvão Bueno	Night
	France x Mexico	Galvão Bueno	Night
	Germany x Uruguay	Cléber Machado	Night
	Netherlands x Slovakia	Luís Roberto	Afternoon/Night
Fold 5	Argentina x Nigeria	Galvão Bueno	Afternoon/Night
	Germany x Spain	Galvão Bueno	Night
	Portugal x North Korea	Luís Roberto	Afternoon
	Atlético Mineiro x Vasco	Luís Roberto	Afternoon/Night
Fold 6	Argentina x South Korea	Cléber Machado	Afternoon
	Spain x Switzerland	Luís Roberto	Afternoon/Night
	Uruguay x Netherlands	Cléber Machado	Night
	Brazil x Italy	Galvão Bueno	Night
Fold 7	Argentina x Mexico	Cléber Machado	Night
	Brazil x Netherlands	Galvão Bueno	Afternoon/Night
	Bayern Munich x Internazionale	Galvão Bueno	Afternoon/Night
	São Paulo x Cruzeiro	Cléber Machado	Night
Fold 8	Chile x Switzerland	Rogério Pinheiro	Afternoon/Night
	Santos x Vasco	Rogério Pinheiro	Afternoon/Night

respectively.

3.5 Results

Tables 3.4 and 3.5 provide a breakdown of the overall results achieved by [1]. In Table 3.4, the 'Operational' mode disregards any error on the beginning and end classification of each detected highlight.

Table 3.4: The automatic and operational validation applied to each fold of the database. (Source: [1])

	Automatic		Operational	
Fold	PR (%)	RR (%)	PR (%)	RR (%)
1	92.32	91.98	83.52	100
2	95.39	94.83	89.28	100
3	90.24	86.07	86.66	95.78
4	92.30	93.07	79.72	100
5	92.37	89.91	86.74	97.29
6	93.89	90.88	84.37	94.18
7	91.26	95.43	67.24	100
Average	92.55	91.74	82.50	98.17
Standard Deviation	1.68	3.19	7.37	2.44

An interesting analysis performed in [1] evaluated the system performance considering video (V) and audio (A) features, in isolated and combined manners. Results in this case are shown in Table 3.6 where one concludes that the audio features already provide a good system performance, whereas the video features only increase the final performance in a marginal rate.

The overall results achieved by the system presented in [1] are provided in Table 3.7.

3.6 Conclusion

This Chapter presented the highlight-detection system developed in [1], detailing its performance for the corresponding database. The present dissertation develops several improvements to the system in an attempt to reduce its computational complexity and increase its detection efficiency and robustness.

Table 3.5: Classification applied to full games of the database. The bold lines is the games which aren't in the training set. IM = Missed important moment. G = Missed goals. (Source: [1])

Match	MIM	MG	PR (%)	RR (%)	SR (time)	SR (%)
1	1	0	20.21	95.00	10:39	13.62
2	0	0	13.38	100	15:37	16.42
3	1	0	13.38	94.40	10:24	10.94
4	1	0	14.58	93.33	11:34	12.42
5	0	0	6.39	100	14:42	15.77
6	1	0	11.29	93.33	12:37	13.37
7	**0**	**0**	**12.29**	**100**	**10:15**	**10.77**
8	1	0	18.55	94.73	9:16	9.64
9	0	0	14.73	100	5:39	6.01
10	0	0	25.00	100	12:23	13.30
11	2	0	28.20	84.60	2:16	2.40
12	0	0	19.14	100	11:06	11.77
13	0	0	17.89	100	11:39	11.92
14	0	0	15.00	100	6:24	6.80
15	0	0	26.08	100	1:45	5.81
16	1	0	22.58	95.45	6:58	12.69
17	3	0	35.71	83.33	2:34	2.70
18	1	0	13.17	94.44	12:48	13.57
19	0	0	21.68	100	7:35	8.10
20	2	0	19.73	88.23	12:27	12.85
21	0	0	22.58	100	10:09	10.79
22	0	0	17.98	100	10:54	11.57
23	0	0	10.65	100	18:53	20.59
24	0	0	17.82	100	7:46	8.30
25	1	0	15.38	93.33	7:37	8.08
26	0	0	12.71	100	11:15	12.10
27	0	0	5.85	100	23:35	28.32
28	**0**	**0**	**12.31**	**100**	**17:14**	**18.25**
29	0	0	8.96	100	12:27	13.05
30	0	0	12.50	100	20:22	21.93
Average	0.46	0	16.85	97.00	10:57	12.12

Table 3.6: Precision and Recall Rates resulting of the experimental script. A = Audio. V = Video. MF = Median Filter(page 21). PF = Past/Future(page 19) Samples. (Source: [1])

Test	Summary	Precision Rate		Recall Rate	
		Average	Std Dev	Average	Std Dev
1	V	61.21%	2.30%	63.69%	11.29%
2	V + MF	61.91%	2.45%	64.75%	11.99%
3	V + PF	65.70%	2.11%	66.40%	9.24%
4	V + MF + PF	66.34%	2.41%	67.23%	9.73%
5	A	83.58%	1.86%	80.24%	4.94%
6	A + MF	86.56%	2.32%	82.27%	5.81%
7	A + PF	90.71%	2.30%	90.04%	4.02%
8	A + MF + PF	91.78%	2.65%	90.61%	4.40%
9	A + V	83.37%	1.41%	80.88%	3.73%
10	A + V + MF	85.95%	1.96%	82.50%	4.66%
11	A + V + PF	91.22%	1.60%	90.92%	2.90%
12	A + V + MF + PF	92.55%	1.68%	91.74%	3.19%

Table 3.7: System configuration and performance metrics. (Source: [1])

	PR Optimal	RR Optimal
Average	92.61%	91.92%
Standard Deviation	1.78%	2.87%
AdaBoost Type	*Gentle*	*Gentle*
$Iterations$	31	33
$Past/Future$	30	30
$SizeofMedianFilter$	15	9
Average Recall	91.82%	-
Average Precision	-	92.59%

Chapter 4

Feature Selection

4.1 Introduction

The work presented in [1] uses a large set of features to achieve good performance in the summarization of football games. In this Chapter, the role of each of these features in discriminating the events of interest is further investigated. The idea is to find a good classifier with a minimum number of features in order to reduce the computational complexity of the overall system. In this analysis, we consider the cross-correlations between all pairs of features and the correlation of each individual feature with the classifier output.

But at first, with a manual analysis of the database provided by [1], repeated columns were found in the features set. The samples extracted for the features *Comb Short-time Energy*, *Comb Short-time Energy Diff* and *Comb Short-time Energy Diff Ascending* seems to be accidentally repeated as the samples for the features *Modified Comb Short-time Energy*, *Modified Comb Short-time Energy Diff* and *Modified Comb Short-time Energy Diff Ascending* respectively. This means these three features aren't been effectively used, because replicated data do not provide useful information to the system. Then, these three last mentioned features are going to be removed from the set considering that will not be any loss to the system.

4.2 Feature Cross-correlations

The cross-correlation exposes the linear dependence of each pair of features X_i and X_j and is a very efficient technique to identify redundant information from a group of features. This analysis is based on the Pearson's correlation coefficient defined as

$$\rho_{X_i,X_j} = \frac{E((X_i - \mu_{X_i})(X_j - \mu_{X_j}))}{\mu_{X_i}\mu_{X_j}}, \tag{4.1}$$

where μ_{X_k} indicates the mean value of feature X_k. A high correlation, in its absolute value, indicates that the information in both features are very similar. To decorrelate the features and verify if there is still some relevant information, one may compute the partial correlation coefficients

$$\rho'_{X_i,X_j} = \frac{(\rho_{X_2,Y} - \rho_{X_1,Y} * \rho_{X_1,X_2})}{\sqrt{1 - \rho^2_{X_1,X_2}}}. \tag{4.2}$$

By estimating the cross-correlation between all features, audio features and video features shown a very low correlation between each other due to their different nature. Then, the correlation table is going to be separated in two parts, the video Table 4.1 and the audio Table 4.3 to simplify the analysis of the Tables. This was done to the Table of the partial correlation coefficients of the video features 4.2 and the audio features 4.4 too.

Table 4.1: Video-feature correlations ρ.

	Hue	PHue	Δ	θ	ρ	$VAR(\Delta)$	$VAR(\theta)$
Hue	1.00	-0,24	-0,04	-0.03	0.01	0.03	0.05
PHue		1.00	0.15	0.03	0.02	-0.02	-0.01
Δ			1.00	-0.13	0.18	-0.15	-0.04
θ				1.00	0.10	0.29	0.02
ρ					1.00	-0.08	0.15
$VAR(\Delta)$						1.00	0.02
$VAR(\theta)$							1.00

Table 4.2: Video-feature partial correlations ρ.

	Hue	PHue	Δ	θ	ρ	$VAR(\Delta)$	$VAR(\theta)$
Hue	1.08	0,22	-0,02	0.04	0.00	-0.02	-0.06
PHue		1.14	-0.13	-0.06	0.00	-0.01	-0.01
Δ			1.12	0.13	-0.22	0.09	0.07
θ				1.13	-0.17	-0.32	0.01
ρ					1.10	0.12	-0.17
$VAR(\Delta)$						1.13	-0.03
$VAR(\theta)$							1.03

In Table 4.1, one clearly notices how the video correlations tend to be very low, indicating a low level of redundant information. The same is noticed in the Table 4.2. On the other hand, when analyzing the audio-feature correlations in Table 4.3 and Table 4.4, one immediately observes high values (above 0.80, for instance) indicating redundant information. In particular, $\rho_{STE,CSTE}$, $\rho_{STEDiff,CSTEDiff}$ and $\rho_{STEDiffA,CSTEDiffA}$ reached high correlation in both tables. Using the correlation to the output (as given in Section 4.3) as a criterion for feature selection, we

Table 4.3: Audio-feature correlations.

	Pitch	PitchDiff	PitchDiffA	STE	STEDiff	STEDiffA	CSTE	CSTEDiff	CSTEDiffA
Pitch	1.00	0.18	-0.12	0.25	0.14	-0.03	0.28	0.15	-0.05
PitchDiff		1.00	-0.22	0.24	0.41	-0.16	0.18	0.37	-0.17
PitchDiffA			1.00	-0.14	-0.09	0.30	-0.09	-0.09	0.32
STE				1.00	0.39	-0.13	0.59	0.42	-0.16
STEDiff					1.00	-0.09	0.32	0.85	-0.08
STEDiffA						1.00	-0.08	-0.09	0.80
CSTE							1.00	0.42	-0.10
CSTEDiff								1.00	-0.09
CSTEDiffA									1.00

Table 4.4: Audio-feature partial correlations.

	Pitch	PitchDiff	PitchDiffA	STE	STEDiff	STEDiffA	CSTE	CSTEDiff	CSTEDiffA
Pitch	1.16	-0.08	0.07	-0.07	0.07	-0.06	-0.22	0.00	-0.02
PitchDiff		1.38	0.16	0.03	-0.31	0.02	0.04	-0.10	0.04
PitchDiffA			1.17	0.03	-0.05	-0.12	-0.01	0.00	-0.24
STE				1.78	-0.21	-0.03	-0.81	-0.07	0.12
STEDiff					3.91	0.03	0.33	-3.12	-0.10
STEDiffA						2.82	-0.01	0.00	-2.21
CSTE							1.70	-0.64	0.01
CSTEDiff								3.94	0.03
CSTEDiffA									2.88

may considered excluding features STEDiffA, CSTE and CSTEDiff from a modified classifier version.

4.3 Correlation Between Features and Labels

The cross-correlation between the remaining features, after the selection performed in previous section, and the labels (highlight or non-highlight classes) may also evaluate the contribution of each feature to the final result of the classification, as seen in Table 4.5. This analysis, however, only explains the linear dependence between the given feature and the labels and any non-linear contribution is not evaluated properly by this metric. In any case, the results of new classifications will confirm the analysis. The bold features are the already excluded ones.

From these results, one observes that θ, ρ, $VAR(\Delta)$, and $VAR(\theta)$ are the features with the lowest absolute values for the correlation coefficient, indicating that those features may also be disregarded. Curiously, all these features are from the video signal.

Table 4.5: Cross-correlation between features and system output.

	Y
Hue	0.11
PHue	-0.18
Δ	-0.15
θ	0.03
ρ	0.01
$VAR(\Delta)$	0.10
$VAR(\theta)$	-0.01
Pitch	0.29
PitchDiff	0.40
PitchDiffA	-0.20
STE	0.42
STEDiff	0.41
STEDiffA	**-0.19**
CSTE	**0.29**
CSTEDiff	**0.35**
CSTEDiffA	-0.21

4.4 Analysis of Feature Relevance

In this section, we evaluate the effect of each remaining feature on the classifier error, by cross-correlating the feature with the classification error obtained relatively to the expected output of each sample. Therefore, in this analysis we incorporate the classifier (in this case, the AdaBoost classifier, used in [1]) into the feature-analysis loop. The result is expressed in the Table 4.6, where ρ is the correlation to the output of the classification and ϵ is the correlation to the classification error. The bold features are the already excluded ones.

Table 4.6: Feature relevance analysis with respect to the classifier output error.

	ρ_o	ϵ
Hue	0.11	0.02
PHue	0.17	0.01
Δ	0.15	0.01
θ	**0.02**	**0.00**
ρ	**0.01**	**0.01**
$VAR(\Delta)$	**0.10**	**0.01**
$VAR(\theta)$	**0.01**	**0.02**
Pitch	0.29	0.07
PitchDiff	0.40	0.06
PitchDiffA	0.20	0.04
STE	0.41	0.06
STEDiff	0.40	0.05
STEDiffA	**0.19**	**0.03**
CSTE	**0.28**	**0.03**
CSTEDiff	**0.34**	**0.03**
CSTEDiffA	0.20	0.06

The features with the highest correlation to the error is Pitch, PitchDiff, STE, and CSTEDiffA, and in principle there should be excluded. However Pitch, PitchDiff, and STE have significant correlation to the output(ρ_o), indicating that if this features were excluded, the system may lose some useful information for the classifier, depreciating the classification results. For that reason, the only feature that can be excluded from the system is CSTEDiffA, which is the Comb Short-time Energy Diff Ascending.

4.5 Decimation

For increasing the classifier performance, [1] also investigated the use of past and future features to classify a given frame. In the end, to maximize the performance, it was concluded that a window of length 60 (30 from the past and 30 from the future) provided the best classification results. For a rate of 29.97 frames per second, this window length corresponds to an interval of 1 second for the past and 1 second for the future features.

An immediate consequence of this feature gathering is the multiplication in the number of input features by a factor of 61, many of them most probably with high correlation to each other. To reduce this information redundancy we may consider inserting an interval between the frames involved in this feature grouping. In that case, instead of adding features from every adjacent frame, we consider the features every K frames, with $K \geq 1$.

The classification results when using only 8 remaining features (Hue, PHue, Δ, Pitch, PitchDiff, PitchDiffA, STE, and STEDiff), instead of the initial 16, and $K = 2$ (in comparison to the original $K = 1$) are shown in Table 4.7. These results consider a complete retraining of the AdaBoost classifier, using the same k-fold strategy as before.

Table 4.7: Results of simplified classifier using feature selection techniques. Where the Inputs are the number of input features which are going to be used in the classifier. It is the multiplication of the number of Features and the number of Frames.

Features	Frames	Inputs	Precision Rate	Recall Rate
16	61	976	92.55%	91.74%
8	61	488	92.30%	91.34%
16	31	496	92.06%	90.94%
8	31	248	92.15%	91.22%

From Table 4.7, one observes how the simplified classifier yields precision and recall rates quite similar to the original classifier configuration proposed in [1], despite

the reduced number of input features (only 248 as opposed to 976).

4.6 Video Dependence

The video features have higher costs to be extracted than the audio features. Then, it is very important to have a confirmation about the true contribution of the video features to the overall system. This analysis can be performed extensively by running the system without those features and analyzing the results. After the feature selection, there are only 3 remaining video features left: mean hue value, image percentage for the mean hue value and motion variation vector 3.25. The influence of these particular features is summarized in Table 4.8.

Table 4.8: Results of the Feature Selection Removing the Video Features

Number of Features	Frames	Inputs	Precision Rate	Recall Rate
5	31	155	91.67%	90.54%

Such results indicate a small decrease on the error and recall rates, what may lead one to think that these features can be removed altogether from the classifier. However, as we will later discuss, this classifier configuration without any video feature looses its ability to generalize the results for other narrators not included in the training database.

4.7 Conclusion

This Chapter discussed some possible simplifications to the original classifier configuration proposed in [1]. After the feature selection, 8 out of the original 16 features were shown to have quite redundant information. Moreover, we considered the case of sub-sampling information from intermediary frames when aggregating past and future frames to the classifier inputs. It was verified how these simplifications led to no significant loss of performance, despite a feature reduction to only about 25% of the original set. This reduction also opens new doors to alternative classifier families, in opposition to the AdaBoost group, which is the subject of the following Chapters.

Chapter 5

Classifiers

5.1 Introduction

As can be imagined, one of the most important steps in classification of a given process is the choice of the best classifier for the specific case. There are classifiers which need more attention before being used, due to their limitation in the number of input features, like the support vector machine, for example. And there are more general classifiers, like AdaBoost and Random Forest, which can be used without much issues about inserting raw data, without any processing or feature selection since these stages are inherent to the classifier training stage. Of course, the processing and feature selection can be used to optimize even those generic classifiers.

According to [31], classifier systems are constituted of five phases: sensing, segmentation, feature extraction, classification, and post processing. The sensing is when the nature of the event, which is going to be observed, is captured and converted into data. In this case, the video and audio being captured during a football broadcast. The segmentation is the stage when the sensing content is separated according to the needs of the classifier. The feature extraction is when the features that are going to be used for this classifier are extracted from the data segments previously obtained. The classification phase is when those features are analyzed to detect to which class each sample belongs. The post processing stage is dedicated to eventual methods to achieve better classification results.

The main scope of this Chapter is to present the main characteristics of some o the most important classifier families found in the related literature. We, therefore, cover the AdaBoost, random forest and support vector machine classifiers, highlighting their main advantages and disadvantages.

5.2 AdaBoost

The so-called Adaptive Boosting (AdaBoost) algorithms constitute an interesting family of classifiers which allows the combination of a large number of features with low discriminating ability to generate a strong final classifier [32]. A short description of the AdaBoost classifier is presented here, whereas the full formulation can be found in [33].

Consider an ensemble of N training samples, constituted by the pair (x_n, y_n), where x_n is the d-dimensional matrix of the training features and y_n is the vector of labels belonging to $\{-1, 1\}$. Those features and labels (x_n, y_n) of each sample are associated with a probability distribution $D_t(n)$, in each t iteration ,for the event of each sample being classified as label 1. At $t = 1$, by convention, the distribution $D_t(n)$ is considered uniform.

After each training iteration, a classifier h_t is determined to achieve the best classification for all samples x_n and the classification error is calculated as

$$e_t = \sum_{m|h_t(x_m)\neq y_n} D_t(m). \tag{5.1}$$

This classification error is then employed to adjust the distribution and the classifier weights according to the relationships

$$\begin{aligned} \alpha_t &= \frac{1}{2}\ln\left(\frac{1-e_t}{e_t}\right) \quad , \\ D_{t+1}(n) &= \frac{D_t(n)e^{-\alpha_t y_n h_t(x_n)}}{z_i} \quad , \end{aligned} \tag{5.2}$$

where z_i is a normalization factor that guarantees that $\sum_{m=1}^{N} D_t(m) = 1$. The classification is implemented by:

$$H(x_m) = \text{sign}\left(\sum_{i=0}^{T} \alpha_i h_i(x_m)\right) \quad , \tag{5.3}$$

where T is the number of training iterations and sign is the signal operator.

Starting from this concept, three AdaBoost variations later appeared in the related literature: the real ([32]), gentle ([34]) and modest ([35]), the differences of which are explained below.

In the real AdaBoost, the weights for the distribution $D_{t+1}(n)$ are given by

$$\begin{aligned} W_{+1} &= \sum_{m|y_m h_t(x_m)=+1} D_t(m) \quad ,, \\ W_{-1} &= \sum_{m|y_m h_t(x_m)=-1} D_t(m) \quad ,, \\ \alpha_t &= \frac{1}{2}\ln\left(\frac{W_{+1}}{W_{-1}}\right) \quad .. \end{aligned} \tag{5.4}$$

In addition, the classification results are real numbers as $|h_t(x_n)|$ can be seen as the classification reliability.

The gentle AdaBoost uses a different weight computation

$$\alpha_t = \frac{1}{2}\ln\left(\frac{W_{+1} - W_{-1}}{W_{+1} + W_{-1}}\right), \tag{5.5}$$

which leads to a stronger and more robust classifier.

Finally, the modest AdaBoost, is defined by the updating equations

$$\begin{aligned} \overline{D}_t(m) &= (1 - D_t(m))\overline{z}_i \quad , \\ \overline{W}_{+1} &= \sum_{m|y_m h_t(x_m)=+1} \overline{D}_t(m) \quad , \\ \overline{W}_{-1} &= \sum_{m|y_m h_t(x_m)=-1} \overline{D}_t(m) \quad , \\ \alpha_t &= W_{+1}(1 + \overline{W}_{+1}) - W_{-1}(1 - \overline{W}_{-1}) \quad , \end{aligned} \tag{5.6}$$

where $\overline{z_i}$ normalizes the resulting distribution. This AdaBoost version tries to generalize the classifier strategy, but it can cause some increase in the resulting error.

The most important advantage of the AdaBoost is its strength in complex systems. Even if it is used with a very high number of inputs, high number of samples and not normalized variables, it can reach acceptable results, although in some cases it may take a long time to converge in the training process.

The main disadvantage of the AdaBoost is the fact the samples inserted into the classifier cannot, at first, carry any temporal relation among them. This becomes an inconvenient when the process at hand has a strong consistency along time, as is the case of the football-highlight system of ours. In this case, it is necessary to create a mechanism which adapts the AdaBoost to that characteristic. As the AdaBoost allows the usage of many features as necessary, the method used in [1] consists of inserting the same features of the previous and subsequent frames as additional features to the present frame.

The AdaBoost implementation considered in [1] and also employed in the present work was found in the GML AdaBoost Matlab Toolbox [30]. This AdaBoost imple-

mentation has two main parameters: the depth of the internal tree, which represent the maximum number of queries that can happen before a leaf is reached and a result obtained and it was selected as 3 because of its performance, and the number of training iterations, which means the number of times the weights are going to be adjusted. The Matlab Toolbox provides all three variations of AdaBoost: Real, Gentle and Modest.

Finally, as the classifier offers the classification result in a frame-by-frame basis, it is interesting to apply some level of post-processing (such as a median filter, for instance) to smooth out the the spurious responses of the classifier along time.

5.3 Random Forest

Decision trees use a series of binary decisions in individual features to perform the classification task. The random forest approach uses a (sufficiently) large ensemble of decision trees to add robustness to the overall classification. The ensemble may be created by bagging [36], which creates each tree using a distinct subset of the training data. Reference [37] also proposed a random feature choice in constructing the individual trees. These two approaches can be combined to generate a more robust forest.

According to [38], a random vector Θ_k is generated representing the kth tree, independent of the past random vectors $\Theta_1, ...\Theta_{k-1}$ but with the same distribution, resulting in classifier $h(x, \Theta_k)$, where x is an input feature vector. A random forest is a classifier consisting of a collection of decision trees $\{h(x, \Theta_k), k = 1, ...\}$, where the $\{\Theta_k\}$, ideally, are independent and identically distributed random vectors and each tree casts a unit vote for the most popular class at input x.

The implementation of a Random Forest algorithm starts by creating a large number of decorrelated trees, then creating subsets of the samples to train each tree. But different of the tree bagging algorithm, the learning algorithm for each tree also selects a subset of features at random to train each tree, this action creates a model with low variance.

The square root of the samples are left out of the sample. This oob (out-of-bag) data is used to get a running unbiased estimate of the classification error as trees are added to the forest. It is also used to get estimates of variable importance.

After all the trees are trained, each tree with get it own decision for each sample. The class which receives more votes is the output class.

The random forest is a robust classifier which can deal with a large number of input features, even hundreds, with a small amount of information, similar to the AdaBoost approach. When comparing these two classifiers, namely the random forest and the AdaBoost, in principle the latter has no random elements. However,

its re-weighting procedure during the training stage tends to emulate the random principle of the random forest, as conjectured in [38].

The chosen random forest algorithm used in this work is the TreeBagger found in the Matlab Statistics Toolbox. This algorithm creates an ensemble of bagged decision trees trained under the original database. The main setup parameter for the TreeBagger implementation is the total number of trees. In theory a large number of trees provides stability to the overall classifier. However, an excessive number of trees may cause overtraining. For this aspect, the TreeBagger command has a parameter (OOBPred) which indicates the performance improvement associated with each added tree.

5.4 Support Vector Machine

One of the most used classification and regression techniques for the video summarization problem is the support vector machine (SVM), described by [39]. The SVM technique separates two classes using hyperplanes (or more complex surfaces) which are optimal in the sense of maximum distance between them.

One approach to define the separating surfaces is the so-called hard margin's method, employed when the classes are linearly detachable [40]. In this case, the hyperplanes (H_{+1} and H_{-1}) are such that

$$\begin{cases} \mathrm{w}^t\mathrm{x}_n + b >= +1, \text{ if } y_n = +1 \\ \mathrm{w}^t\mathrm{x}_n + b <= -1, \text{ if } y_n = -1 \end{cases}, \tag{5.7}$$

what corresponds to a distance of

$$l = \frac{2}{||\mathrm{w}||}. \tag{5.8}$$

Therefore, to maximize such a distance, one must minimize $||\mathrm{w}||$, what can be written as a quadratic problem with linear constraints:

$$\min_x \frac{1}{2}||\mathrm{w}||^2, \quad \text{s.t. } y_n(\mathrm{w}^t\mathrm{x}_n + b) - 1 \geq 0, \forall(\mathrm{x}_n, y_n) \in T. \tag{5.9}$$

This problem can be reformulated using Lagrange multipliers α_n, where $n = 1, ..., N$, as

$$L_p = \frac{1}{2}||\mathrm{w}||^2 - \sum_{n=1}^{N} \alpha_n(y_n(\mathrm{w}^t\mathrm{x}_n + b) - 1) + \sum_{n=1}^{N} \alpha_n \tag{5.10}$$

By equating the partial derivatives of L_p with respect to w and b to zero, one gets

that

$$\mathrm{w} = \sum_{n=1}^{N} \alpha_n y_n \mathrm{x}_n, \tag{5.11}$$

$$\sum_{n=1}^{N} \alpha_n y_n = 0, \tag{5.12}$$

which, when plugged into equation (5.10) lead to

$$L_d = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathrm{x}_i^t \mathrm{x}_j, \tag{5.13}$$

The optimized solution $\boldsymbol{\alpha}^*$ of the dual problem must satisfy the Karush-Kuhn-Tucker conditions, which include $\alpha_i \geq 0$, leading to the general case

$$\alpha_n [y_n(\mathrm{w}^t \mathrm{x}_n + b) - 1] = 0, \forall n. \tag{5.14}$$

In this solution, $\alpha_n > 0$ only for the points on the hyperplanes H_{+1} and H_{-1} (which constitute the support vectors). For the other points, the Karush-Kuhn-Tucker conditions are obeyed only if $\alpha_n = 0$. The remaining rules for the classification are given by

$$g(x) = \mathrm{sign}\Big(\sum_{i \in SV} y_i \alpha_i^* \mathrm{x}_i^t \mathrm{x} + b^* \Big), \tag{5.15}$$

$$b^* = \frac{1}{n_{SV}} \sum_{i \in SV} \Big(\frac{1}{y_i} - \sum_{i \in SV} y_i \alpha_i^* \mathrm{x}_i^t \mathrm{x}_i \Big), \tag{5.16}$$

where SV is the set of support vectors and n_{SV} is the number of vectors in this set.

As mentioned before, this method can only be applied to linear separable classes. But most of the times the classes are not linear separable due to noisy samples, outliers and the non-linear nature of the problem. In such cases, one may employ the soft-margin SVM, which consists of inserting some non-negative slack variables ξ_n in the conditions of equation (5.7), which becomes

$$\begin{cases} \mathrm{w}^t \mathrm{x}_n + b >= +1 - \xi_n, \text{ if } y_n = +1 \\ \mathrm{w}^t \mathrm{x}_n + b <= -1 + \xi_n, \text{ if } y_n = -1 \end{cases} \tag{5.17}$$

Those additional variables allow some data samples to be between the hyperplanes if $0 <= \xi_n <= 1$, or even in the region correspondent to the other class (error) if $\xi_n > 1$. Then, reformulating the equations, one gets

$$\min_x \frac{1}{2} ||\mathrm{w}||^2 + C \Big(\sum_{n=1}^{N} \xi_n \Big), \tag{5.18}$$

where C is an arbitrary parameter. Then,

$$y_n(\mathrm{w}^t\mathrm{x}_n + b) - 1 - \xi_n >= 0, \forall(\mathrm{x}_n, y_n) \in T, \tag{5.19}$$

$$L_p = \frac{1}{2}||\mathrm{w}||^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N}\alpha_n(y_n(\mathrm{w}^t\mathrm{x}_n + b) - 1 - \xi_n) + \sum_{n=1}^{N}\mu_n, \tag{5.20}$$

where μ_n is also the Lagrangian multiplier in primal form such that

$$\alpha_n, \mu_n \geq 0. \tag{5.21}$$

Hence, the null partial derivatives correspond to

$$\mathrm{w} = \sum_{n=1}^{N}\alpha_n y_n \mathrm{x}_n, \tag{5.22}$$

$$\sum_{n=1}^{N}\alpha_n y_n = 0, \tag{5.23}$$

$$C - \alpha_n - \mu_n = 0. \tag{5.24}$$

Substituting these relationships into the original formulation, one obtains the dual formulation

$$L_d = \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \mathrm{x}_i^t\mathrm{x}_j, \tag{5.25}$$

with the constraints $0 < \alpha_n < C$,

$$\alpha_i(y_i(\mathrm{w}^t\mathrm{x}_i + b) - 1 + \xi_i) = 0, \mu_n\xi_n = 0. \tag{5.26}$$

After finding the $\boldsymbol{\alpha}^*$ which solves the dual problem, one can use equations (5.22) and (5.26) to determine $\mathbf{w}^*$ and $\mathbf{b}^*$, respectively.

As in the previous method, the support vectors are such that $\alpha^* \geq 0$. However, there are now four types of support vectors: if $\alpha^* < C$, then $\xi_n^* = 0$ and the support vector is over the margin; if $\alpha^* = C$, then there are three cases – an error, if $\xi_n^* > 1$; the sample is between the margin, if $0 < \xi_n^* \leq 1$; and it is over the margin; if $\xi_n^* = 0$.

A third SVM method applies when the soft margin still is not enough for the non-linearly separable classes. In such cases, a non-linear mapping Φ is applied to the feature space increasing its dimension:

$$\Phi : \mathbb{R}^d \to \mathcal{H}, \tag{5.27}$$

where $\mathcal{H}$ is the new high-dimension feature space. One then applies the soft margin

approach to the transformed $\Phi(\mathrm{x}_n)$ features, leading to

$$L_d = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \Phi(\mathrm{x}_i)^t \Phi(\mathrm{x}_j), \tag{5.28}$$

with

$$g(x) = \mathrm{sign}\Big(\sum_{i \in SV} y_i \alpha_i^* \Phi(\mathrm{x}_i)^t \Phi(\mathrm{x}) + b^* \Big) \tag{5.29}$$

and

$$b^* = \frac{1}{n_{SV}} \sum_{i \in SV} \Big(\frac{1}{y_i} - \sum_{i \in SV} y_i \alpha_i^* \Phi(\mathrm{x}_i)^t \Phi(\mathrm{x}_j) \Big). \tag{5.30}$$

In practice, finding a proper non-linear mapping Φ can be a very hard or even untraceable task. But, in the above equations the data only appears as inner product of the transformed space. Then, it is enough to know the inner product $K(\mathrm{x}_i, \mathrm{x}_j) = \Phi(\mathrm{x}_i)^t \Phi(\mathrm{x}_j)$, commonly referred to as a kernel function, which must sustain the convex solution of the optimization problem. To guarantee that, the kernel function must satisfy the Mercer conditions, which claim that $K(x, y) = K(y, x)$ and

$$\int K(x, y) f(x) f(y) \; dxdy \geq 0, \tag{5.31}$$

with $f(x)$ such that

$$\int f(x^2) dx < \infty. \tag{5.32}$$

The most common kernel functions are the polynomial, sigmoidal, and Gaussian (also referred to as the radial basis function, RBF), respectively defined as

$$K(x, y) = (1 + x^t y)^d, \tag{5.33}$$

$$K(x, y) = \tanh(k x^t y - \delta), \tag{5.34}$$

$$K(x, y) = \exp\Big(- \frac{|x - y|^2}{\sigma^2} \Big). \tag{5.35}$$

Perhaps the main drawback associated with the SVM algorithm is its inherent high complexity to determine the hypersurfaces in the system training stage. This is particularly true in the non-linear separation cases, which uses high-dimensional feature representation. In these cases, the SVM needs to solve a quadratic programming (QP) which increases the system complexity as the number of data samples and features grow.

The chosen SVM implementation for this work is from the Matlab Statistics Toolbox. For its basic usage, it requires the matrix (samples vs features) and a vector with the class labels. By default, it uses the linear kernel function, the Karush-Kuhn-Tucker violation is 0, the BoxConstraint (C) is equal to 1. The parameter Kernel Function can be chosen over six different values: linear, quadratic, polyno-

mial, rbf, mlp, @kfun, where the latter is employed when the user wants to specify a different function. The Karush-Kuhn-Tucker violation and the BoxConstraint parameter control the flexibility incorporated into the soft margin and the algorithm convergence accordingly. The iterative algorithm employed to solve the optimization problem has three options: quadratic programming, which is the most accurate and most computationally intensive option, the sequential minimal optimization (SMO) and the least squares algorithms.

5.5 Neural Network

Artificial Neural Network, also known by just Neural Network, is the concept of a algorithm based in the way neurons are connects into other neurons by the use of synapses. Each type of neuron has a response to the pulse that comes from its synapses. And it synapses have a variable conductibility. By this way, the synapses have some resistance for each input for the neuron.

The algorithm of Neural Network works the same way, it threats the neuron like a activation function, it means, each neuron has a transfer function for the input value it receives. And the synapses also have their resistance, but in the form of a weight for each input.

The neurons are divided into layers. The first is the input layer, responsible for connect the input samples to the mesh. The mesh might have many middle layers as it needs, but usually one middle layer are enough, and they are called hidden layer. And the last layer is called output layer, because it receives the responses of the neurons of the hidden layer and send the combined values to one specific output class. It means, the number of neurons in the input layer are defined by the number of input features of the system. Also the number of neurons in the output layer are defined by the number of output classes.

The figure 5.1 shows a neural network composed by two input features and a single decision class.

The training algorithm is how the weights of the synapses are calculated, and it works as the following:

First step is to propagate the sample to the output, in that way save the values of the output of each neuron and call it as $v0_i^l$, where i is the number of neuron in each l layer. The output of each class is $\tilde{y}_i$. The derivative of each function used in the neurons also must be calculated and named v_i^l. The weight of each synapse is called w_{ij}^l, where i is the source neuron of the synapse, j is the destination neuron and the l is the layer of the source neuron.

Then, the back propagation must be done to correct the weights of the synapses. Collect the error of each class using 5.36.

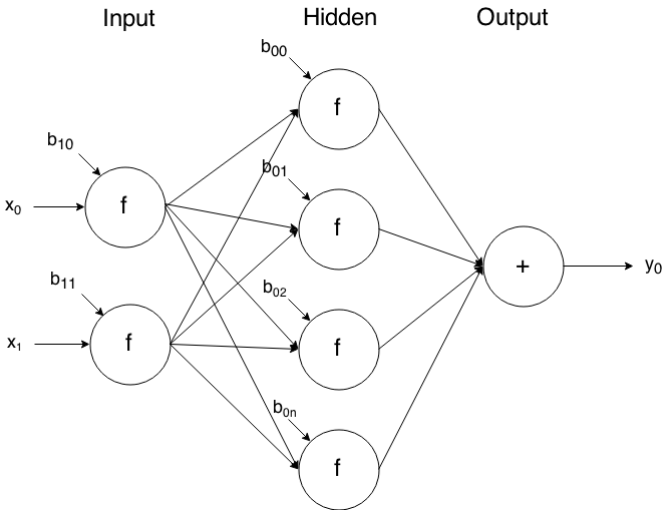


Figure 5.1: Examples of a neural network

$$\epsilon_i = y_i - \tilde{y}_i \tag{5.36}$$

Propagate it back through each neuron using 5.37, this value is the error in the input of the neuron. It must be propagated back to the input features through each layer.

$$\delta_i^l = v_i^l \epsilon_i \tag{5.37}$$

After the error reached the input of the features, the weights of each synapses must be corrected by the equation 5.38, where α is an arbitrary value for step of the gradient.

$$\Delta w_i j^l = 2\alpha \frac{1}{P} \sum_{p=1}^{P} v_j \delta_i|_p \tag{5.38}$$

The Neural Network algorithm can be very used for generic problems, it can use many input features and many input samples and still reach good convergence. There are varieties of configuration, number of layers, types of activation functions that can be used in the problem. The implementation chosen for this problem is the MATLAB Neural Network Toolbox, because of it high efficiency and configuration options.

5.6 Conclusion

This Chapter provided a brief introduction of the three types of algorithms used for the classification of football highlights. Generally speaking, the AdaBoost and random forest are simpler classifiers and more robust to hard conditions of train-

ing. But the support vector machine has a more refined method of training, which may not even converge in hard data conditions. When properly implemented, however, the SVM constitutes a powerful classification tool for very sophisticated data patterns.

Chapter 6

Experiments

6.1 Introduction

This Chapter presents the performance comparison of some classifier types and the best feature configuration.

The Table with the features enumeration is going to be repeated in Table 6.1 for convenience. Because many features are going to be cited by their abbreviations in this Chapter.

Table 6.1: Low-level audio and video features employed in reference [1], repeated for convenience.

Label	Parameters	Abbreviation	Feature	Type
1	Hue Mean 3.18	Hue	Dominant Color	Video
2	Percent of pixels associated to Hue	PHue	Dominant Color	Video
3	Δ 3.25	Δ	Camera Movement	Video
4	θ 3.26	θ	Camera Movement	Video
5	ρ Peak of correlation	ρ	Camera Movement	Video
6	$VAR(\Delta)$	$VAR(\Delta)$	Camera Movement	Video
7	$VAR(\theta)$	$VAR(\theta)$	Camera Movement	Video
8	Pitch 3.2	Pitch	Pitch Estimation	Audio
9	Pitch Diff 3.5	PitchDiff	Pitch Estimation	Audio
10	Pitch Diff Ascending 3.7	PitchDiffA	Pitch Estimation	Audio
11	Short-time Energy 3.3	STE	Energy	Audio
12	Short-time Energy Diff 3.5	STEDiff	Energy	Audio
13	Short-time Energy Diff Ascending 3.7	STEDiffA	Energy	Audio
14	Comb Short-time Energy 3.4	CSTE	Energy	Audio
15	Comb Short-time Energy Diff 3.5	CSTEDiff	Energy	Audio
16	Comb Short-time Energy Diff Ascending 3.7	CSTEDiffA	Energy	Audio

The script of the experiments is going to follow the order below:

- Evaluation of the selected features of the Chapter 4 using AdaBoost;
- Experiment different decimation values using AdaBoost;

- Calculate the operating curve for the best configuration of AdaBoost;
- Test the parameters for configuration of the Random Forest classifier;
- Experiment of a new method for feature selection and comparison of results of Chapter 4;
- Evaluation of the selected features of the Chapter 4 using Random Forest;
- Experiment different decimation values using Random Forest;
- Test the parameters for configuration of the Support Vector Machine and it value for this work.
- Evaluation of the selected features of the Chapter 4 using Neural Network;

6.2 AdaBoost and Features

The AdaBoost results reported in [1] were summarized in Section 3.4. In short, the experiments using automatic validation indicated a best performance of about 92.55% of Precision Rate and 91.74% of Recall Rate for the gentle AdaBoost, using 30 training iterations and an output median filter of size 10.

The initial experiment compares the AdaBoost performance with distinct feature combinations, following the analysis provided in Chapter 4. Results in this case are summarized in Table 6.2 were obtained using the cross-validation with 7 folds. In Table 6.2, *Video* (Hue to $VAR(\theta)$) and *Audio* (Pitch to CSTEDiffA) refer to the whole groups of video and audio features, respectively, from Table 3.1. Analogously, *Sel Video* (Hue, PHue, Δ) and *Sel Audio* (Pitch, PitchDiff, PitchDiffA, STE, STE-Diff) are the respective groups of video and audio features after the feature selection process. *All Sel* combines the *Sel Video* and *Sel Audio* sets. In addition, *PF* refers to the case where 30 frames before and 30 frames after are added to the group of input features and *PF Sel* is the case that uses only 15 frames before and 15 frames after as additional features. *MF* is Median Filter of size 10, because it gives the best results as studied in [1]. The column *Inputs* describes the total of features at the input of the classifier calculated in the form: $(2 * PF + 1) * Features$. All the results in the Table were achieved using the best configuration of AdaBoost used in Chapter 3 described above.

From the results shown in Table 6.2, the main conclusion is that the performance loss is not significant after the feature selection. It can be proved comparing the result of the *All Sel + PF Sel* to the *ALL + PF*, where the configuration with the selected features achieved almost the same Precision and Recall Rates, with only a

Table 6.2: AdaBoost results using different sets of input features.

Features	PR	PR std	RR	RR std	Time(s)	Inputs
Video + MF	61.91%	2.45%	64.75%	11.99%	477	7
Audio + MF	86.56%	2.32%	82.27%	5.81%	616	9
All + MF	85.95%	1.96%	82.50%	4.66%	832	16
Sel Video + MF	61.15%	3.46%	65.75%	10.74%	360	3
Sel Audio + MF	86.37%	2.16%	81.23%	5.26%	474	5
All Sel + MF	85.21%	1.83%	82.02%	4.27%	608	8
Video + PF + MF	66.34%	2.41%	67.23%	9.73%	12183	427
Audio + PF + MF	91.78%	2.65%	90.61%	4.40%	19710	549
ALL + PF + MF	92.55%	1.68%	91.74%	3.19%	32918	976
Sel Video + PF + MF	65.22%	2.52%	67.71%	8.31%	6545	183
Sel Audio + PF + MF	91.93%	2.57%	90.85%	3.70%	11619	305
All Sel + PF + MF	92.30%	1.55%	91.34%	2.90%	18102	488
Video + PF Sel + MF	65.61%	2.45%	66.06%	10.45%	6438	217
Audio + PF Sel + MF	91.71%	2.58%	90.54%	4.37%	11899	279
ALL + PF Sel + MF	92.06%	1.79%	90.94%	2.97%	19170	496
Sel Video + PF Sel + MF	65.56%	2.25%	67.70%	8.81%	3461	93
Sel Audio + PF Sel + MF	91.67%	2.56%	90.54%	3.79%	6842	155
All Sel + PF Sel + MF	92.15%	1.79%	91.22%	2.94%	10621	248

small decrease in comparison to the original configuration employed in [1]. But the training time was more than 3 times faster.

Another experiment evaluated the influence of the time decimation factor for the past and future features used as additional inputs to the classifier, as discussed in Section 4.5. In this case, the maximum interval of 30 past or future frames is kept fixed and only the decimation factor is changed. Then, a decimation factor of 15 frames will mean that the present frame, the 15^th and the 30^th frames from both the past and the future are fed to the classifier, leasing to a total of 40 input features as only the 8 selected feature types are considered in this experiment. Results for several decimation factor are depicted in Figure 6.1. The curves of the Figure 6.1 oscillate but in general the precision rate and the recall rate decreases as the decimation factor increases.

Results in Figure 6.1 indicate that a good performance (just above 90% in both the Precision Rate and Recall Rate) is already reached with a decimation factor of 30, corresponding to only 24 features used by the classifier. As the decimation factor decreases, more features are added to the classifier, whose performance oscillated between 89% and 92%, reaching its peak with a decimator factor of 4. Depending on the application, if there are more flexible conditions, where some loss can be accepted, the decimation factor of 30 could be used. It would cause a reduction to 24 inputs features at the classifier. A reduction of 48 times the initial set of features, proposed by [1].

The different operating points of the classifier can lead to different pairs of recall

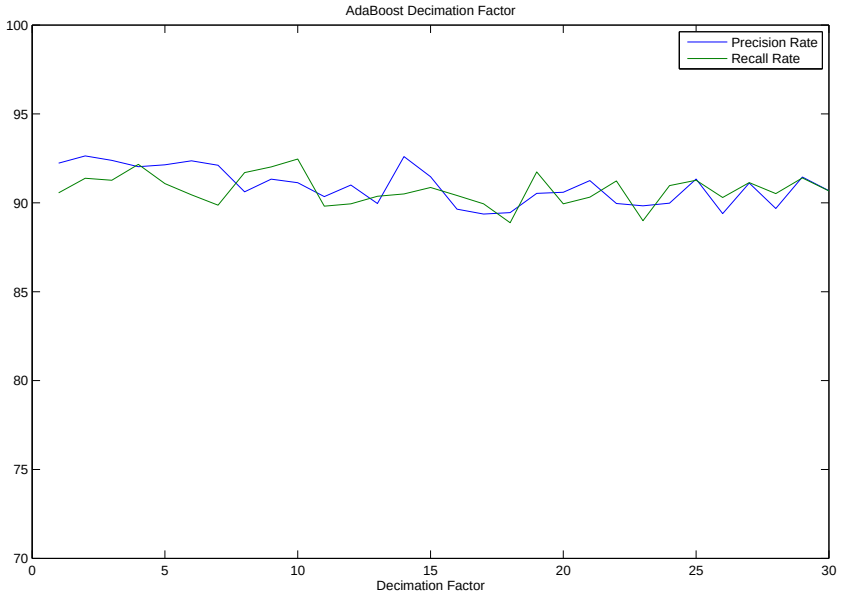


Figure 6.1: AdaBoost Decimation Fator.

rate and precision rate. With this information, each application can choose the best operating point for the classifier, for the constraints of the problem. For example, if a 99% recall rate is needed in the system, it can be achieved by decreasing the precision rate to about 80%. The 6.2 shows the curve of AdaBoost classifier ((1-Recall Rate) x Precision Rate) configured with the following parameters: 30 training iterations, 15 past and future samples, Gentle algorithm, median filter size 10.

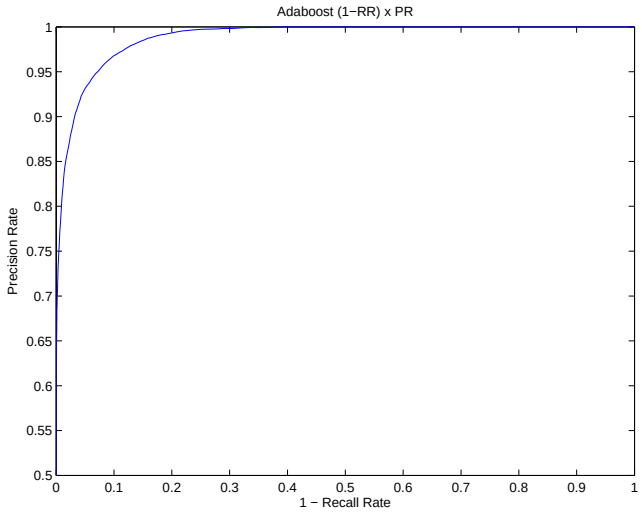


Figure 6.2: Operating points of AdaBoost.

6.3 Random Forest

The TreeBagger function in Matlab implements the Random Forest classifier allowing one to specify the total number of trees. The parameter OOBPred is used to evaluate the effect of each additional tree, as shown in Figure 6.3. In this scenario, one sees no clear effect after about 60 trees, because classification error doesn't fall significantly anymore, as can be seen in the Figure 6.3. Incorporating a safety margin to that lower limit, we have chosen to employ 100 trees in all subsequent random-forest experiments. The figure was generated using the TreeBagger with the the whole set of 16 features and 30 past and future frames, because it would guarantee the minimum trees for the maximum number of features. For lower numbers of features the minimum number of trees will be enough as well.

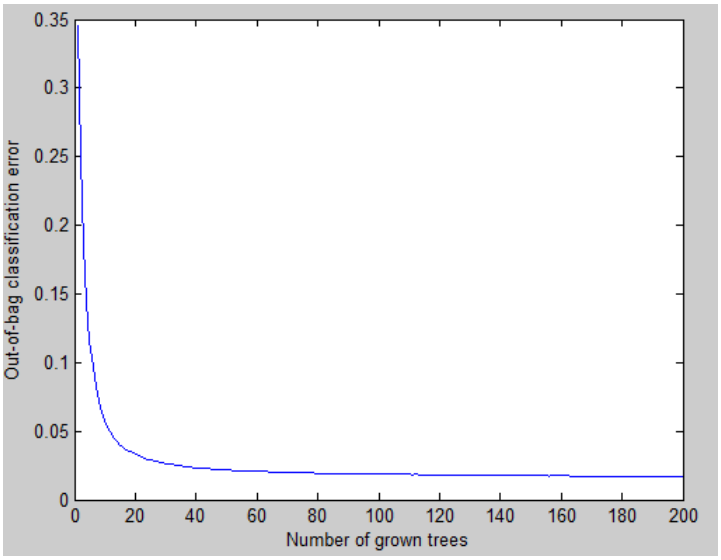


Figure 6.3: Effect of each additional tree (up to 200) in the overall performance of the random forest classifier.

Figure 6.4 shows the performance contribution of each of the 16 original features through the parameter *OOBVarImp* during the classifier training. This analysis shows feature Pitch (see Table 6.1) as the most important feature of the whole set, with PitchDiff, PHue, Hue, Δ and STE coming in sequence. All these features were also selected by the analysis performed in Chapter 4. The two next features pointed in figure 6.4 are the $VAR(\theta)$ and ρ. However, these features were excluded in the Chapter 4. Instead, in the Chapter 4 the features PitchDiffA and STEDiff were selected. Then the differences of both selection relies over the selection of $VAR(\theta)$ and ρ instead PitchDiffA and STEDiff.

Following the feature-importance analysis shown in figure 6.4, the random forest classifier was retrained using the best 8 features. The performance of the random forest classifier with the above selected features was tested using a *k*fold proce-

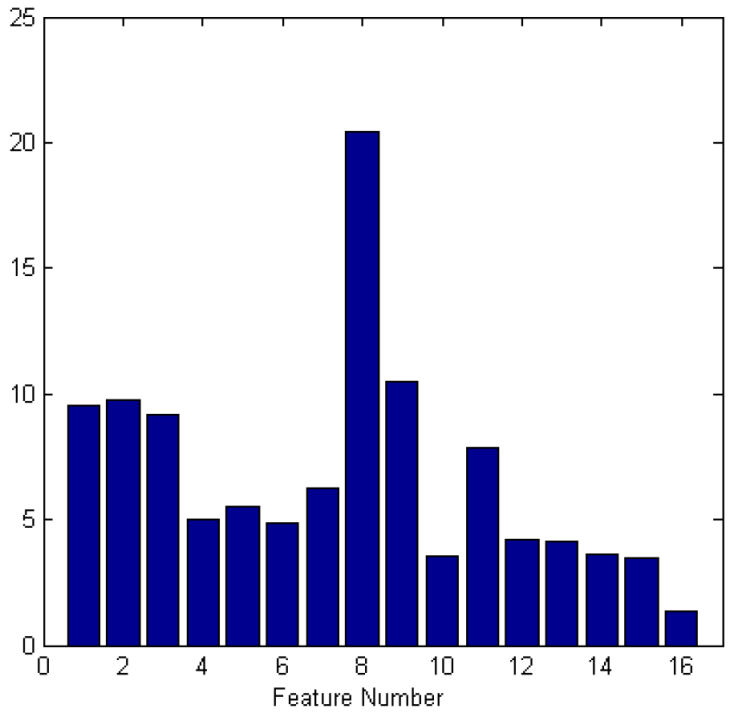


Figure 6.4: Feature selection for the random forest classifier according to the *OOB-VarImp* parameter of the TreeBagger Matlab command.

dure, leading to an average Precision Rate of 91.01% and an average Recall Rate of 90.77% for the testing dataset. The corresponding standard deviations were 3.07% and 3.30% for the Precision Rate and Recall Rate, respectively. When using the features selected in Chapter 4, the cross-validation results were a Precision Rate of 92.14% and a Recall Rate of 92.17% with standard deviations of 2.26% and 3.55%, respectively. This indicates that the TreeBagger feature selection provides good results, although the feature selection performed in Chapter 4 seems to yield slightly better results.

After all this initial tuning, the performance of the random forest classifier was assessed in the same conditions as the AdaBoost classifier in Section 6.2, leading to the results shown in Table 6.3 which follows the same nomenclature as Table 6.2. In general, one notices quite similar results achieved by both the AdaBoost and Random Forest classifier, with a slightly better Precision Rate of the former and a slightly better Recall Rate for the latter.

The decimation-factor experiment was also performed for the random forest, leading to the results shown in Figure 6.5. The curves of the Figure 6.5 are more stable than the curves of the Figure 6.1. But the Precision Rate decreases faster than the Recall Rate, as the decimation factor increases.

Table 6.3: Random forest results using different sets of input features.

Features	PR	PR std	RR	RR std	Time(s)	Inputs
Video + MF	66.10%	13.77%	70.58%	15.26%	16086	7
Audio + MF	88.50%	5.05%	85.75%	7.74%	18382	9
All + MF	87.85%	5.43%	86.61%	6.75%	20592	16
Sel Video + MF	64.34%	14.56%	69.95%	14.21%	7925	3
Sel Audio + MF	88.13%	5.16%	83.95%	7.73%	9370	5
All Sel + MF	86.37%	5.95%	85.34%	6.70%	10348	8
Video + PF + MF	69.85%	12.31%	72.95%	14.08%	35934	427
Audio + PF + MF	93.09%	3.62%	92.42%	5.65%	41269	549
All + PF + MF	92.83%	3.69%	93.37%	4.68%	73622	976
Sel Video + PF + MF	68.29%	13.14%	72.80%	13.58%	22303	183
Sel Audio + PF + MF	93.24%	3.70%	92.35%	5.48%	27008	305
All Sel + PF + MF	92.89%	3.60%	93.05%	4.37%	45907	488
Video + PF Sel + MF	69.80%	12.32%	73.17%	14.41%	52890	217
Audio + PF Sel + MF	93.09%	3.64%	92.37%	5.66%	72831	279
All + PF Sel + MF	92.85%	3.75%	93.46%	4.71%	35353	496
Sel Video + PF Sel + MF	68.58%	13.01%	72.80%	13.66%	19319	93
Sel Audio + PF Sel + MF	93.18%	3.72%	92.26%	5.44%	41518	155
All Sel + PF Sel + MF	93.02%	3.53%	93.05%	4.46%	73456	248

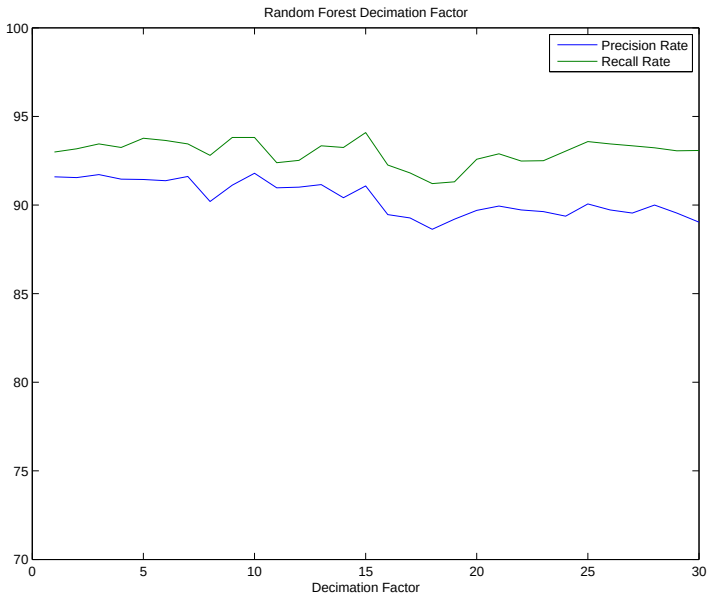


Figure 6.5: Random Forest Decimation Fator.

6.4 Support Vector Machine

The support vector machine (SVM) is also a family of strong classifiers in the machine-learning field. In our problem, however, which deals with large number of input features (particularly when we consider the past and future samples as well), the SVM approach may present convergence problems in its training stage.

Using the Matlab implementation, the *autoscale* parameter was set to *True* to force a feature normalization that leads to zero-mean unitary-variance inputs. The *kernelcachelimits* was set as high as the resources support, which was about 50000; we also considered several setups for the *boxconstraint*, *kernel_function*, *kktviolationlevel*, *method* parameters to optimize the overall classifier performance, as discussed below. However, even for the simplest configuration, with the linear kernel function, the simulation took several hours to evaluate a single game fold. More complex setups required more that a day to be trained without any convergence. Also, even in configurations where the algorithm converges as shown in Table 6.4, depending on the random initialization the classification diverges to a single class and the same configuration needs to be repeated over and over again. Because of it, the evaluation of configurations was a very hard task to be performed. Results of all different configurations are given in Table 6.4, which also shows the attempted setups that did not converge after a week.

Table 6.4: SVM results using different sets of input features.

Features	Inputs	Kernel Function	Error	Recall
All Sel	8	Linear	81.43%	77.41%
All Sel	8	Quadratic	84.10%	77.09%
All Sel	8	Polynomial	84.30%	81.73%
All Sel	8	RBF	no convergence	no convergence
All Features	16	Linear	no convergence	no convergence
All Features	16	Quadratic	no convergence	no convergence
All Features	16	Polynomial	no convergence	no convergence
All Features	16	RBF	no convergence	no convergence
All Sel + PF Sel	248	Linear	no convergence	no convergence
All Sel + PF Sel	248	Quadratic	no convergence	no convergence
All Sel + PF Sel	248	Polynomial	no convergence	no convergence
All Sel + PF Sel	248	RBF	no convergence	no convergence

The best result without using past and future frames was using the polynomial kernel function. Even in this case, however, the result was inferior than the ones for the other tested classifiers in the same configuration, besides the fact that the training process took much longer than for any other tested classifier.

6.5 Neural Network

The Neural Network is the last classifier experimented in this master thesis. Usually, Neural Network are very generic classifier. It can be adapted to many uses with good performance and simplicity. But there are some parameters that need to be configure to each conditions.

The main configuration is the network itself. The number of hidden layers and the number of neurons for each layer must be set. Rarely a problem is going to need a network with more than one hidden layer. Even though, a two hidden layer was experimented in this project, but the results weren't satisfactory. Mostly because as the number of layer grows, harder becomes the convergence of the algorithm.

The best configuration of the number of neurons on each layer was with 10 neurons in the hidden layer, because for higher values the precision rate and the recall rate grows very slowly and the memory resources and time spent grows very fast, turning the classification unfeasible with high number of inputs. The input layer was set as the same number of inputs in the classifier. And also, it was set one neuron in the output layer because there are only one class to be trained in this system.

Another important parameter is the activation function. As this problem is using the neural network for classification and not for regression, the best activation function is the *tansig* for the hidden layer and the output layer. The back-propagation method used was *trainlm* because its fast training step. The learning function applied was the *learngdm* and *mse* as performance function.

The table 6.5 shows the results for the best configuration of the neural network applied for the database.

The results obtained with Neural Networks were almost as good as the ones obtained with AdaBoost and Random Forest. Decreasing about 2% and 3% the Precision Rate and Recall Rate, respectively, comparing to the others. But the memory resources were also a important matter, because it makes necessary to use much memory resources to reach good results with a short training time spent, reaching almost $27GB$ of the RAM of the equipment.

6.6 Summarization

To evaluate the system under real conditions, the best performance configuration was tested over the entire matches. Using the entire matches, the number of highlights are much lower than the number of non-highlight moments. This fact causes a decrease of the Precision Rate, but now the Summarization Rate can be calculated. The Table 6.6 show all the matches including two matches with the untrained nar-

Table 6.5: Neural Network results using different sets of input features.

Features	PR	PR std	RR	RR std	Time(s)	Inputs
Video + MF	6719%	1347%	6591%	1710%	2530	7
Audio + MF	8847%	502%	8471%	807%	2712	9
All + MF	8684%	616%	8592%	795%	4011	16
Sel Video + MF	6632%	1393%	6697%	1642%	3867	3
Sel Audio + MF	8613%	770%	8408%	897%	1325	5
All Sel + MF	8700%	559%	8316%	819%	1831	8
Video + PF + MF	7128%	1011%	6877%	1302%	13768	427
Audio + PF + MF	9082%	490%	8902%	771%	19234	549
ALL + PF + MF	8905%	668%	8692%	941%	72250	976
Sel Video + PF + MF	7061%	1207%	6932%	1431%	7117	183
Sel Audio + PF + MF	9135%	455%	8968%	733%	18660	305
All Sel + PF + MF	9102%	462%	8923%	769%	43643	488
Video + PF Sel + MF	7074%	1202%	6962%	1487%	10133	217
Audio + PF Sel + MF	9107%	452%	8991%	641%	10903	279
ALL + PF Sel + MF	8963%	620%	8951%	787%	22881	496
Sel Video + PF Sel + MF	7043%	1212%	7032%	1423%	5540	93
Sel Audio + PF Sel + MF	9142%	452%	8996%	742%	5856	155
All Sel + PF Sel + MF	9120%	429%	9026%	652%	11051	248

rator(matches 7 and 28). The Summarization Rate is represented in the Table 6.6 by *SR(Time)* (which provides the values in [minutes:seconds] and *SR(%)* (which provides the result in percent).

The conclusion of the Table 6.6 is that the Recall Rate still is very high and the Summarization Rate is great. In the average the reduction is almost 8 times. With this results, the need of manual intervention is at least 7 times less. It means, one person can handle 7 matches at once.

6.7 Conclusion

This Chapter presented the highlight-detection results achieved by three different classifier families. In such analysis, we considered the use of several feature sets as classifier inputs, reducing the number of features from the original 16 employed in [1] to only 8, without any noticeable performance degradation. The use of past and future frames was also investigated, leading to a decrease in the number of total input frames, which correspond to a higher value of the time decimation factor.

The experiments confirm the methods for feature selection applied in Chapter 4 were successful. It was verified that the AdaBoost, Random Forest and Neural Network classifiers achieved quite similar results, what can be explained by the close relationship between the two intrinsic principles of these two classifier families. In addition, the performance of the SVM classifier was assessed leading to much poorer results in comparison to the AdaBoost, Random Forest and Neural Network

Table 6.6: Summarization results using best configuration of Randon Forest.

Match	PR	RR	SR(time)	SR(%)
1	16.04%	100.00%	13:16	14.12%
2	12.45%	100.00%	14:30	15.43%
3	12.81%	100.00%	10:55	11.62%
4	13.96%	100.00%	13:53	14.77%
5	9.09%	100.00%	18:14	19.39%
6	13.71%	100.00%	12:18	13.09%
7	9.31%	86.51%	11:16	11.98%
8	14.22%	96.53%	11:10	11.88%
9	29.13%	100.00%	4:00	4.26%
10	17.53%	100.00%	13:54	14.79%
11	26.64%	100.00%	3:11	3.38%
12	15.40%	100.00%	12:26	13.23%
13	18.35%	100.00%	10:54	11.59%
14	8.29%	87.81%	7:57	8.45%
15	20.02%	100.00%	7:50	8.33%
16	17.36%	100.00%	12:58	13.80%
17	31.85%	100.00%	3:52	4.11%
18	10.16%	100.00%	14:32	15.46%
19	16.98%	100.00%	9:19	9.92%
20	15.84%	100.00%	12:47	13.60%
21	15.23%	94.32%	11:44	12.49%
22	19.38%	100.00%	12:35	13.38%
23	8.24%	97.51%	20:54	22.24%
24	20.61%	100.00%	8:26	8.97%
25	10.00%	100.00%	10:28	11.13%
26	7.78%	98.84%	17:21	18.46%
27	5.16%	100.00%	21:54	23.29%
28	9.76%	100.00%	16:50	17.91%
29	16.29%	100.00%	10:01	10.65%
30	8.30%	100.00%	21:28	22.83%
Average	14.99%	98.71%	12:21	13.15%

cases, probably due to the large number of features and data being considered in the experiments.

Chapter 7

Conclusion

The main goal of this master thesis was to improve and optimize the football highlight-detection system developed in [1]. Such investigation included a feature-selection analysis and the test of different classification families of algorithms.

The first step in this direction was to perform a wide research in the related literature of video annotation, in particular for sport events, as given in Chapter 2. In such survey, it was observed that most current works deal with the semantic annotation of the whole video, and not only to the highlight detection, which is the focus of our work. In general, in the semantic analysis, the event of interest is identified a prior using simple rule-of-thumb methods such as replay or slow-motion track or inserted graphic interpretation, such as the ones used to indicate yellow/red cards, scored goals, player substitutions etc. Such simple methods, however, are quite production-dependent, indicating the need of more general algorithms to perform the highlight detection in a robust manner.

In Chapter 3, a summary was presented for the work developed in [1], which is the starting point of the practical developments implemented in this dissertation. The description included all the features extracted by the previous method, the annotated database, and the results achieved by the original system. Such review facilitates comparison of the techniques proposed here with the original ones, providing a self-contained characteristic to the present dissertation.

As seen in Chapter 3, the original highlight-detection system employed a total of 16 distinct types of features. The system also combines the information from the present video/audio frame with the features from the previous 30 and subsequent 30 frames, leading to a total of $16 \times 61 = 976$ inputs for each frame. Chapter 4 then describes a feature-importance analysis which indicated that only 8 out of the original 16 features carry most of the underlying information for the highlight detection process. The techniques used were based on the cross-correlation between any 2 features and between any feature with the desired classifier output. An initial study on the possible decimation (by, at first, only a rate of 2) of the past and future

frames indicated that such additional information could also be greatly removed. Thank to all these experiments, it was concluded that $8 \times 31 = 248$ could provide the same performance as the original system, at a much lower computational effort.

Chapter 5 provided a brief introduction to the families of algorithms included in our analysis, namely the AdaBoost, the Random Forest, the Support Vector Machine and Neural Network. The basic theory of these methods was given along with their main characteristics and setup configuration (in Matlab).

Chapter 6 presents the experimental results achieved by the proposed system variations, considering different types of feature sets (as discussed in Chapter 4) and classifier families (as discussed in Chapter 5). In the latter analysis, the influences of some parameters were also investigated. Table 7.1 summarizes the results achieved in the present work and allow one to compare them with the results from [1]. The [1] configuration was with the whole set of 16 features, using the samples of 61 frames and the Gentle AdaBoost classifier. The best configuration of this work was concluded to be with the 8 selected features, using the samples of 31 frames and the TreeBagger algorithm for Random Forest. Both were achieved from a cross-validation method. From such analysis, it is safe to conclude that the feature selection performed here allowed a great simplification of the system complexity without deteriorating the system performance.

Table 7.1: Performance comparison of simplified and original ([1]) systems.

Works	Features	Precision Rate(%)		Recall Rate(%)	
	Number	Average	Standard Deviation	Average	Standard Deviation
Vasconcelos [1]	976	92.55	1.68	91.74	3.19
This Work	248	93.02	3.53	93.05	4.46

The Chapter 6 also could conclude that if the system could accept more flexible results with some few losses in the Precision Rate and the Recall Rate, the configuration with the decimator factor of 30 would have a reduction of 40 times the initial number of input features at the classifier.

7.1 Future Works

Despite the improvements achieved in our development, the current highlight-detection system may still benefit from additional

- Adaptation from low-level features to mid-level features. Aiming more semantic meaning of the problem. It might simplify the task of the final classifier. Also it can be used classify the semantic each event;

- New semantic features, like detection of the penalty box. The penalty box is a good example because every goal attempt needs to have the camera pointed to the penalty box, this feature could efficiently decrease the false positives and generate a higher summarization of the match;

- Segmentation of the events. Because it could simplify the treatment of the past and future samples. Using this concept the samples of the past and the future could be earlier joined into a single value per feature. Reducing the complexity of the classifier.

Most importantly, perhaps the main contribution of the present work is to understand the behavior of some standard machine-learning techniques in the problem of football automatic highlight detection, and open new venues for further investigations on the subject.

Bibliography

[1] L. G. L. B. M. Vasconcelos, "Sumarização automática em melhores momentos de transmissões televisivas de futebol," Master's thesis, Universidade Federal do Rio de Janeiro, 2011.

[2] M.-S. Hosseini and A.-M. Eftekhari-Moghadam, "Fuzzy rule-based reasoning approach for event detection and annotation of broadcast soccer video," *Applied Soft Computing*, vol. 13, no. 2, pp. 846–866, Feb 2013. [Online]. Available: http://dx.doi.org/10.1016/j.asoc.2012.10.007

[3] F. Coldefy and P. Bouthemy, "Unsupervised soccer video abstraction based on pitch, dominant color and camera motion analysis," *Proceeding MULTIMEDIA '00 Proceedings of the eighth ACM international conference on Multimedia*, 2004.

[4] R. Leonardi, P. Migliorati, and M. Prandini, "Semantic indexing of soccer audio-visual sequences: a multi-modal approach based on controlled markov chains," *IEEE Trans. Circuits Syst. Video Technol.*, 2004.

[5] D. Sadlier and N. O'Connor, "Event detection in field sports video using audio-visual features and a support vector machine," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 10, p. 9, 2005. [Online]. Available: http://dx.doi.org/10.1109/TCSVT.2005.854237

[6] M. Tavassolipour, M. Karimian, and S. Kasaei, "Event detection and summarization in soccer videos using Bayesian network and copula," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 2, p. 13, 2014. [Online]. Available: http://dx.doi.org/10.1109/TCSVT.2013.2243640

[7] B. Li and M. I. Sezan, "Event detection and summarization in sports video," *Content-Based Access of Image and Video Libraries, 2001. (CBAIVL 2001). IEEE Workshop*, 2001.

[8] D. Tran, J. Yuan, and D. Forsyth, "Video event detection: From subvolume localization to spatiotemporal path search," *IEEE Trans. Pattern Anal.*

Mach. Intell., vol. 36, no. 2, pp. 404–416, 2014. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2013.137

[9] H.-S. Chen and W.-J. Tsai, "A framework for video event classification by modeling temporal context of multimodal features using HMM," *Journal of Visual Communication and Image Representation*, vol. 25, no. 2, pp. 285–295, Feb 2014. [Online]. Available: http://dx.doi.org/10.1016/j.jvcir.2013.12.001

[10] L.-Y. Duan, M. Xu, T.-S. Chua, Q. Tian, and C.-S. Xu, "A mid-level representation framework for semantic sports video analysis," *ACM Multimedia*, 2003.

[11] L.-Y. Duan, M. Xu, Q. Tian, C.-S. Xu, and J. Jin, "A unified framework for semantic shot classification in sports video," *IEEE Transactions on Multimedia*, vol. 7, no. 6, pp. 1066–1083, 2005. [Online]. Available: http://dx.doi.org/10.1109/TMM.2005.858395

[12] C. Liu, Q. Huang, S. Jiang, L. Xing, Q. Ye, and W. Gao, "A framework for flexible summarization of racquet sports video using multiple modalities," *Computer Vision and Image Understanding*, vol. 113, no. 3, pp. 415–424, Mar 2009. [Online]. Available: http://dx.doi.org/10.1016/j.cviu.2008.08.002

[13] D. W. Tjondronegoro and Y.-P. P. Chen, "Knowledge-discounted event detection in sports video," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 40, no. 5, pp. 1009–1024, 2010. [Online]. Available: http://dx.doi.org/10.1109/TSMCA.2010.2046729

[14] H. S. Lexing Xie, Shih-Fu Chang Ajay Divakaran, "Structure analysis of soccer video with hidden markov models," *Pattern Recognition Letters - Video computing*, vol. 25, pp. 767 – 775, 2004.

[15] C. Xu, J. Wang, K. Wan, Y. Li, and L. Duan, "Live sports event detection based on broadcast video and web-casting text," *Proceedings of the 14th annual ACM international conference on Multimedia*, 2006.

[16] D. S. R. Rajkumar and Jayanth.A.S, "Survey on motion vector filtering and object segmentation methods in compressed domain," *International Journal of Advancements in Technology*, vol. 2, No 2, 2011.

[17] A. M. T. Ahmet Ekin and R. Mehrotra, "Automatic soccer video analysis and summarization," *Image Processing, IEEE Transactions*, vol. 12, pp. 796 – 807, 2003.

[18] Y.-P. P. Dian Tjondronegoro and C. B. Pham, "The power of play-break for automatic detection and browsing of self-consumable sport video highlights," *Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval*, 2004.

[19] R. Cabasson, A. Divakaran, and M. E. R. Laboratories, "Automatic extraction of soccer video highlights using a combination of motion and audio features," *Conference: Storage and Retrieval for Media Databases*, 2003.

[20] A. Hanjalrc, "Generic approach to highlights extraction from a sport video," *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on (Volume:1)*, no. cuts, 2010.

[21] S. Dagtas and M. Abdel-Mottaleb, "Multimodal detection of highlights for multimedia content," *Multimedia Systems*, vol. 9, no. 6, p. 586 593, Jun 2004.

[22] N. Rea, R. Dahyot, and A. Kokaram, "Classification and representation of semantic content in broadcast tennis videos," *Image Processing, 2005. ICIP 2005. IEEE International Conference*, 2004.

[23] Z. Xiong, X. S. Zhou, Q. Tian, Y. Rui, and T. S. Huang, "Semantic retrieval of video- review of research on video retrieval in meetings, movies and broadcast news, and sports," *SIGNAL PROCESSING MAGAZINE*, 2009.

[24] D. Gerhard, "Pitch extraction and fundamental frequency: History and current techniques," University of Regina, Tech. Rep. 2003-06, 2003.

[25] T. Tolonen and M. Karjalainen, "A computationally and efficient multipitch and analysis model," *IEEE TRANSACTIONS ON SPEECH AND AUDIO PROCESSING, VOL. 8, NO. 6, NOVEMBER*, 2000.

[26] D. Rocchesso, *Introduction to Sound Processing.* David Rocchesso, 2003.

[27] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed. Prentice-Hall, Inc, 2007.

[28] D. Pearson, *Image Processing (Essex Series in Telecommunication and Information Systems).* Mcgraw-Hill, 1991.

[29] P. Diniz, S. Netto, and E. D. Silva, *Digital Signal Processing: System Analysis and Design.* New York, NY, USA, Cambridge University Press, 2002.

[30] A. Vezhnevets. (2011, June) Gml adaboost matlab toolbox. [Online]. Available: http://graphics.cs.msu.ru/en/science/research/machinelearning/

[31] R. O. Duda, P. E. Hart, and D. G. Stork, "Pattern classification," *Wiley-Interscience*, nov. 2001.

[32] *Improved Boosting Algorithms Using Confidence-rated Predictions*, 1997.

[33] Y. Freund and R. E. Schapire, "A short introduction to boosting," in *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 1999, pp. 1401–1406.

[34] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *Annals of Statistics*, vol. 28, p. 2000, 1998.

[35] A. Vezhnevets and V. Vezhnevets, "Modest adaboost-teaching adaboost to generalize better," in *Graphicon*, 2005.

[36] L. Breiman, "Bagging predictors," University of California, Tech. Rep. 421, 1994.

[37] T. G. Deitterich, "An experimental comparison of three methods for constructing ensembles of decision trees," in *Machine Learning*, 1999.

[38] L. Breiman, "Random forest," *Machine Learning*, 2001.

[39] V. N. Vapnik, *Statistical Learning Theory*, S. Haykin, Ed. John Wiley & Sons, INC, 1998.

[40] G. M. Araujo, E. A. B. D. Silva, and A. G. Ciancio, "Rastreamento robusto de olhos usando consistência temporal e restrições geométricas," *XXXI Simpósio Brasileiro de Telecomunicações*, 2013.