Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

MASTER THESIS

TITLE: Automatic event detection for tennis broadcasting

MASTER DEGREE: Master in Science in Telecommunication Engineering & Management

AUTHOR: Javier Enebral González

DIRECTOR: Francesc Tarrés Ruiz

DATE: July 8th, 2011

Title: Automatic event detection for tennis broadcasting

Author: Javier Enebral González

Director: Francesc Tarrés Ruiz

Date: July 8th, 2011

Overview

Within the image digital processing framework, this thesis is situated in the automatic content indexation field. Specifically during the project, different methods and techniques will be developed in order to achieve event detection for broadcasting tennis videos.

Audiovisual indexation consists in the generation of descriptive tags based on the existing audiovisual data. All these tags are used to search the desired material in an efficient way. Televisions and other entities are looking for the improvement in operational efficiency. For this reason, the content indexation is a key factor and it is supposed to be automatic in the near future

In some sports as football where the demand of video highlights is high, a big amount of economic and human resources are available to do this task. However, in other sports like tennis where the financial resources are not so high, automation becomes an important issue.

The report starts describing the method to separate the useful frames. These frames are those that provide information, in other words, they appear when the camera is focused on the tennis court. The following step is looking for the court and players location, with the aim of translating their perspective coordinates into the real world coordinates.

Depending on the distance travelled by the players, their movements and the duration of the shot, the implemented algorithm will be able to distinguish between different kind of shots, as aces, baseline rallies or net approaches. The code has been developed in Matlab programming language.

The program has been tested with three tennis videos belonging to different surfaces: hard court, grass and clay. The results in terms of event detection and computing times will be detailed at the end of the report.

CONTENTS

INDEX OF FIGURES

INTRODUCTION

This project is situated within the wide framework of digital image processing. Specifically, during this report different methods and techniques will be studied and developed in order to achieve event detection for broadcasting tennis videos. The detection of events will allow the automatic content indexation of full tennis matches.

The content indexation of sport videos, not only in the particular case of tennis, but also in any sport, is useful to get an easy and fast access to a database containing particular fragments of interest, the highlights of a game or something similar. By definition, the audiovisual content indexation consists in the generation of descriptive tags based on the existing audiovisual data. All these tags, usually called metadata, are used to search the desired material in an efficient way.

There are different kinds of metadata that can be extracted from the audiovisual material. These differences come from the type of descriptors that have been used. The low level descriptors are referred to the most basic information, as could be features as colour, texture, shape, movement, etc. They can be calculated by computing programs easily.

The high level descriptors are the most important when indexing any content, but at the same time they are the most complicated to compute. The main reason is that some human intelligence is usually needed to develop them properly. These descriptors are able to describe actions, time or space. Some examples of that could be the description of a play or the tactic analysis of a match.

At present, the non existence of specific software able to do the high level content indexation causes that this task has to be done manually by individuals. However, the constant propagation of sports game broadcasting which produces the growth of sports video material, has caused the need of accessing to the content anytime and anywhere. Thus, the automation in sport video classification seems to be essential.

The television and other audiovisual entities are looking for the improvement in operational efficiency. For this reason, the content indexation is a key factor and it is supposed to become automatic in the near future. As commented before, the execution of all these tasks is carried out in a manual way, therefore it means a big expense of money and time.

In some sports as football where the demand of video highlights is high, a big amount of economic and human resources are addressed to the content indexation. Nevertheless, in other sports like tennis, the financial resources are not so high. Therefore, the automation of content indexation becomes a key factor in this kind of sports.

For this reason, the development of this thesis has the aim of providing a specific solution for automatic content indexation in the particular case of tennis. Although the final proposed solution or algorithm only can be applied in this sport, some of the techniques that will be explained to get the basic low level descriptors could be used in other sports. As examples of these descriptors could be the lines position, the dominant colour of the court or players situation.

Other higher level descriptors as the time duration of a shot, the distance covered by the player or the applause timing among different points have to be interpreted taking into account the specific tennis guidelines.

Although the implemented algorithm will be a single solution for a specific sport, the main objective of the project is giving a global solution for all kind of tennis broadcasting videos. That is to say, the software is intended to work properly regardless of the court surface or the tournament from where the video was recorded.

Because of that, during the development and testing of the code, three different surfaces have been used: the hard court of United States Open, the British grass of Wimbledon and finally the clay of Roland Garros French Open. The tests will need lots of computational calculations, resulting in spending a lot of time. Taking into account that they will be carried out by ordinary computers, several points from each tournament have been extracted from its original full video match, with the aim of making the tests in a faster and feasible way. As a consequence of this segmentation, a database of nearly eighty tennis points has been used during the project. The code has been developed in Matlab programming language.

The first chapter of the report is aimed to introduce the reader into the audiovisual content description world, defining the MPEG-7 standard and describing its main properties.

The next chapter is starting to develop the tennis topic. It tries to solve the problem of separating the video frames that will provide useful information and the frames that will not. In tennis broadcasting, the camera always switches to court view just before one of the players is ready to serve, and this view remains here until the shot ends. Therefore, a good way to segment the video into plays is looking at view changes of the camera.

The frames located between the useful ones, usually show players close-ups or panoramic views of the audience which will not give meaningful information, so they have to be discarded. The method implemented to differentiate the changes of the camera is related with comparing histograms.

Once the playing frames are detected, the next step is getting the situation of the court, so the following chapter is devoted to that. For calculating the position, it is necessary to find the lines which form its geometry. The technique used to extract the straight lines from the video is the Hough transform. Previously to apply the transform in a frame, it is mandatory to look for some candidate points that accomplish a few features.

One of these features is the colour of the pixels. As it is known that they are supposed to be white, those pixels which do not reach a minimum intensity in the gray scale image will be discarded. Besides these constraints, the resultant white points are filtered to check if they have straight line geometry. The pixels which pass both constraints become in candidate points, so the Hough transform will be calculated on them.

After obtaining the several candidate lines which appear in a frame, the software has to be able to discard the false detections, because there is an expected pattern that they may satisfy. The main criteria to do it are the location, length and slope of the straight lines. If they have the expected features, they will be considered as lines belonging to the court.

Although at this moment the most part of the lines are detected, what is really important is to know the position of the outside four corners. Its positions will be required later when calculating the relative location among the player and them. For this reason, a final validation is done to check if the four corners found are the correct ones. To do it, the validation takes into account if they verify the geometrical features of a rectangle seen in perspective, in other words, the tennis court seen from the camera. The part of the algorithm dedicated to get the situation of the court ends here.

The fourth chapter describes the methods which have been developed to detect the players. Before starting a shot, the position of them is situated in a well known area, therefore it is only necessary to look for there and locate the position through the use of vertical and horizontal filters. Once the players are detected in the first frame, the search in the following ones is done by tracking the last position in the previous frame. The tracking of the player is obtained by computing the difference between consecutive frames. The main reason for calculating on this way is reducing the number of mathematical operations and to improve the code efficiency.

The next section of the report is aimed to explain the coordinate conversion among the data acquired before (that is calculated from the view on perspective of the camera) and the two dimensional plane. This conversion will be useful to extract the high level descriptors of the video.

The last descriptive part defines the descriptors which will allow determining the kind of shot that the software is processing. Examples of that are the aces, baseline rallies or net approaches. Finally, some experimental results are shown for giving an idea about how the whole algorithm works, followed by some conclusions that can be gathered from the work. The following flow diagram shows the overall structure of the algorithm (**Fig. I.1**).

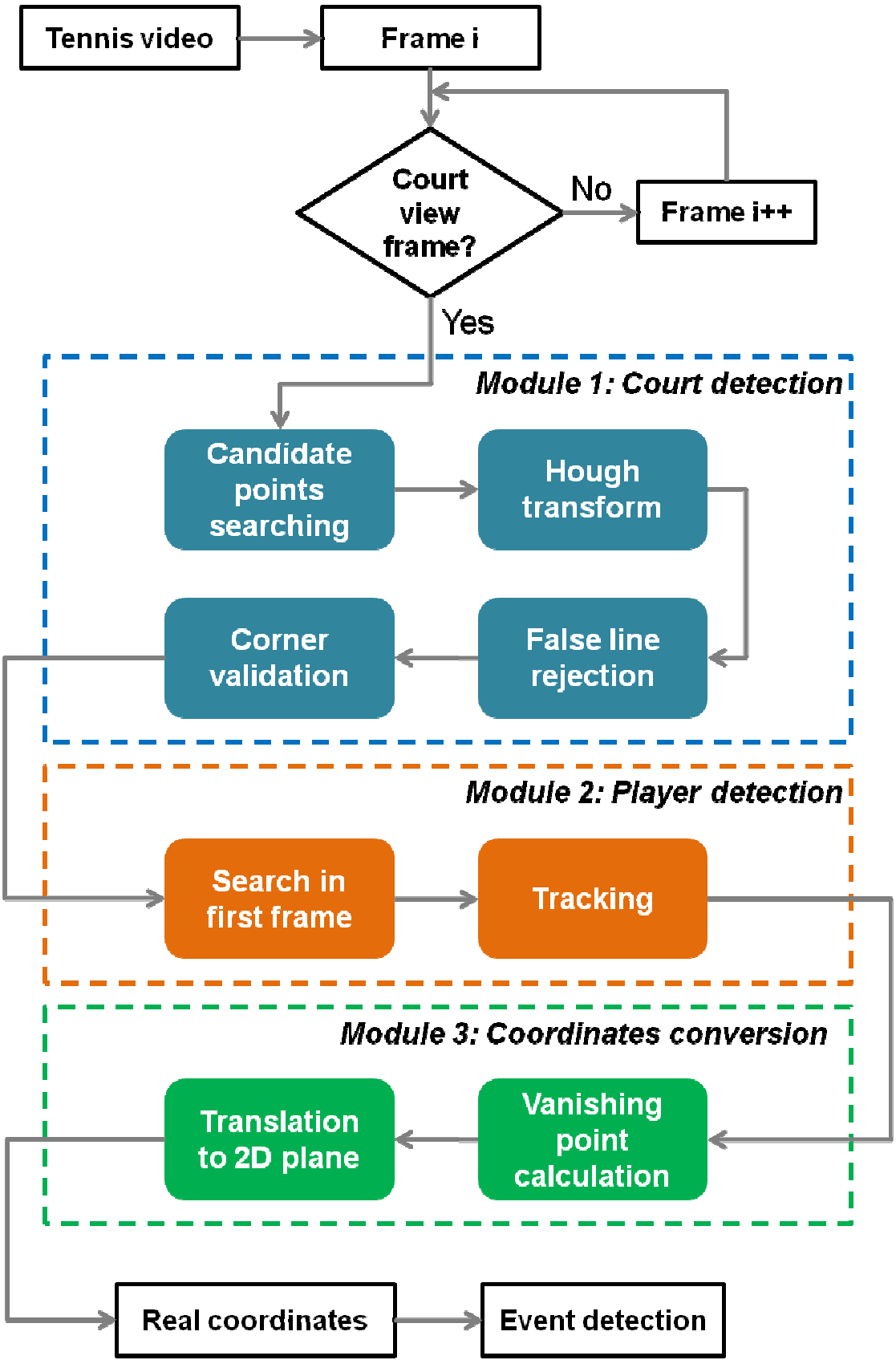

Fig. I.1: Global flow diagram of the algorithm.

CHAPTER 1. STATE OF THE ART IN AUTOMATIC VIDEO INDEXATION

The evolution of sports game broadcasting has lead to a big growth of sports video content. Consequently, it has caused the increasing need for accessing to sports video content anytime, anywhere and using a variety of receiving electronic devices **[1]**.

This enrichment in sports video content has also resulted in much difficulty and complexity for the users to access or even edit some fragments of them, due to the huge amount of existing videos nowadays.

When trying to access into a large video database, the ability of analyzing intelligently the content allows looking for information efficiently. Moreover, other issues as video enhancement and customization can be done in an easier way. All these issues are expected by the content providers and the final viewers. Several approaches have been deployed with this objective, making an automatic or semi-automatic content video analysis.

1.1 Sports video analysis framework

The main objective of the sports video analysis is extracting semantic information in order to adapt the content into different applications. This information is used to query and browse through the sport video, whereas the low-level features acquired by automatic processing cannot provide semantics directly.

The scheme displayed in **Fig. 1.1** describes the relation among low-level features and the high-level semantics. The video analysis must start from the low-level characteristics extraction, composed by visual, audio and text. There are mid-level properties which bridge between low and high-level features, due to the difficulty to directly mapping both.

Employing the domain knowledge and using some machine learning approaches, the mid-level models become essential for describing the corresponding high-level features for several applications. Examples of these applications could be the event or highlights detection, tactic analysis, content insertion, etc.

In the tennis case, low-level features can be the surface color or the lines detection, whereas mid-level ones could be the relative position between the player and the court or the ball trajectory. Examples of high-level features are the detection of specific events, as aces, double faults, net approaches, etc.

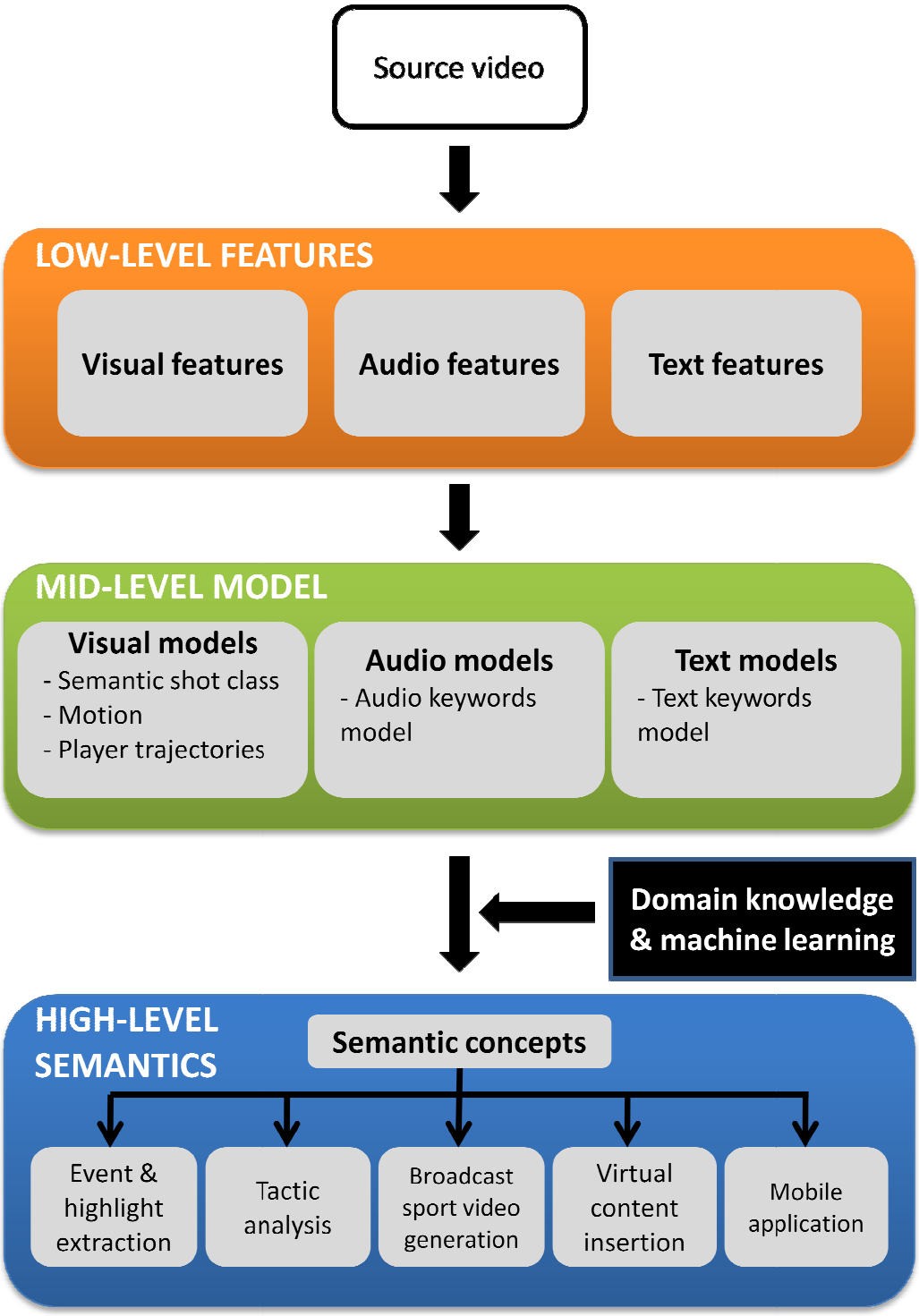

Fig. 1.1: Audiovisual feature levels.

This project is focused on the visual model part of the scheme. Sports videos are formed by semantic shots, which appear repeatedly with similar actions in the same shot class. The structure of the sports video makes it feasible to classify the events according to the semantic meanings. A mid-level feature can describe a part or a full semantic concept.

1.2 MPEG-7 standard

1.2.1 Definition

The MPEG-7 standard tries to answer the need for accessing to sports video content, which is constantly growing. It is an ISO/IEC standard developed by MPEG (Moving Picture Experts Group). The standard is formally named "Multimedia Content Description Interface", and it is devoted to describe the multimedia content data that supports some degree of interpretation of the information meaning. MPEG-7 is not aimed at any one application in particular. The standardized elements support as wide range of applications as possible (see **[2]** and **[3]**).

MPEG-7 offers an exhaustive set of audiovisual description tools to create descriptions, which will form the basis for several applications. It enables the need of effective and efficient access for searching, filtering and browsing while working with multimedia content.

Audiovisual data content that has MPEG-7 descriptions associated may include still pictures, graphics, 3D models, audio, speech, video, and composition information about how these elements are combined in the multimedia scenario. However, these descriptions do not depend on how the content has been coded or stored.

MPEG-7 addresses many different applications in many different environments, therefore it appears the need of providing a flexible and extensible framework for describing audiovisual data. Therefore, the standard does not define a monolithic system for content description but rather a set of methods and tools for the different viewpoints of the description of audiovisual content. The standard defines the XML as the language for doing the textual representation of content description. This selection has been done in order to facilitate interoperability with other metadata standards in the future.

1.2.1.1 Scope of the standard

Fig. 1.2 shows a block diagram of a possible MPEG-7 processing chain. This chain includes the video feature extraction, the description itself and the search engine, which is the final application. The scope of the standard is focused on the feature descriptions.

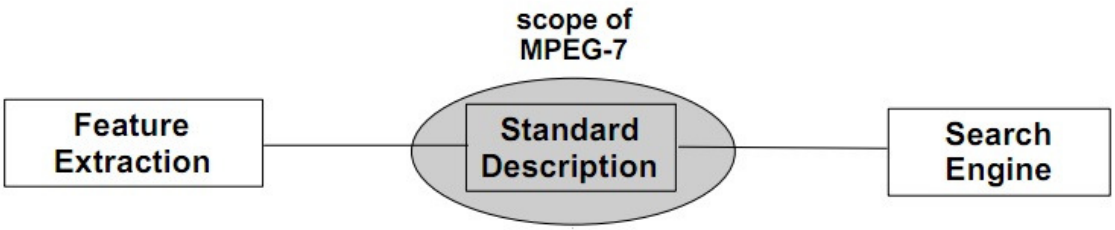

Fig. 1. 2: Scope of MPEG-7 standard.

Although the automatic extraction of features is extremely useful, it is not always possible. As commented before, the higher the level of abstraction, the more difficult automatic extraction is. In any case, these issues are not inside the scope of the standard. The search engines or any other program that can use the descriptions are not specified either within the scope of MPEG-7.

1.2.2 Elements

1.2.2.1 Descriptors (D)

The Descriptors define the syntax and semantics for the representation of features.

1.2.2.2 Description Schemes (DS)

The Description Schemes specify the structure and semantics of the relationships between their components. These components may be Ds or DSs.

1.2.2.3 Description Definition Language (DDL)

The Description Definition Language allows the definition of the MPEG-7 description tools, both Descriptors and Description Schemes, providing the means for structuring the Ds into DSs. The DDL also allows the extension for specific applications of particular DSs. It is based on XML.

1.2.2.4 Systems tools

The system tools support coding, multiplexing and synchronization of descriptions. The relationships between the elements which form the MPEG-7 standard are detailed in **Fig. 1.3**.

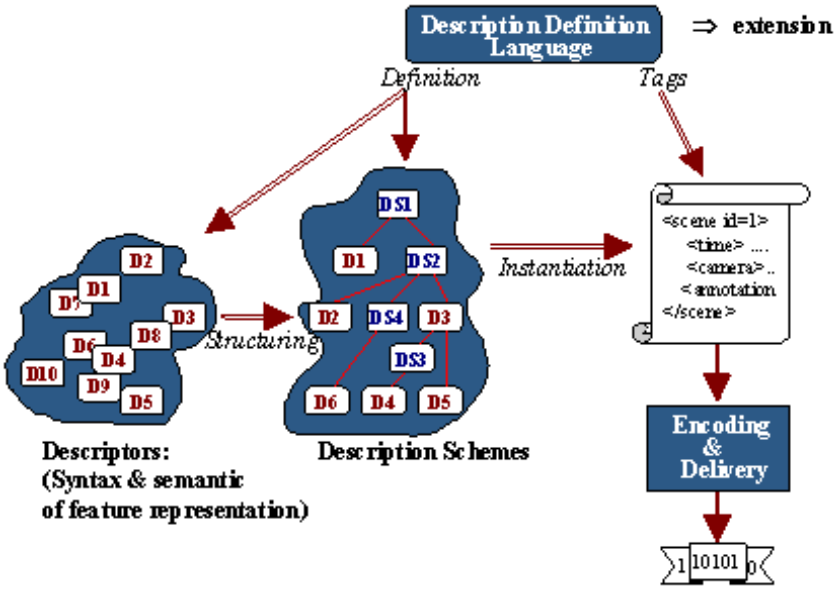

Fig. 1. 3: MPEG-7 elements.

1.2.3 Functionalities

The following sections contain the main functionalities offered by the different parts of the MPEG-7 standard.

1.2.3.1 Systems

It includes the definition of a binary description format for efficient transport and storage. Moreover, it allows synchronization between content and description.

1.2.3.2 Description Definition Language

The DDL is based on XML Schema Language. But because XML Schema Language has not been designed specifically for audiovisual content description, there are certain MPEG-7 extensions which have been added. As a consequence, the DDL can be broken down into three logical normative components (structural language components, data-type language components and specific extensions).

1.2.3.3 Visual

Visual Description Tools consist of basic structures and descriptors that cover basic visual features, as colour, texture, shape, motion, localization, and face recognition. Each category consists of elementary and sophisticated descriptor.

1.2.3.4 Audio

Audio provides structures for describing audio content. Using those structures are a set of low-level descriptors for audio features that cut across many applications (e.g., spectral, parametric, and temporal features of a signal).

High-level tools include general sound recognition and indexing description tools, instrumental timbre, spoken content, audio signature description scheme and melodic description tools.

1.2.3.5 Multimedia Description Schemes

Multimedia Description Schemes comprises the set of description tools (Descriptors and Description Schemes) dealing with generic as well as multimedia entities. Generic entities are features, which are used in audio and visual descriptions, and therefore they are generic to all media.

Apart from this set of generic description tools, more complex ones are standardized. They are used when more than one medium needs to be described (for instance, audio and video). These description tools can be grouped into five different classes according to their functionality (content description, content management, content organization, navigation and access and user interaction).

1.2.3.6 Reference Software

The eXperimentation Model (XM) software is the simulation platform for the MPEG-7 Descriptors (Ds), Description Schemes (DSs), Coding Schemes (CSs), and Description Definition Language (DDL). Besides the normative components, the simulation platform needs also some non-normative components, essentially to execute some procedural code to be executed on the data structures.

The data structures and the procedural code together form the applications. The XM applications are divided in two types: the server (extraction) applications and the client (search, filtering and transcoding) applications.

1.2.3.7 Conformance

Conformance includes the guidelines and procedures for testing conformance of MPEG-7 implementations.

1.2.3.8 Extraction and use of descriptions

Extraction and use of descriptions includes informative material about the extraction and use of some of the description tools, both providing additional insight into MPEG-7 Reference Software implementation as well as alternative approaches.

1.2.3.9 Profiles

The MPEG-7 "Profiles and Levels" collects standard profiles and levels for MPEG-7, specified across ISO/IEC 15938 parts.

1.2.3.10 Schema definition

The MPEG-7 "Schema Definition" collects the complete MPEG-7 schemas, collecting them from the different standards, corrections and amendments.

1.3 Tennis particular case

Nowadays there are several approaches or proposals related with automatic event detection in the tennis case. They face the topic from different points of view, so it is possible to list the most common subjects which are currently employed to do this task. The following sections are devoted to explain that.

1.3.1 Global architectures

A large body of work has been done in the area of creating decision-making schemes. One of the most important pieces of work has been the introduction of Hidden Markov Models (HMMs). The HMMs are employed to model a tennis

match with hierarchical structures, based on stochastic processes (**Fig. 1.4**). They can be efficiently applied to join audio and visual cues.

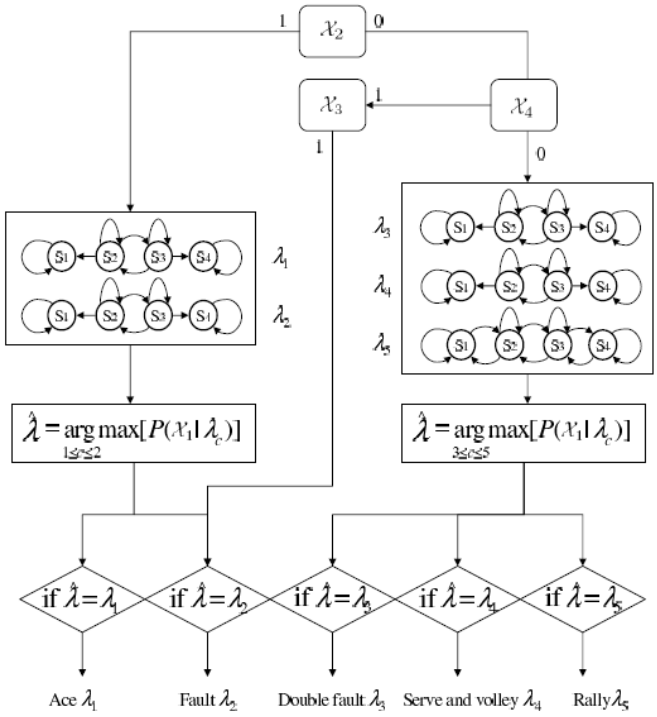

Fig. 1. 4: Event HMMs with binary classifiers.

The HMMs can exploit multimodal information and temporal relations between shots in order to identify the global structure, because it provides an efficient way to integrate features from different media.

The solution provided by this project will not take into account this work, because the event detection will be tested with video fragments that cover single shots, not entire matches. Therefore, the relations among points are not taken in mind for the analysis.

1.3.2 Audiovisual features

The related work shows different visual characteristics which have to be analyzed in tennis videos in order to detect events. First of all, it is essential to distinguish between playing frames and useless frames. Some papers propose ways to do that, as the dominant color ratio (DCR) descriptor.

There are also techniques or methods for detecting the players and the court, as could be constructing background models as well as looking for the lines through white pixel detection and specific transforms. In both cases is usually advised to do a tracking after knowing the initial positions, in order to carry out the computation process more efficiently.

It is also proposed the transformation of the extracted data into real world coordinates, using camera calibration and other parameters. Thus, the classification of events is easier than working directly with image coordinates.

Besides the visual features, there is so much work related with the audio spectral characteristics, above all focused on the racket hits and the sounds coming from the audience when a shot finishes.

Taking into account all the methods which are proposed in the consulted bibliography, during the development of the software it has been chosen those that best fit with the aim and requirements of the thesis. Furthermore, other parts of the code has been made with own ideas or modifications of related work techniques.

1.3.3 Labelling

As commented in section 1.2, MPEG-7 standard is focused on describing the multimedia content metadata which is previously extracted from an audiovisual source. In this way, sport videos can be indexed with standardized XML labels for future content searching.

Although some theoretical approaches have been found for sports in general, any specific proposal has been found for tennis environments. For this reason, an easy example for tennis content labelling is proposed in appendix A.3.

CHAPTER 2. MATCH SEGMENTATION

Tennis is a non continuous sport, so it has several time outs between shots. The method to separate the parts of the video that truly provide useful information will be detailed in this first chapter. After programming this part of the code, the implemented software will be able to distinguish among real playing scenes and those that do not. All the tests have been done from a video database which includes three different surfaces. The main characteristics of this database are detailed in section 7.1.

2.1 Frame classification

In a tennis broadcasting video, there are court view sequences and non-court view sequences. The camera always switches to court view just before the point is starting, and it remains on this way until one of the players wins the shot.

In order to detect the playing times, it is necessary to pay attention in the view changes of the camera. Sections 2.1.1 and 2.1.2 will define the two types of frame which can be found (see **[4]** and **[5]**).

2.1.1 Court view frames

When the shot is being played, the camera is situated behind one of the players, giving a perspective view of the court, as **Fig. 2. 1** shows.

Fig. 2. 1: Example of court view frame.

2.1.2 Non-court view frames

The frames temporally situated between the previous ones do not contain information about the match that is being processed, so it is necessary to discard them. They usually show the rest time of tennis players, views of the

city where the tournament is located or even close ups of the player relatives. Some examples of that frames are displayed in **Fig. 2. 2**.

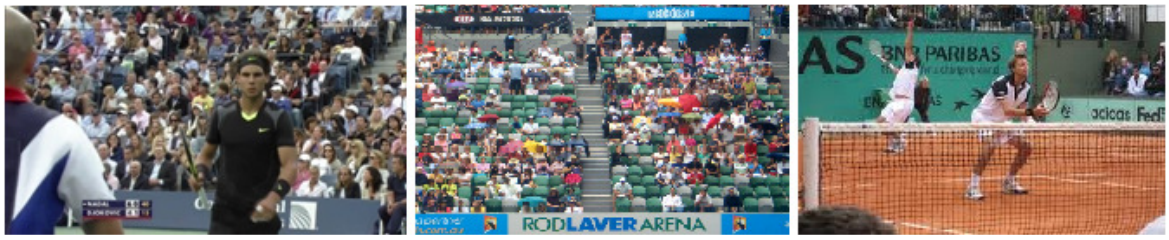

Fig. 2. 2: Examples of non-court view frames.

2.2 Histogram comparison method

The method implemented for the detection of changes in the camera views is based on the comparison between histograms. In digital image processing, a histogram is a graphical representation that shows the distribution of the gray levels in an image (or two-dimensional array).

As the frames that are being processed are RGB images, they are composed by three different matrixes. Accordingly, three histograms can be obtained from each one. **Fig. 2. 3** presents the three components of a non-court view frame with their respective histograms below.

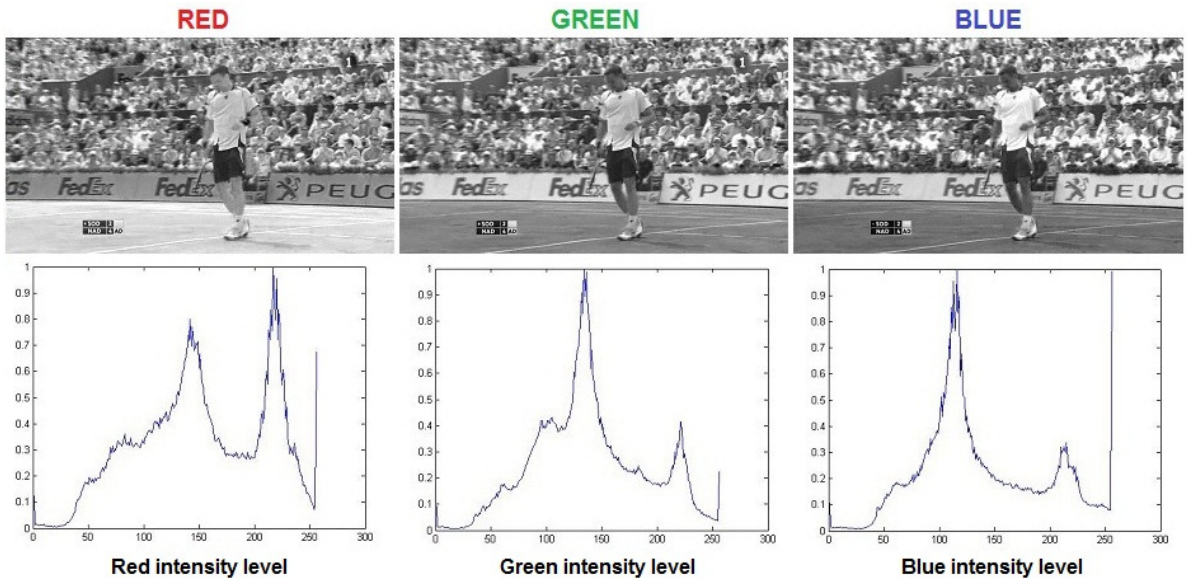

Fig. 2. 3: Histograms of the three components of a non-court view frame.

For determining the time at which the shot starts, it is necessary to differentiate the shape of the histograms among the frames. Thus, it is essential to have the shape patterns of the court view frames with the aim of having a reference. These reference histograms are defined in **Fig. 2. 4**.

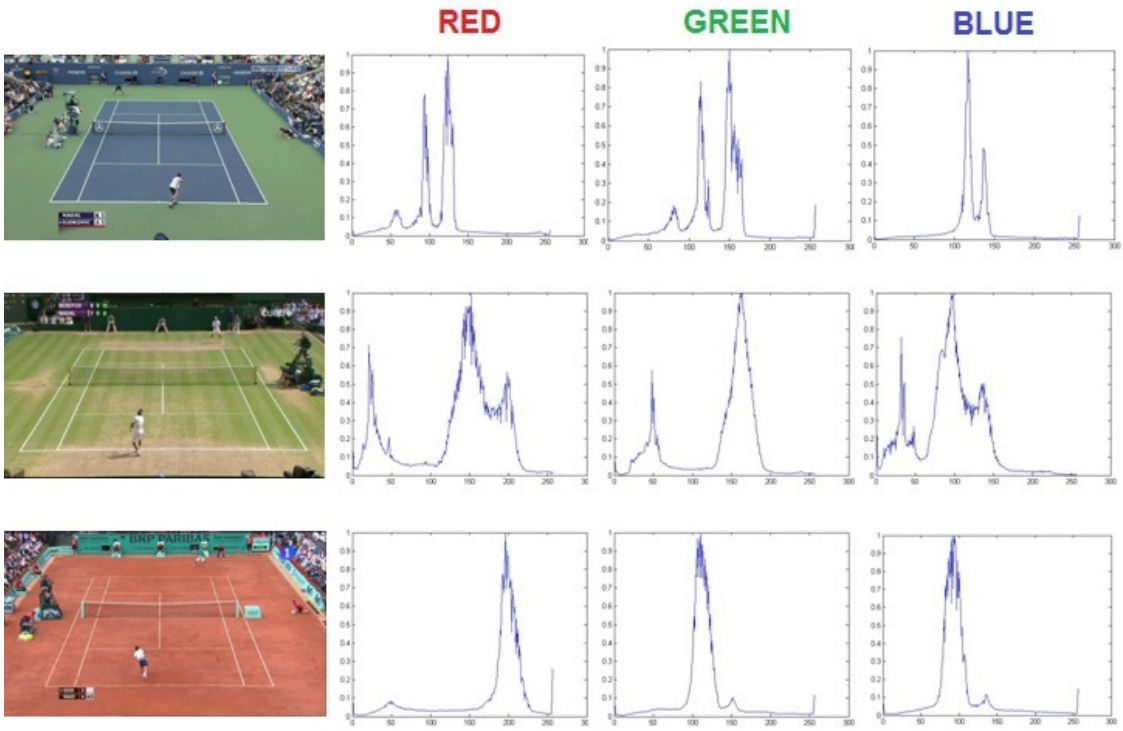

Fig. 2. 4: Histogram patterns in different surfaces.

As **Fig. 2. 4** shows, a court view shot contains a large number of court pixels, resulting in a very particular histogram shape. In fact, typically the playing surface colours contribute toward a large proportion (more than 70%) of the overall colour distribution of the whole image (**[6]**). This feature is used as a descriptor to extract the useful frames.

There are several mathematical methods for computing the comparison between histograms. In the case of this project, two options have been considered. The first one is calculating the correlation between the histogram pattern and the current frame, for all the components RGB **(Eq. 2.1)**. Obviously, after computing these operations three values are obtained.

$$R_{pattern\_current,red} = pattern_{red} * current\_frame_{red}$$

$$R_{pattern\_current,green} = pattern_{green} * current\_frame_{green}$$

$$R_{pattern\_current,blue} = pattern_{blue} * current\_frame_{blue} \qquad \textbf{(Eq. 2.1)}$$

A threshold value is defined for each component. If the comparison is being done among the pattern and any non-court image, it is so probable that one of the three calculated values exceed the threshold. In that case, the program will decide that the current frame is not showing a play in this moment.

The second method considered is conceptually easier than the previous one. It is based on calculate the subtraction (in absolute value) between histograms and average them (**Fig. 2. 5**). As well as in the case of correlations, three values will be obtained.

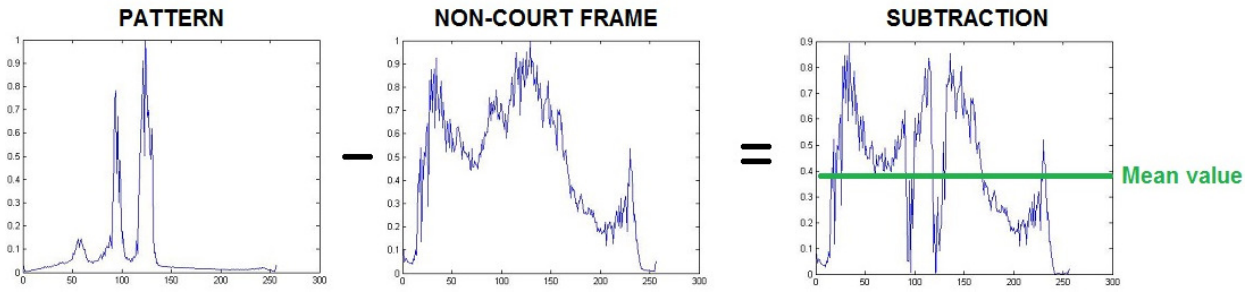

Fig. 2. 5: Subtraction method for comparing histograms.

Comparing the mean value with a threshold is possible to discard or not a frame. In the images that truly belong to a play fragment, the calculated mean value is supposed to be quite low. However, this second method has not been used finally because it would not be invariant in case of illumination changes.

There are certain complicate non-court images which are difficult to discriminate from the useful ones, as is displayed in **Fig. 2. 6**. The reason is that their histograms are so similar to the patterns. In those cases, it would be required to improve the algorithm, although it is not the goal of that thesis.

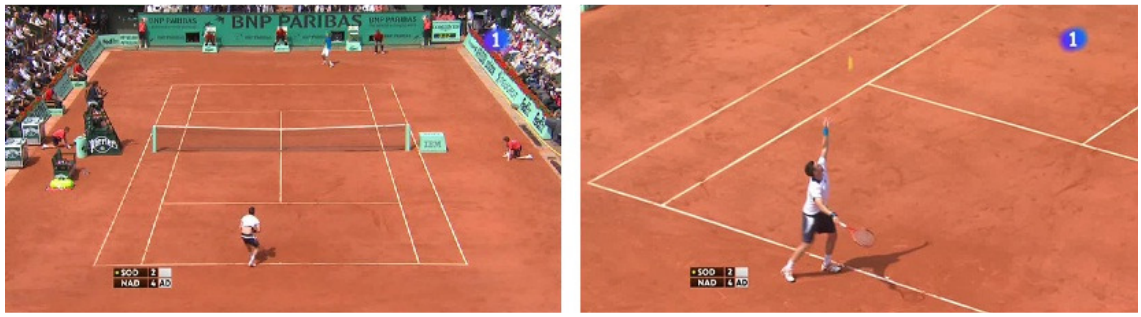

Fig. 2. 6: Left: Pattern image. Right: Example of difficult frame for discarding.

The performance obtained for the frame classification algorithm is good enough. It has been found between 3 and 6 incorrect classified frames while testing each shot. Taking into account that a shot video is composed by 300 frames on average (12 seconds), it means a performance around 98%.

CHAPTER 3. COURT DETECTION

This chapter is addressed to describe the manner to locate the rectangle which forms the court. It is a very important step because the subsequent location of the players will be referred to it. From now, all the frames that will be processed are those that have passed the filtering of previous chapter.

For getting the line positions, two stages are needed. The first one is the search of candidate points through the image. The second step is the Hough transform, which is the method chosen to check if the candidate points are really straight lines. Sections 3.1 and 3.2 are devoted to explain these issues in a more accurate way **[7]**.

After that, some later checks have to be done in order to verify if the line detection is correct. The sections 3.3 and 3.4 will discuss these topics.

3.1 Candidate points

As commented previously, the step before looking for straight lines is searching some candidate points. Although there are many techniques to find them, two different algorithms have been implemented and tested during the development of the project.

The first algorithm is the white pixel detection, which is based on previous knowledge that the lines of the court have to be white **[8]**. The second one is related with linear filters, because they are able to provide information about the edges of an image, so the lines will appear.

Both techniques have been tested separately with different results depending on the surface (hard court, grass and clay). For example, whereas white pixel detection works quite well in clay, in grass it does not. On the other hand, filtering get acceptable results in hard court while in clay not so much.

For this reason, with the aim of creating the program as robust as possible, the option that finally has been chosen is a combination of both. In this manner, it can be obtained enough good results in the three surfaces used for testing. The following sections will explain both methods accurately.

3.1.1 White pixel detection

One of the main features of the court lines is their colour. As this feature is independent of the surface type, the method can be used in any case. To find these set of points through the frame, it is necessary to transform the RGB image into the gray scale. Thus, the desired pixels may be detected by comparing their gray level with a minimum defined threshold.

Those pixels which exceed the threshold will not be discarded, and therefore they will be addressed to the next step, the filtering. At first, three different gray level thresholds were fixed, one per each surface. The values were obtained empirically, doing some tests.

However, the definition has not to be done in absolute terms, that is to say, fixing a number between 0 (totally black) and 255 (totally white). The reason is clear. If the thresholds are defined as simple constants, the algorithm will only work properly in the tennis videos which have been using for testing during the project. When using other videos in which the weather or the illumination changes, the algorithm probably will stop working.

In order to avoid this issue, the three threshold values are defined as percentages. It is easy to know approximately what percentage of pixels forms the white lines in relation to the total number of pixels of a frame (**Table 3. 1**)

Table 3. 1: Definition of thresholds in percentage.

Tournament	White pixel threshold
US Open	4 %
Roland Garros	7.5 %

Fig. 3. 1 tries to exemplify this concept of thresholding in the US Open particular case. The white pixels that form the lines are situated in the little right peak of the histogram, which have a gray value higher than 245.

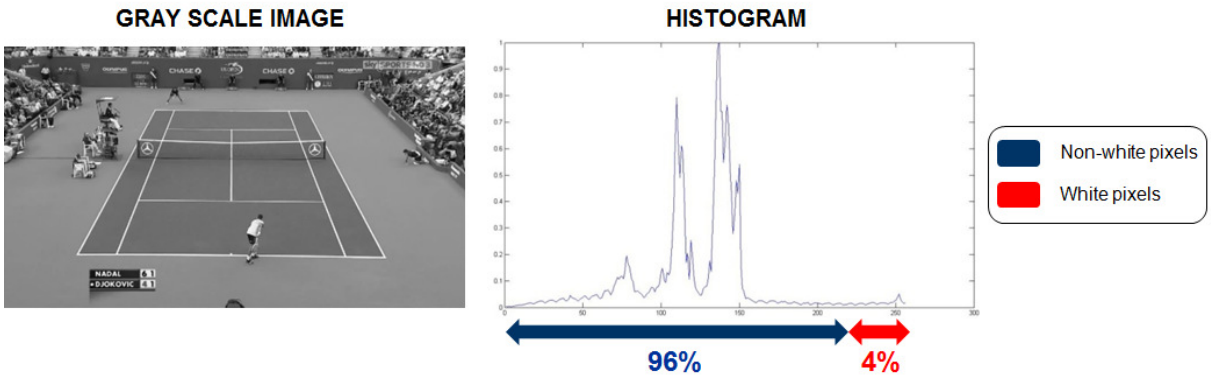

Fig. 3. 1: Histogram of gray scale image.

Applying the values of **Table 3. 1**, the results obtained are shown in the images of **Fig. 3. 2**.

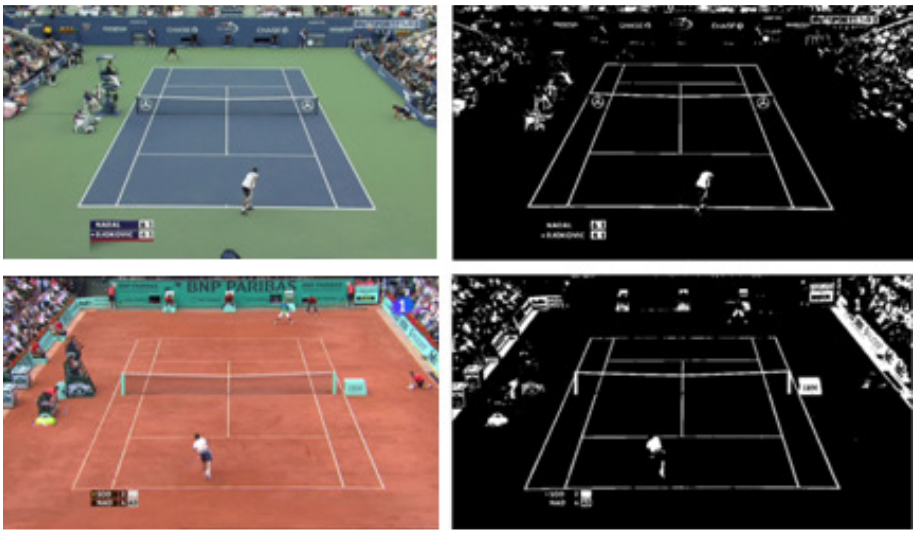

Fig. 3. 2: Up-left: US Open original image. Up-right: US Open white pixels. Down-left: Roland Garros original image. Down-right: Roland Garros white pixels.

In case of Wimbledon surface, there is a particular problem that does not allow using exactly the same white pixel method. The problem and the solution applied are discussed in section 3.1.1.1.

3.1.1.1 Wimbledon special case

Unlike the other two cases, in the Wimbledon surface it appears an additional problem. The non-uniformity of the court due to the sand causes that the difficulty for finding the horizontal lines increases so much. This problem is seen in **Fig. 3. 3**.

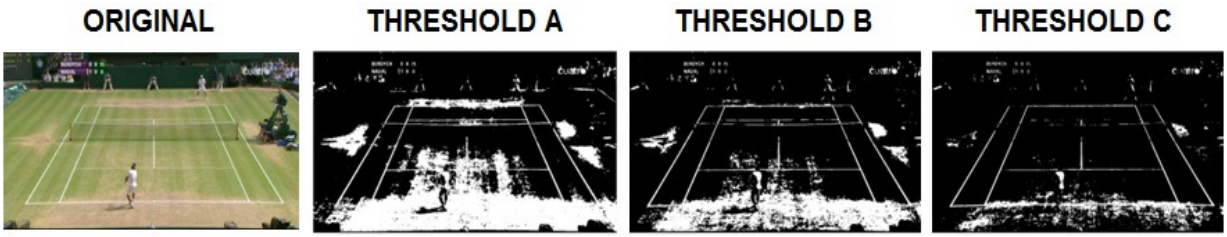

Fig. 3. 3: Problem with the sand.

If the gray level value threshold applied on the image is so low (threshold A in **Fig. 3. 3**), it will be not possible to differentiate between the horizontal lines and the sand of the ground. On the other side, if the value is too restrictive (threshold C) the most part of the pixels belonging to sand are deleted, but some useful pixels which pertain to lines are also removed.

For this reason the simple algorithm defined to the other surfaces does not fit well in this special case. The proposed solution is to define four regions in the frame, in order to look for candidate points in four different ways (**Fig. 3. 4**). Whereas in previous cases one single value was fixed to the whole image, in the current case four value thresholds will be defined, one per spatial region.

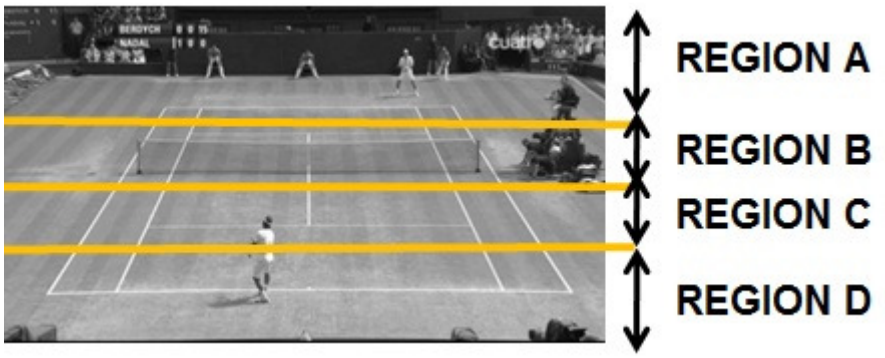

Fig. 3. 4: Spatial regions defined for Wimbledon case.

As is predictable, in region A and region D (where the most part of the sand is located) it will be necessary to be highly restrictive when fixing the threshold value. Thus, the false white candidate points will be removed, remaining the correct ones. The percentage values applied to Wimbledon frames are detailed in **Table 3. 2**.

Table 3. 2: Threshold definition in Wimbledon special case.

Spatial region	White pixel threshold
A	2.75 %
B	14.5 %
C	4.5 %
D	2.75 %

Once applied this variation in the code, the detection improves substantially. The results are evidenced in the right image of **Fig. 3. 5**.

Fig. 3. 5: White pixel detection by region.

3.1.2 Filtering

The filtering is the next stage of the candidate points searching. Although an important pre-selection has been done up to now, there are still several white pixels that have to be rejected as candidates. Some examples of these pixels are the situated in the audience, billboards around the court or even above the players themselves (**Fig. 3. 6**).

Fig. 3. 6: False initial candidate points.

With the aim of having as less number of points as possible (for saving in computing operations during the later Hough Transform), different kind of linear filters could be passed on the image. In this way, only those white points which define edges will stay as candidates.

There are different linear filters that can be used, as Sobel, Prewitt, Roberts or Canny. All of them work well when looking for edges. Nevertheless, each one has specific features that make it work better in a surface than another. After testing all the possibilities, the best results have been obtained following the scheme of **Table 3. 3**.

Table 3. 3: Linear filters used.

Tournament	Filter
US Open	Roberts
Wimbledon	Canny
Roland Garros	Canny

Roberts filter is a differential operator which approximates the gradient of an image through discrete differentiation. It is achieved by computing the sum of the squares of the differences between diagonally adjacent pixels. It returns edges at those points where the gradient of the image is maximum.

Canny filter is more complex than Roberts, because it has multiple stages. It finds edges by looking for local maxima of the gradient too. This gradient is calculated using the derivative of a Gaussian filter **[9]**.

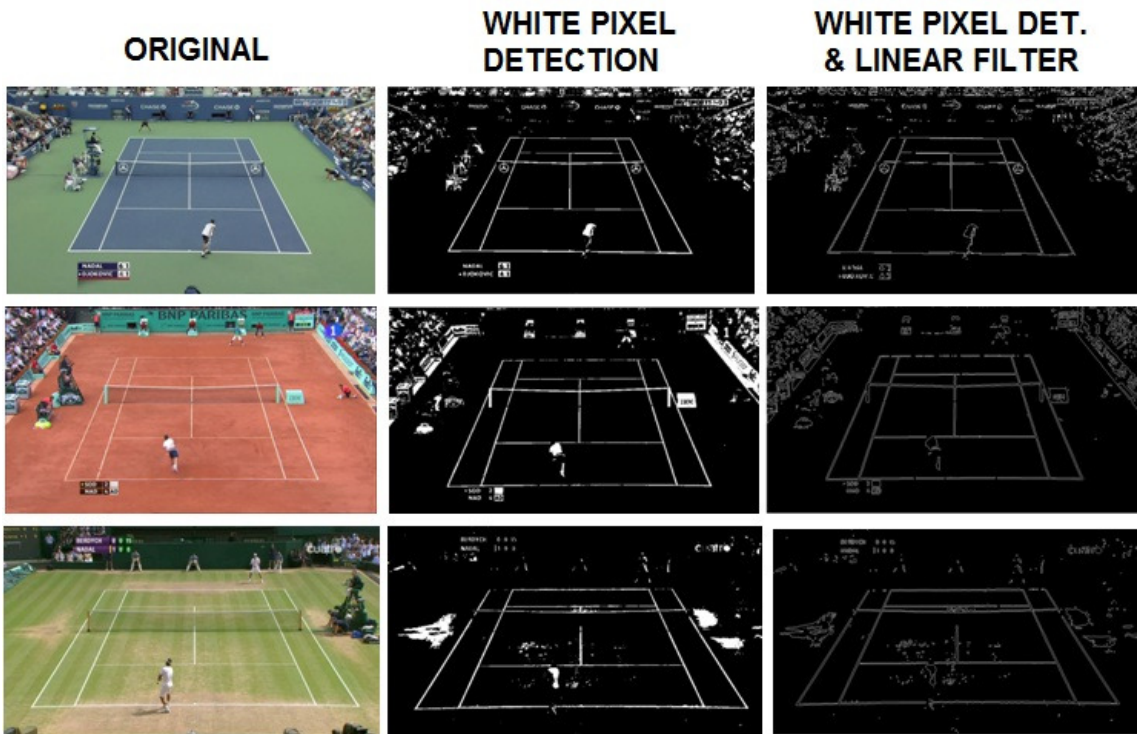

Fig. 3. 7: White pixel detection and filtering.

However, not all the useless pixels could be removed. For example, in right images of **Fig. 3. 7** it can be appreciated some points that actually do not belong to the surface lines. They usually appear in the audience or in player edges, but they are so difficult to eliminate.

Nevertheless, due to the last restriction imposed (linear filtering), the number of candidate points has been considerably reduced. **Fig. 3. 8** shows the steps that have been explained and developed up to this part of the report. The following section will discuss about the next necessary phase, always with the final goal of finding the situation of the court.

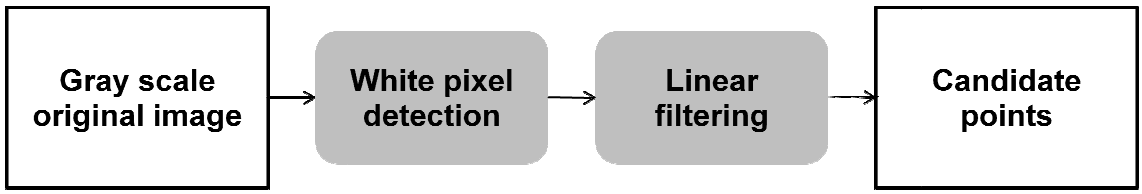

Fig. 3. 8: Stages implemented to obtain candidate points.

3.2 Hough transform

The set of points which remains after the previous stage are considered as the input of the next one, the Hough transform. The output of this transform will not be single points, but candidate straight lines (see **Fig. 3. 9**).

Fig. 3. 9: Input and output of the Hough transform.

3.2.1 Definition

The Hough transform is a technique which can be used to isolate features of a particular shape within an image. As it requires that the desired features to be specified in some parametric form, the classical Hough transform is most commonly used for the detection of regular curves such as lines, circles or ellipses **[10]**.

The main advantage of the Hough transform technique is that it is tolerant of gaps in feature boundary descriptions and it is relatively unaffected by image noise.

The function that implements this transform is included in the Matlab toolbox, therefore it is not necessary to develop it from scratch. However, it is important to know the main operation principle in order to configure properly some parameters. The section 3.2.2 is aimed to do so.

3.2.2 How it works

The Hough technique is particularly useful for computing a global description of a feature, given local measurements. The motivating idea behind the Hough technique for line detection is that each input measurement (coordinate points coming from the prior stage) indicates its contribution to a globally consistent solution.

The easiest case is the linear transform for detecting straight lines. In the image space, the straight line can be described as **(Eq. 3.1)**.

$$y = mx + b \qquad \textbf{(Eq. 3.1)}$$

The main idea is to consider the characteristics of the straight line not as image points (x_1, y_1), (x_2, y_2), but instead, in terms of its parameters. These parameters are the slope m and the intercept parameter b. Based on that, the straight line $y = mx + b$ can be represented as a point (b, m) in the new parameter space.

Nevertheless, for computational reasons, it is better to use a different pair of parameters for defining the lines, named as ρ and θ.

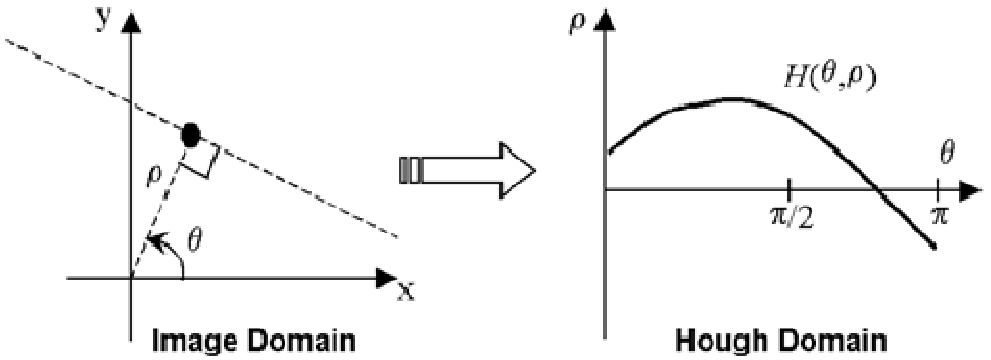

Fig. 3. 10: Image to Hough domain transformation.

The parameter ρ represents the distance between the line and the origin, whereas θ is the angle of the vector from the origin to this closest point. Using these new parameters, the equation of the line can be written as in **(Eq. 3.2)**.

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right)x + \left(\frac{\rho}{\sin\theta}\right) \qquad \textbf{(Eq. 3.2)}$$

(Eq. 3.2) can be reorganized to **(Eq. 3.3)**, which is more convenient for describing a set of lines.

$$\rho = x\cos\theta + y\sin\theta \qquad \textbf{(Eq. 3.3)}$$

The coordinates of the points of an edge segment are known, so they are considered as constants in **(Eq. 3.3.)**. On the other hand, ρ and θ are the unknown variables which have to be calculated. This point-to-curve transformation is the Hough transformation for straight lines.

Points which are collinear in the cartesian space become so clear in the Hough space. They yield curves which intersect at a common [ρ, θ] point, as seen in the example of **Fig. 3. 11**.

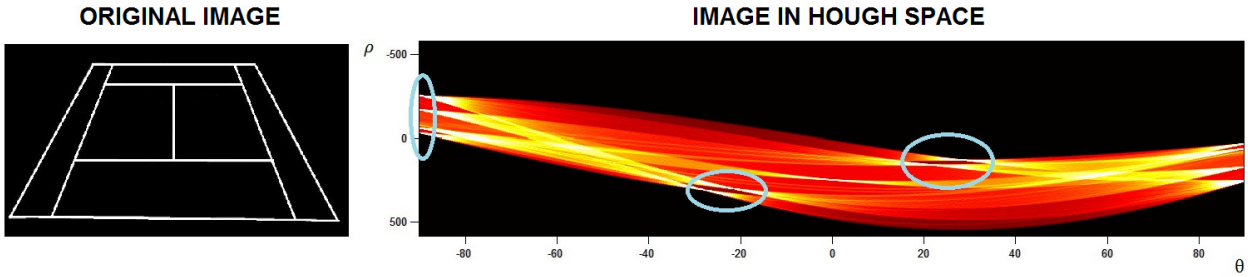

Fig. 3. 11: Example of image transformation into Hough space (the blue circles highlight the main intersections).

The intersections found in the Hough space of a frame are necessary to determinate the angle and distance to origin of the court lines. Once these features are known, it is possible to locate the lines in the frame. A first approach of the results obtained is displayed in **Fig. 3. 12**.

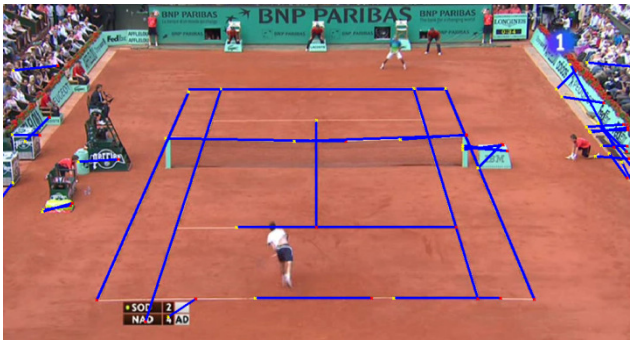

Fig. 3. 12: Initial results of the transform.

The previous example shows the lines computed and provided by the Hough transform. There are some parts of the court and some segments which have not been detected. The reasons can be explained as the low contrast among the line and the clay background.

Other possible reason could be the ρ and θ resolution. As commented before, during the development of the thesis has not been necessary to implement the Hough transform, because it is already provided by the programming language tools. Nevertheless, it is important to know the main features in order to adjust its parameters correctly. The resolution configuration of ρ and θ can solve problems as, for example, the baseline situated at the bottom of **Fig. 3. 12**. It is separated in different segments, when it should not be so. In any case, the fact of losing some lines is not essential for calculating the court position, as will be explained in section 3.4.

The most part of the lines highlighted in blue in **Fig. 3. 12** are well located above the court. However, it can be observed some false line detections. They are situated over the audience and billboards mainly. With the purpose of eliminating the false lines, a new stage have to be implemented, the false line rejection module.

3.3 False line rejection

Different candidate lines have been acquired up to now. The next part of the algorithm is aimed to distinguish between the correct and the wrong ones. To do it, it is necessary to establish some requirements that the detected straight lines have to satisfy. The conditions which have to be defined are related with the location, length and slope. At the end of the stage, the lines will be finally validated.

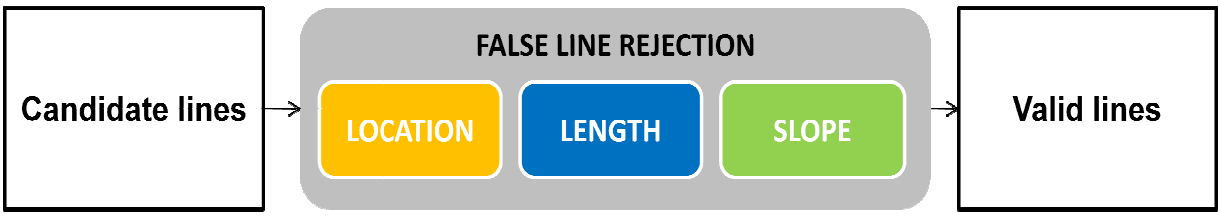

Fig. 3. 13: Sub-stages composing the false line rejection module.

The orange ellipse drawn in **Fig. 3. 14** highlights examples of lines which are located in a incorrect known area. It is easy to discard them, simply by checking their positions.

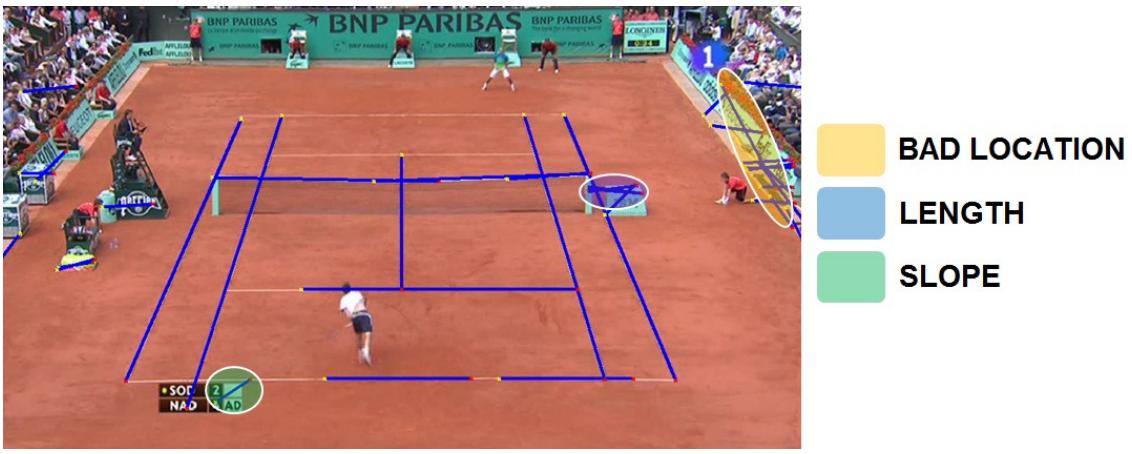

Fig. 3. 14: Example of bad line detections.

There are other cases where the lines are detected in coherent positions, but they do not accomplish the requirements of length or slope. The blue and green ellipses are indicating these kinds of issue. Specifically in the slope feature case, only five possible angles are supposed to appear, four of them to describe the external vertical lines and one for the rest of horizontal lines (zero degrees). For sure, a tolerance of some degrees has to be defined for each accepted slope.

In the rejection by length, a line will be discarded if is longer than the longest expected line of the court. The same would happen if too short. After applying the three filters explained in this section, the results improve considerably, as demonstrated in **Fig. 3. 15**.

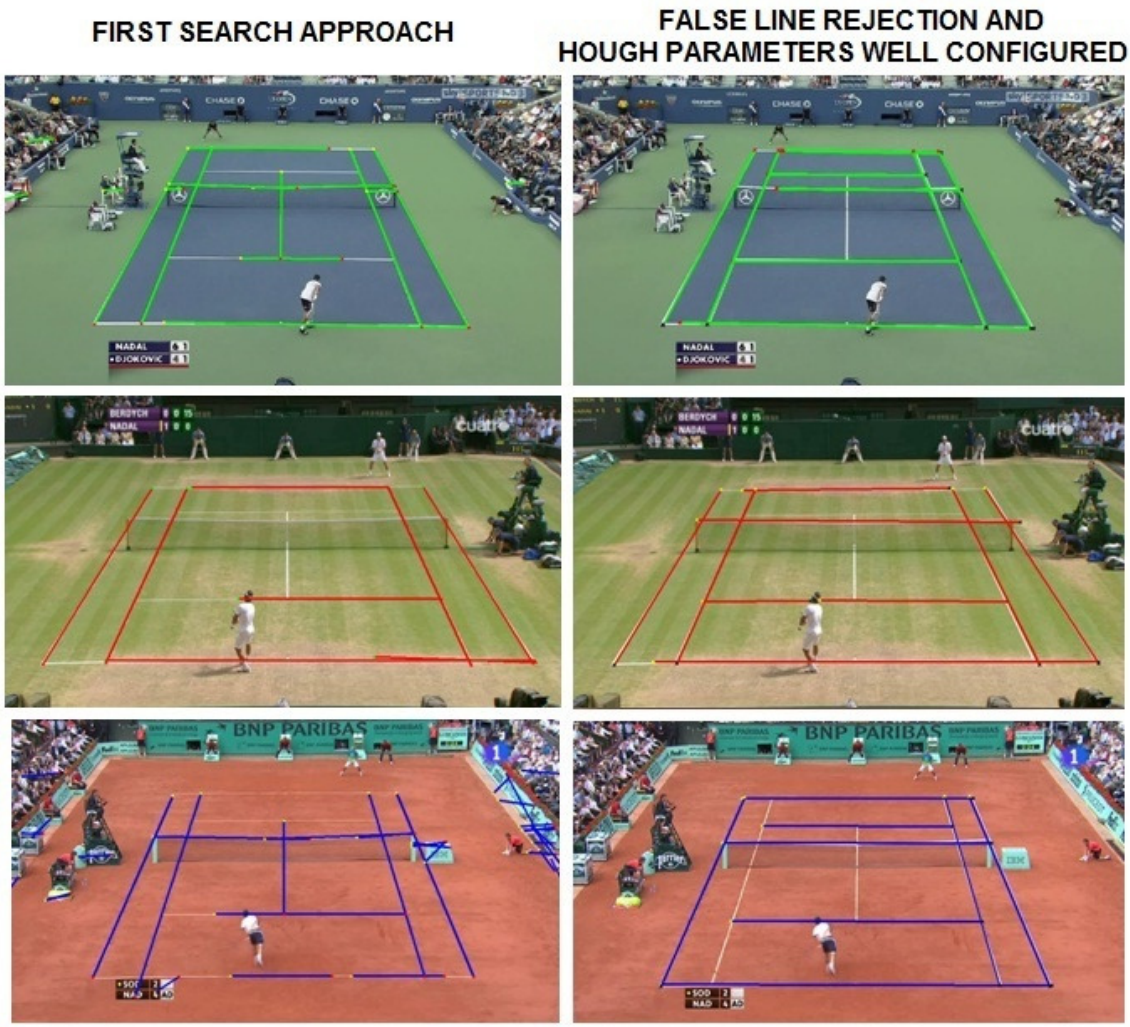

Fig. 3. 15: Improvements when imposing restrictions and configuring Hough parameters.

The court geometry and the dimensions are always the same. Despite of that, the camera position is different depending of the tournament. For this reason, the restrictions of location, length and slope cannot be the same for the three cases studied. This issue is so evident in **Fig. 3. 15**, where in Wimbledon surface the camera is located so much close to the players than in the other two cases.

3.4 Corner validation

The corner validation is the last stage of the court detection module. Up to this moment, the detected lines have been validated and checked individually, but not together while forming the rectangle. This part of the algorithm can be separated in two steps, following the scheme of **Fig. 3. 16**.

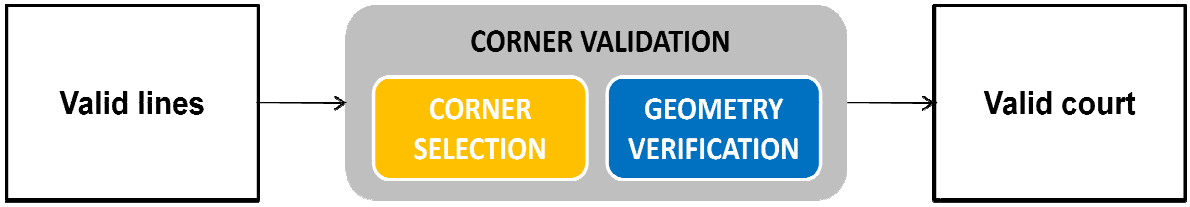

Fig. 3. 16: Sub-stages composing the corner validation module.

3.4.1 Corner selection

It has been decided to get the location of the players using the four corners of the court as reference. Thus, it is necessary to extract them from the validated lines coming from the previous stage, specifically from the endpoints of each segment.

An easy way to do it is selecting the points which minimize the distance between their position and the external corner of the frame. This issue is displayed in **Fig. 3. 17**.

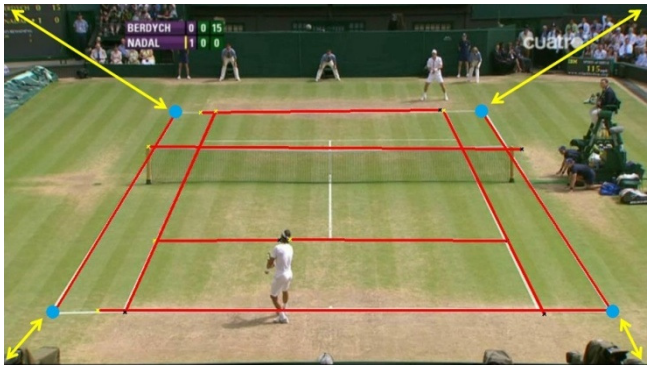

Fig. 3. 17: In yellow: Distance to be minimized. In blue: Final selected corners.

Moreover, it could be interesting to make sure that these points are situated where two lines intersect. Thus, the algorithm becomes more robust. The following step is aimed to finally check the geometry of the rectangle which is defined by the points selected.

3.4.2 Geometry verification

Once the algorithm has decided where the corners are located, it is necessary to check if they really are, and therefore, if the whole court detection has been satisfactory. The method used to test is simply verifying if the rectangle which is formed fit with some geometric features. These conditions are related with how a rectangle is seen in perspective.

For example, a basic condition could be that both horizontal lines must be parallel (A and B in **Fig. 3. 18**), with some tolerance accepted. If this condition is not satisfied, the rectangle cannot be validated.

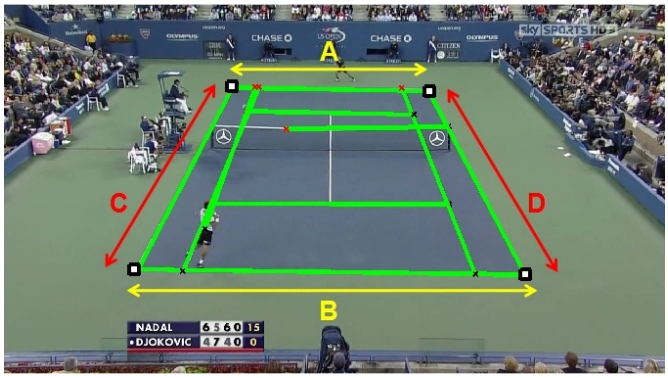

Fig. 3. 18: In yellow: Horizontal lines (A and B). In red: Vertical lines (C and D).

Another restriction is related with the proportionality. Whereas segments C and D may have similar length, A and B have to maintain the same proportion in order to meet with the requirements **(Eq. 3.4)**.

$$k \cdot A = B$$
$$C \approx D \qquad \textbf{(Eq. 3.4)}$$

It is possible that some important lines can be lost in specific frames. It can be caused by a change of illumination, low contrast in some parts of the image or even by the wrong discarding of the line in a previous stage. An example of this situation is shown in **Fig. 3. 19**.

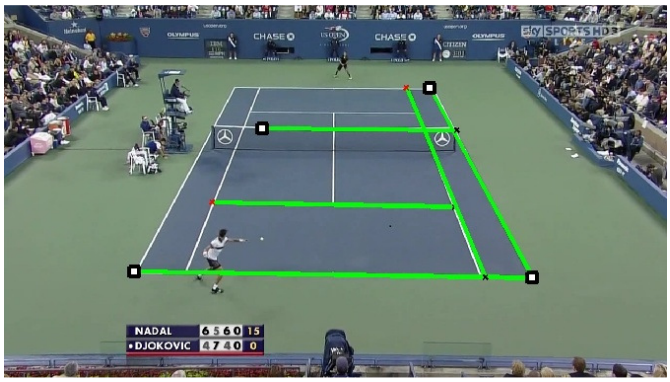

Fig. 3. 19: Example of wrong corners that have to be discarded.

In **Fig. 3. 19** the situation is the following. All the green segments have passed properly the location, slope and length tests. Nevertheless, the lack of the vertical-left and horizontal-up line detections causes that the four selected corners are the white points. As they do not fit with the conditions defined in **(Eq. 3.4)**, the detection is wrong and they have to be discarded.

When this particular situation happens, the implemented algorithm uses the corners information of validated courts in past frames. The error introduced is low due to the small movements of the camera.

The performance obtained for the court detection module is around 68%. It is not so high, but there are several factors which make difficult to do this task. Although it can be improved, the goal of the thesis is to provide a whole solution for the event detection topic, not to be focused on a specific part. For this reason, the local results obtained for this part have been considered as enough. More detailed information about is described in section 7.2.1.

CHAPTER 4. PLAYER DETECTION

This section aims to define the techniques and methods proposed to detect the tennis players in the image. The chapter is divided in two parts, one per each module of the algorithm.

The first one is devoted to explain how to make the initial searching. This operation will be done one time, because once the players are detected in a first frame, the locations in subsequent frames will be calculated by tracking themselves. In this manner, the code is more efficient than calculating the positions every time. The second section will show the selected way to do the tracking (see **[8]** and **[11]**).

4.1 Search in first frame

When the first court view frame is detected, it is supposed that the players are ready to start a shot. The previous knowledge about tennis positioning allows looking for the players in very specific areas. This knowledge makes easier to find them and therefore, save in computing time.

Before serving, both players are always situated very close to their base lines. Up to now, the role of each player is unknown (service or receiver), therefore the searching has to be done around the entire horizontal axis. The areas where the algorithm will look for the players are specified in **Fig. 4. 1**.

Fig. 4. 1: In red: Searching area for the upper player. In yellow: Searching area for the lower player.

Once defined the area to look up, it is necessary to define which method has been chosen for doing it. A simple way for the player detection is the vertical and horizontal filtering. The templates which characterize these filters are detailed in **Fig. 4. 2**.

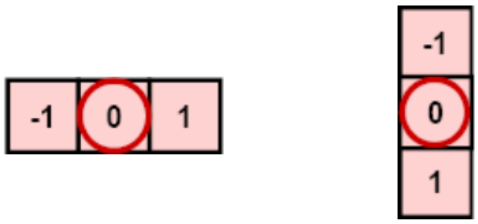

Fig. 4. 2: Left: Horizontal filter template. Right: Vertical filter template.

The horizontal filter is used to highlight the vertical lines in an image. On the other hand, the vertical filter is aimed to obtain the horizontal ones. The examples of **Fig. 4. 2** are the simplest, but the filters that have been truly used are larger (3x3 size).

A player is not a geometric figure composed by straight lines, but its irregular shape will provide the necessary information to locate it. If each filter is applied above the regions defined before and the results are added, several points will appear in the surroundings of the player.

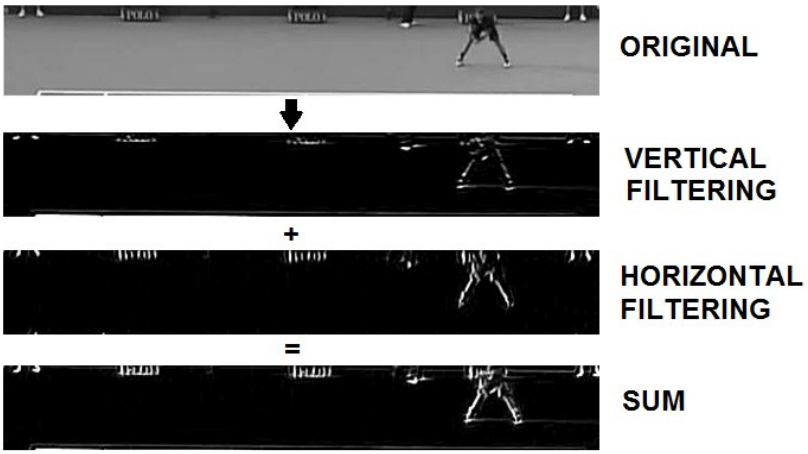

Fig. 4. 3: Filters applied over the upper searching region.

As displayed in **Fig. 4. 3**, the most part of the white points appear in the player's edges. In order to get exactly his position, it is necessary to define a window for calculating which part of the filtered region has the highest density of white points.

The position in respect to the vertical axis is known by definition, so the only question to solve is the horizontal position. To get it, the window starts situated on the left part of the region. The operation is based on adding the number of white pixels that are contained on it. Afterwards, the window is moved from left to right through the area.

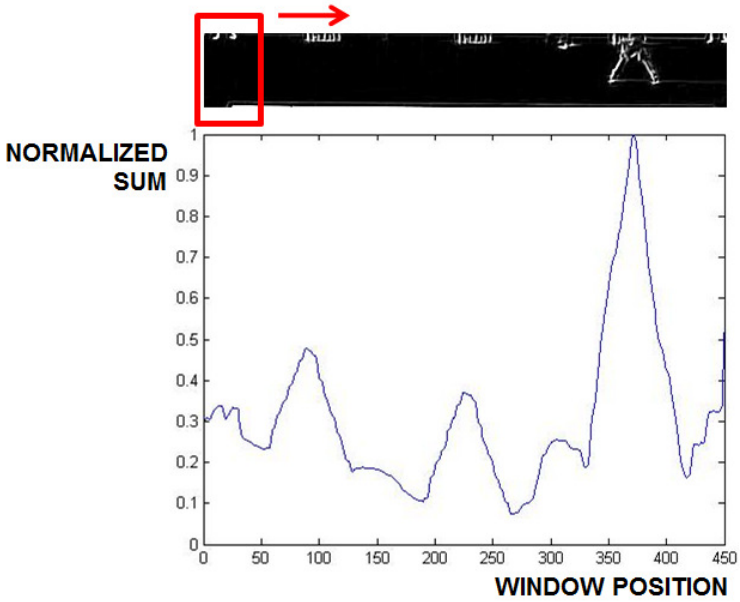

Fig. 4. 4: Window movement and adding results in upper region.

In the example of **Fig. 4. 4**, when the window (in red) reaches the player position, a sharp peak appears in the graphic. The algorithm will consider this position as the more confident for the upper located player. The same scheme is followed during the searching in the lower region case (**Fig. 4. 5**).

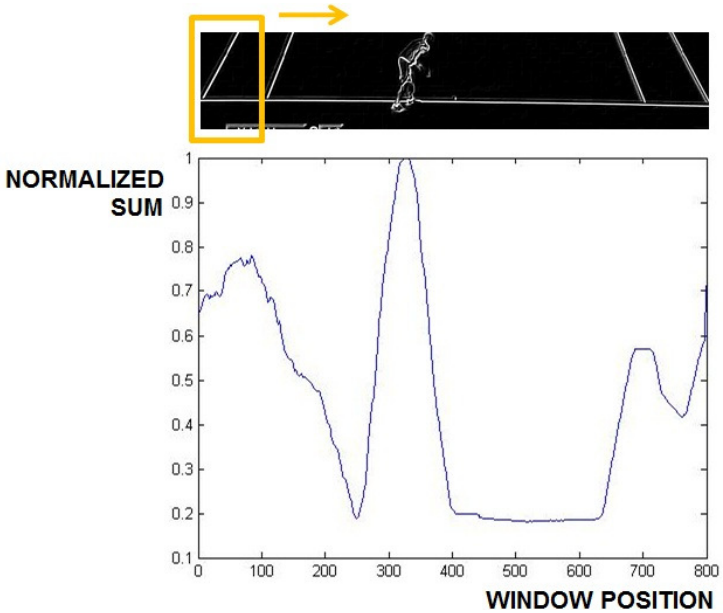

Fig. 4. 5: Window movement and adding results in lower region.

In the graphic displayed above it is also easy to detect the higher peak. However, in the extreme parts of the filtered region some lines (which belong to the court) appear. Although usually the peaks generated by these lines are not as large as the generated by the player, in some cases they could lead to a

wrong detection. This effect can be avoided by filtering the region in more directions, not only the vertical (90 degrees) and horizontal (zero degrees), as made until now. In this manner, the peak generated by the player should grow considerably. Here it is produced a trade-off between number of calculations and the strength of the algorithm. As summary, **Fig. 4. 6** details the different steps described in this section.

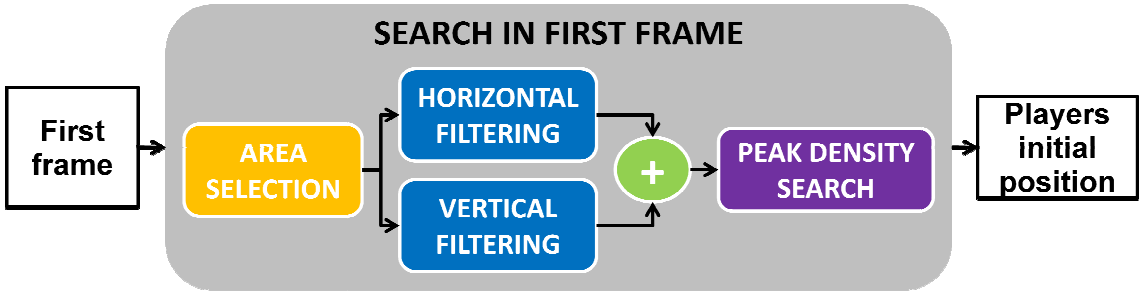

Fig. 4. 6: Sub-stages composing the player search in the initial frame.

4.2 Player tracking

The player tracking is the second and last step of the player detection module. After getting the positions in the first frame searching, it is not necessary to compute all the calculations again for the next ones. As the locations in consecutive frames are practically the same, the past information can be exploited with the aim of saving in computing time.

A neighbourhood is defined around the position obtained in the previous step (**Fig. 4. 7**). Taking this little region that surrounds each player and comparing it with the same region in the forward frame, it is possible to get the new position with very simple operations.

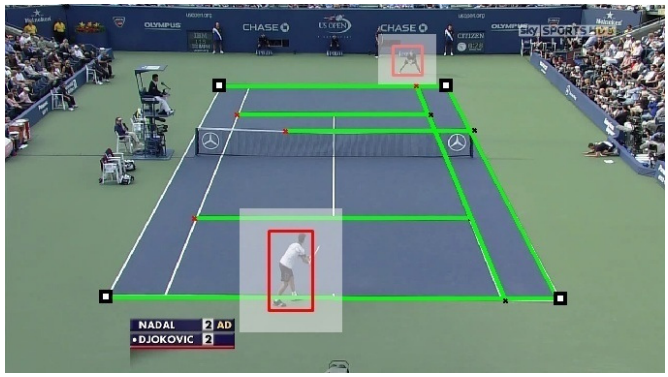

Fig. 4. 7: In red: Player detection in previous frame. In white: Neighbourhood defined.

The method is based on looking for the differences, taking into account that the only expected variations are supposed to come from the player movement. Computing the subtraction of consecutive neighbourhood frames, it can be obtained the new position, as shown in **Fig. 4. 8**. Looking for the area with the highest white point density, the location will be updated.

Fig. 4. 8: Subtraction method for tracking the player movement.

Fig. 4. 9 shown the different stages defined in this section. Its appearance is quite similar to **Fig. 4. 6**, however by using only subtraction instead of filters produces an important save in computational load.

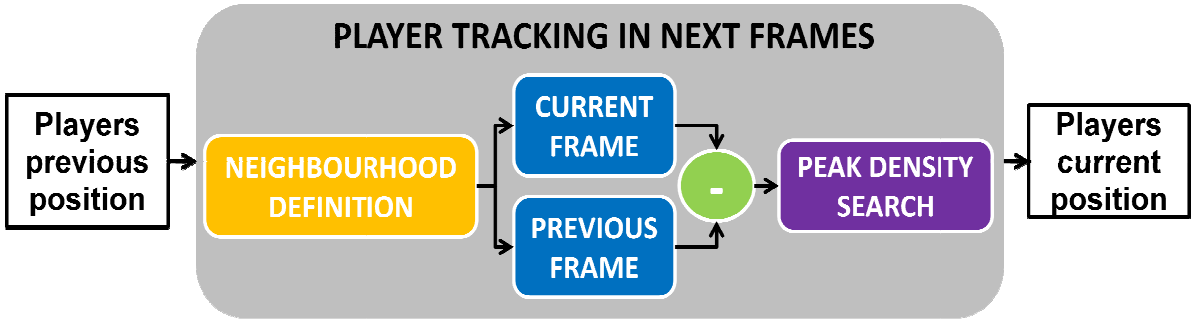

Fig. 4. 9: Sub-stages composing the player tracking in the subsequent frames.

The performance obtained for the player detection module is around 90%. It is really better than the previous module, because there are not so many variables that can affect the detection. More information about this issue is detailed in section 7.2.2.

CHAPTER 5. COORDINATES CONVERSION

The semantic analysis of sport videos depends on the information about player and ball positions. Nevertheless, the locations obtained in the previous chapter are referred to an image coordinate system, so they are not useful enough for classifying events **[12]**.

Therefore, it is necessary to implement a new module able to convert the screen coordinates into real-world coordinates. This chapter is intended to explain a method to do that.

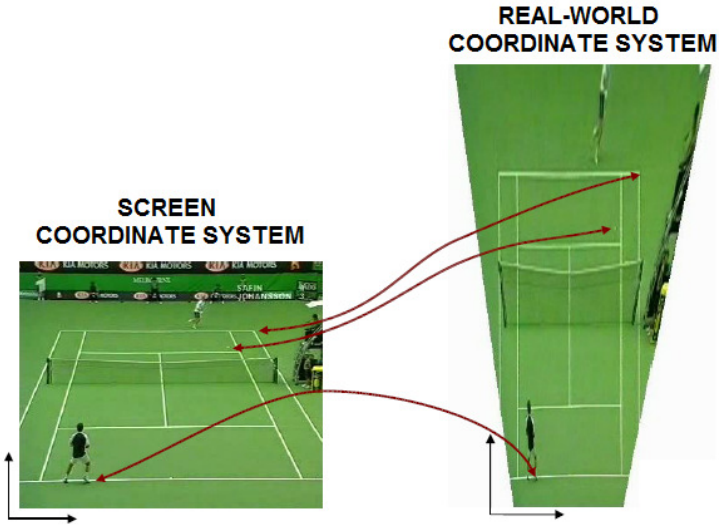

Fig. 5. 1: Screen to real-world coordinates conversion **[12]**.

As shown in **Fig. 5. 1**, when the conversion is applied on the original image, the court lines form a perfect rectangle in the real-world coordinates system, whereas in the initial system they perform a trapezoidal figure. The new situation will allow getting the real coordinates of the players, and therefore being able to extract the shot information.

The technique used to compute the transformation is related with the perspective properties of the tennis frontal view. From the vertical lines obtained after the court detection module is possible to calculate the vanishing point coordinates, which is the point where these lines intersect. The section 5.1 is devoted to detail this part of the algorithm.

Then, doing some vertical and horizontal projections of the player position in the screen coordinate system, the real coordinates can be obtained, as explained in section 5.2. It will be the last part of the algorithm dedicated to extract low level descriptors.

5.1 Vanishing point calculation

The vanishing point is the point in a perspective image where parallel lines converge (although in fact they do not appear as parallel in the image). In case of tennis broadcasting view, there is only one vanishing point to be calculated, the simplest case.

The first step is defining the equations that come from the external straight lines of the court. This equation can be extracted from two points belonging to each straight line, as can be the corners calculated in previous modules.

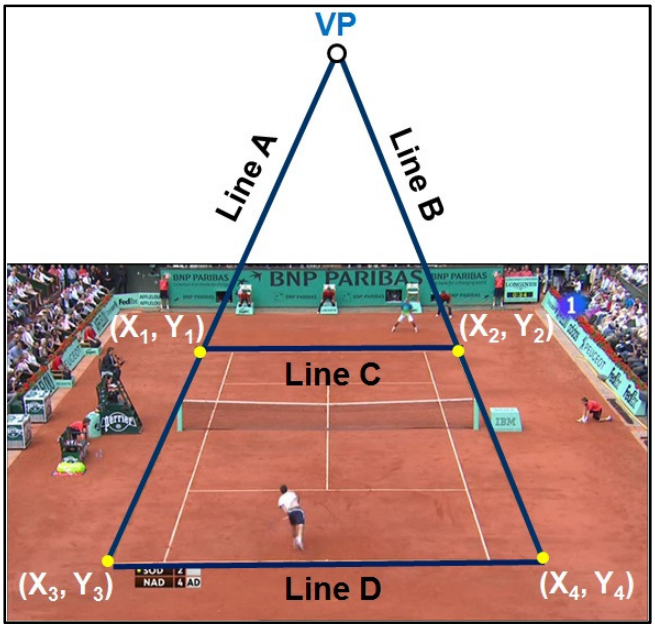

Fig. 5. 2: Vanishing point calculation.

Taking into account the nomenclature defined in **Fig. 5. 2**, the equations can be obtained from **(Eq. 5.1)**, **(Eq. 5.2)**, **(Eq. 5.3)** and **(Eq. 5.4)**.

$$A \equiv y = \left(\frac{Y_1 - Y_3}{X_1 - X_3}\right) x + \left(Y_3 - \frac{Y_1 - Y_3}{X_1 - X_3} \cdot X_3\right) = m_A x + n_A \qquad \textbf{(Eq. 5.1)}$$

$$B \equiv y = \left(\frac{Y_4 - Y_2}{X_4 - X_2}\right) x + \left(Y_2 - \frac{Y_4 - Y_2}{X_4 - X_2} \cdot X_2\right) = m_B x + n_B \qquad \textbf{(Eq. 5.2)}$$

$$C \equiv y = \left(\frac{Y_2 - Y_1}{X_2 - X_1}\right) x + \left(Y_1 - \frac{Y_2 - Y_1}{X_2 - X_1} \cdot X_1\right) = m_C x + n_C \qquad \textbf{(Eq. 5.3)}$$

$$D \equiv y = \left(\frac{Y_4 - Y_3}{X_4 - X_3}\right) x + \left(Y_3 - \frac{Y_4 - Y_3}{X_4 - X_3} \cdot X_3\right) = m_D x + n_D \qquad \textbf{(Eq. 5.4)}$$

The next step is computing the intersection among line A and line B. Although the other lines (C and D) are not necessary to obtain the vanishing point location, they will be useful for the following stage, the translation to two-dimensional plane. The vanishing point can be finally calculated as **(Eq. 5.5)** and following the scheme of **Fig. 5. 3**.

$$VP(x,y) = \left(\frac{n_B - n_A}{m_A - m_B}, \frac{m_A \cdot n_B - m_B \cdot n_A}{m_A - m_B}\right) \quad \textbf{(Eq. 5.5)}$$

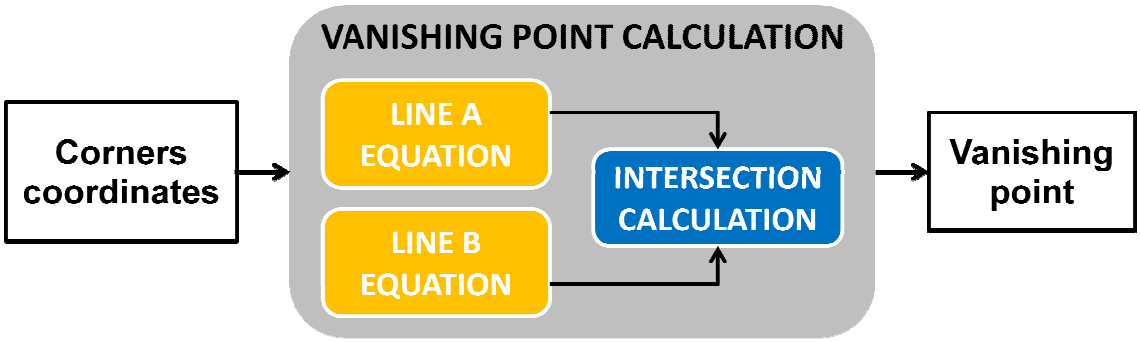

Fig. 5. 3: Sub-stages composing the vanishing point module.

5.2 Translation to 2D plane

Once the position of the vanishing point is calculated, it is necessary to relate it with the players' positions, both still in screen non-real coordinates. In order to perform it, some projections have to be done from the players to the four lines which come from the prior module.

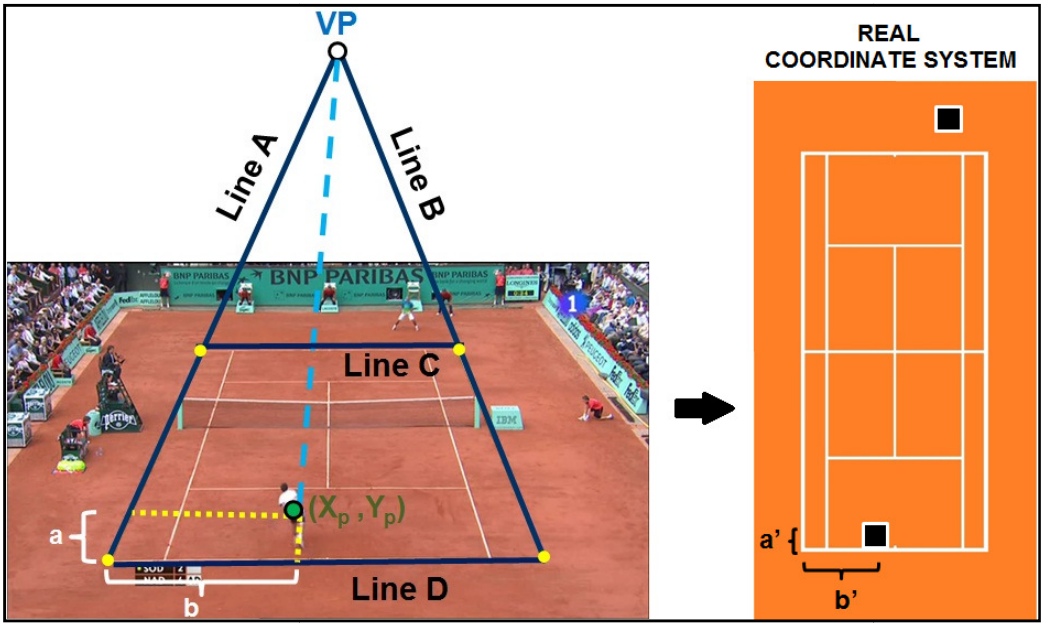

Fig. 5. 4: Left: Projections from player to external lines. Right: Translation to real coordinate system.

First of all, the equation from the player to the vanishing point may be defined. In this way, it is obtained the expression of **(Eq. 5.6)**.

$$VP \rightarrow Player \equiv y = \left(\frac{VP_y - Y_p}{VP_x - X_p}\right)x + \left(Y_p - \frac{VP_y - Y_p}{VP_x - X_p} \cdot X_p\right) \qquad \textbf{(Eq. 5.6)}$$

The equation defines the light blue straight line appearing in **Fig. 5. 4**. The player position is highlighted with a green circle, whereas the yellow dotted lines are the projections obtained from the player to the court external lines. The intersections between them will provide the required information to know the position in the real coordinate system.

The parameters 'a' and 'b' are used finally to situate the player in the two-dimensional plane. All these steps must be done twice, one per each player. **Fig. 5. 5** tries to show an overview of the last module of the algorithm.

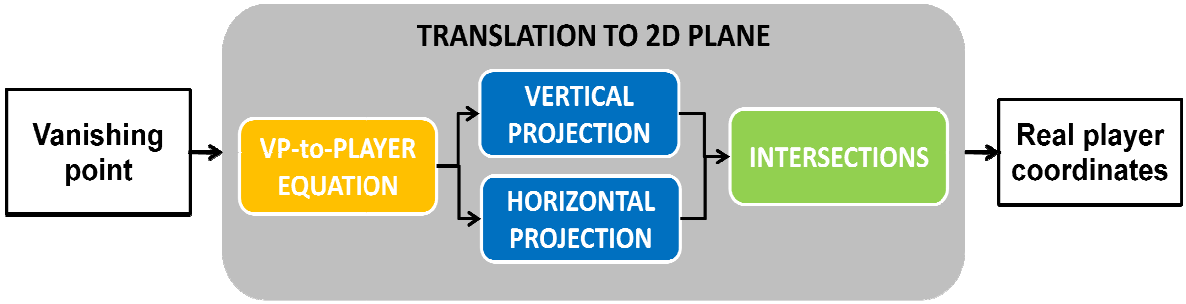

Fig. 5. 5: Sub-stages composing the vanishing point module.

After testing the module, the players positions have been correctly converted to the real world coordinates in 100% of the cases. As it will be explained in section 7.2.3, the translation to two-dimensional plane does not depend on the image. The conversion will be properly computed if the data provided by the previous module is accurate.

CHAPTER 6. EVENT CLASSIFICATION

This chapter is dedicated to explain how the low level data must be understood in order to obtain higher level video features. By integrating spatial movement information, temporal information or even audio effects is possible to characterize and therefore distinguish among different events in tennis video broadcasting (**see [13]**, **[14]** and **[15]**).

The detection and identification of these events can be so useful to build rich summaries in semantic content. The system is aimed to link the numerical image information and the symbolic description of the scene. For carrying it out, it is necessary to know the game rules and select the key visual features that will allow discriminating the events.

The following sections will define the different key features chosen and the events which can be detected from them.

6.1 Event definition

The first task to do is to define the different kind of shots that appear in a tennis match. Each one can be described in terms of visual features and sound characteristics. The main tennis events can be classified into five types, as described in following sections **[4]**.

6.1.1 Ace or non-returned service

The ace is produced when a player successfully serves and his opponent fails to return the ball. Specifically to be considered as ace, the opponent is not able to touch the ball, so it reaches the wall behind the receiver player.

On the other hand, if the opponent can touch slightly the ball but is still unable to successfully return it, this event is called non-returner service. In this case, the ball touches the net or is out of the court. In both cases, the public usually express their feelings through applauses. The software implemented in this thesis will categorize both events as the same.

6.1.2 Fault

The player who is serving fails to place the ball in the correct area of play, therefore the point cannot start. When it happens, the camera usually switches out of the court view. In terms of sound, a silence is produced from the audience when a fault is committed.

6.1.3 Double fault

The player who is serving consecutively fails in two serves. When it happens, the audience does not clap, although it can be heard an astonishment sound. This sound will probably describe a very different spectrum pattern from the applause one.

6.1.4 Baseline rally

A baseline rally is produced when a player successfully serves and the opponent successfully returns. The condition to consider the shot as baseline one is that both players remain around it until one of them fails to return. Therefore, both do not get close to the net during the whole shot.

This event can be detected easily looking for the video features, without taking into account the audio patterns.

6.1.5 Net approach

The net approach is considered when, after a player successfully serves and the opponent successfully returns, one of them or both approaches to the net with the aim of stressing the opponent. As in the previous case, it can be described enough with the visual properties. The net approaches and the baseline rallies are easier to discriminate than other cases.

6.2 Feature extraction

The thesis has been focused on the image processing part, leaving aside the audio part of the video sequences. However, it could be interesting to highlight some of them, in order to get a more general vision of the matter. The features that will be under study are detailed following.

6.2.1 Moving distance of the player

Once the player is detected, the algorithm will add the displacement of him between consecutive frames in the real-world coordinates. At the end of the shot, the total distance travelled by each player is obtained. This information will be helpful to separate the different serving events (ace, fault and double fault) and the rest of them.

6.2.2 Relative position between player and court

The results of court and player detection will allow, as explained in previous chapters, know the positioning of them in the two-dimensional plane. This fact becomes in an essential issue to differentiate among baseline shots and net approaches. It is necessary to define two regions for reaching this goal. The definition is displayed in **Fig. 6. 1**.

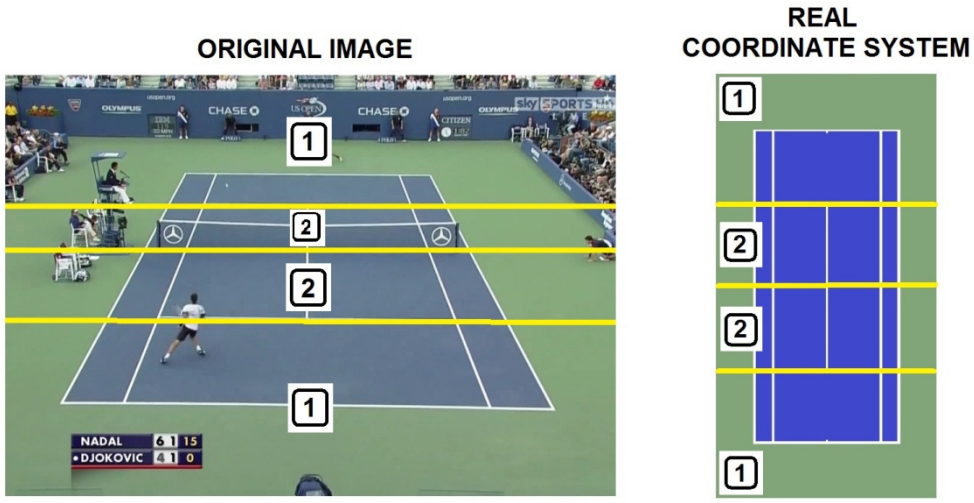

Fig. 6. 1: Region definition for event detection.

6.2.3 Length of the point

The temporal length of a shot is an easy parameter to be calculated, but it will provide important information in terms of shot classification. For example, a double fault can be distinguished from a single fault paying attention to this feature.

6.2.4 Sound effects

The sound characteristics of the video have not been analyzed during this project, but it is interesting to describe them briefly. Some audio effects, as the applauses or the cheer of the audience usually involve specific events.

For instance, audiences offer applauses after seeing good shots. Inside a possible list of good plays could be aces or long baseline rallies. From another point of view, the spectators usually remain quiet if the player makes a fault or a double fault.

While the shot is being played, the audio pattern is quite regular, following the racket hits. In other situations, as when audience is clapping or during close-ups of the players, the patterns are more irregular. **Fig. 6. 2** details this fact, showing different patterns of audio depending on the frame time positioning.

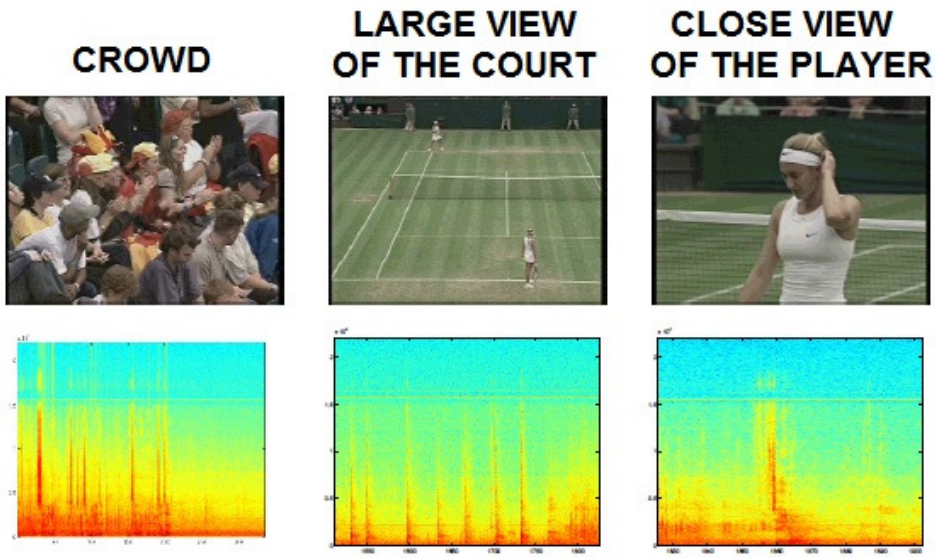

Fig. 6. 2: Examples of audio spectrum patterns.

6.3 Mapping between features and events

In previous sections it has been defined some events for being detected. Afterwards, the main audiovisual features for classifying these events have been described. Therefore, the last necessary step is to determinate the correspondences among them. **Table 6. 1** shows the mapping used by the algorithm in order to differentiate between events.

Table 6. 1: Mapping between events and features.

	Ace	Fault	Double fault	Baseline rally	Net approach
Moving distance of the player	Short	Short	Short	Medium / long	Medium / long
Relative position between player and court	Region 1	Region 1	Region 1	Region 1	Region 2
Length of the play	Short	Short	Medium	Medium / long	Medium / long
Sound effects	Applause	Silence	Silence	Applause	Applause

Using the four features detailed below, it could be possible to detect five possible events. Nevertheless, taking into account that this project only deals with the visual characteristics, three events can be finally classified: Baseline rally, net approach and serving actions (composed by ace, fault or double fault which will not be distinguished).

CHAPTER 7. EXPERIMENTAL RESULTS

This final chapter is dedicated to show the results obtained. The full system has been tested using three tennis video matches, each one belonging to different type of surface: hard court, grass and clay. After describing the video database which has been used, the performance of each stage is detailed. After that, the results in terms of event features will be shown. Finally, the computer time results will be analysed.

7.1 Features of the used database

A video database composed by shots of three different surfaces has been employed. The main characteristics are specified in **Table 7. 1**.

Table 7. 1: Database specifications.

SURFACE	TOURNAMENT	PLAYERS	SOURCE	IMAGE RESOLUTION	FRAME RATE
Hard court	US Open 2010	Nadal-Berdych	Sky Sports HD	1280 x 720	30 frames/s
Grass	Wimbledon 2010	Nadal-Djokovic	Cuatro	1024 x 576	25 frames/s
Clay	Roland Garros 2010	Soderling-Nadal	TVE 1	1024 x 570	25 frames/s

The selected broadcast videos have not been recorded by the same tennis filmmaker, so the camera movements are significantly different. This fact will be so useful to evaluate the program operation.

7.2 Modules performance

The different modules of the algorithm have to be tested. As their distribution is in cascade, the failures or errors caused in the first stages are critical in the following ones. For this reason the analysis of the results is done by parts.

7.2.1 Court detection

The court detection is the first important module. Its obtained performance is described in **Table 7. 2**.

Table 7. 2: Court detection results.

COURT DETECTION				
Surface	**Tested shots**	**Correct**	**False**	**Correct (%)**
Hard	45	31	14	68.9%
Grass	15	9	6	60.0%
Clay	15	11	4	73.3%
TOTAL	**75**	**51**	**24**	**68.0%**

The percentage of non-correct detections of this module is so high. The errors are caused by so many reasons, as changes in illumination, strong shadows or partial occlusions of the court. For this reason, it would be interesting to pay special attention on this part for improving its performance for future enhancements.

7.2.2 Player detection

The player detection stage starts when the court is detected. If the previous detection of the court has been wrong, the player's module will fail too. Therefore, the following results (**Table 7. 3**) are supposed to be after right court detection.

Table 7. 3: Player detection results.

PLAYER DETECTION				
Surface	**Tested shots**	**Correct**	**False**	**Correct (%)**
Hard	31	28	3	90.3%
Grass	9	8	1	88.9%
Clay	11	10	1	90.9%
TOTAL	**51**	**46**	**5**	**90.2%**

7.2.3 Coordinates conversion

In the same way, the coordinate conversion module will work properly if the player detection has been done successfully. This stage does not depend on the image itself, but it just depends on the data from the previous module. For this reason, whether the players have been properly located, the conversion will be correctly done (**Table 7. 4**).

Table 7. 4: Coordinates conversion results.

COORDINATES CONVERSION				
Surface	**Tested shots**	**Correct**	**False**	**Correct (%)**
Hard	28	28	0	100.0%
Grass	8	8	0	100.0%
Clay	10	10	0	100.0%
TOTAL	**46**	**46**	**0**	**100.0%**

7.3 Overall performance

The provided software is able to compute the distance travelled by each player, as well as the time duration of the shot and whether any of them has approached to the net. With this information, the user will be able to decide what kind of shot has been produced. Therefore, the final decision is taken by the user, not by the algorithm.

If the algorithm computes the three modules with no errors, the provided information will be accurate enough to describe the shot. On the other hand, if one of them fails, the final information will be totally erroneous. After testing the video database, **Table 7. 5** shows the overall results.

Table 7. 5: Full system results.

FULL SYSTEM			
Surface	**Tested shots**	**Extracted info is correct**	**%**
Hard	45	28	62.2%
Grass	15	8	53.3%
Clay	15	10	66.7%
TOTAL	**75**	**46**	**61.3%**

From all the tested shots, the 61.3% has obtained truthful information. The most critical part is the court detection, because is there where the majority of the errors have been produced.

As commented before, the decision about the final classification of a shot is taken by the user, not by the algorithm. In order to get a threshold value for the user who is classifying, the following graphics are helpful to do that. **Fig. 7. 1**

shows the average distance that a player covers during the shots, obtained after checking the database. Defining a threshold (red line), it is possible to differentiate easily between a serve action and the rest of shots.

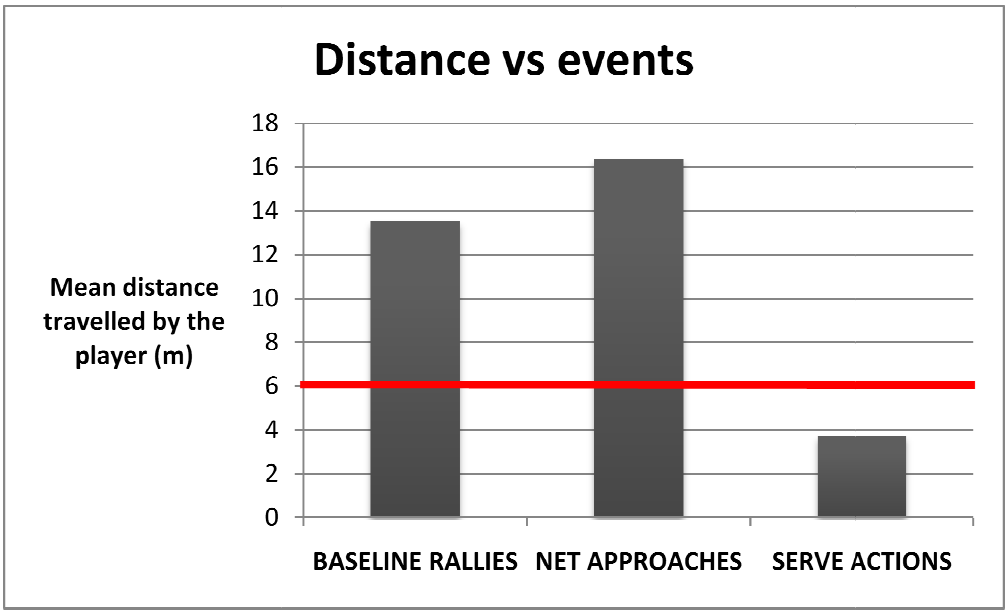

Fig. 7. 1: Statistics. Mean distance covered by players depending on the shot.

Fig. 7. 2 shows the mean time duration of the shots. In the same way, defining a threshold (red line), it is also possible to differentiate among serve actions and the other ones. Hence, using both thresholds, this kind of shot will be isolated simply.

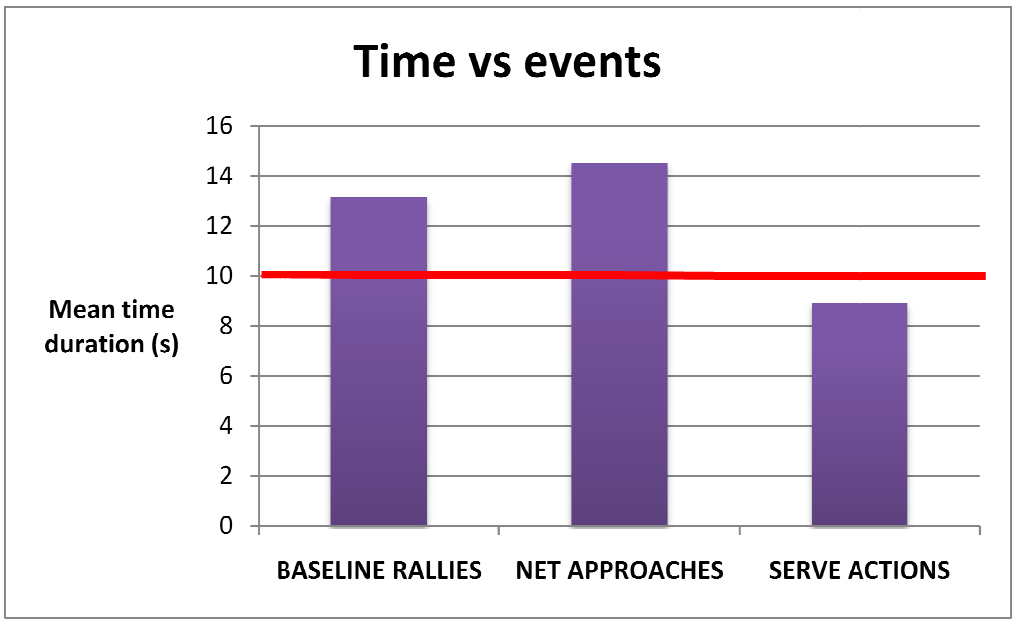

Fig. 7. 2: Statistics. Mean time duration depending on the shot.

The distance and time features are not useful for discriminating between baselines and net approaches (as it was detailed in **Table 6.1**). In order to distinguish among them, the 'net approach' indicator will be a key factor.

7.4 Computing times

This section is aimed to detail the timings of each module of the algorithm. The absolute time values are not so relevant, because they largely depend on the computer features where the code has being processed (CPU, RAM...). Moreover, the video quality is another key factor over this issue.

However, it is interesting to compare the timings in relative terms between the different stages of the program, in order to know the parts which carry more computational load.

7.4.1 Match segmentation

The frames which are labelled as non-court view frames will only delay the time dedicated to detect them, because there is any additional operation done over them. On the other hand, the court view frames will spend the time to be labelled and all the later processing (**Fig. 7. 3**).

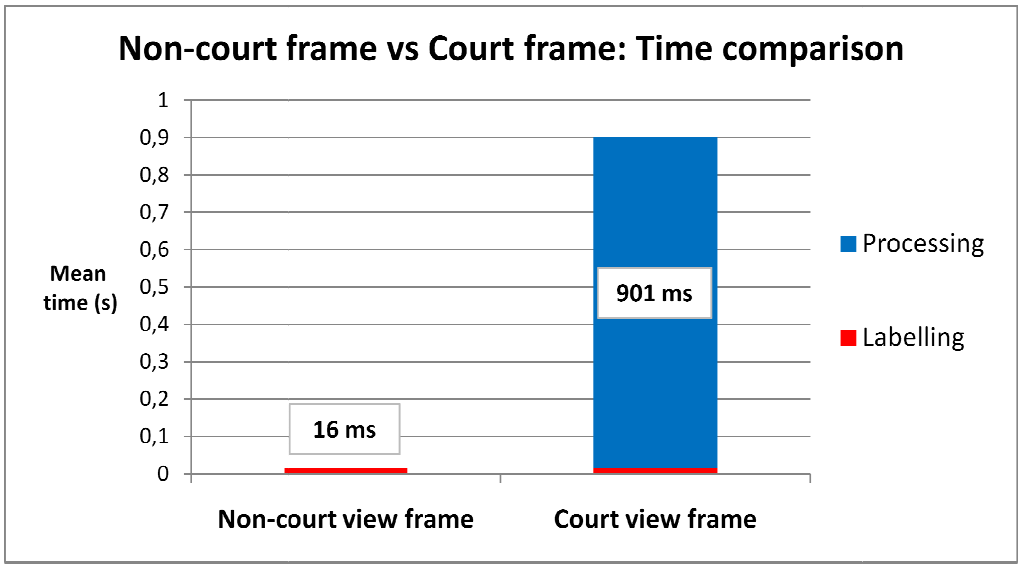

Fig. 7. 3: Non-court frame vs Court frame. Processing time comparison.

7.4.2 Court detection

The court detection is divided in four sub-stages. The search of candidate points and the Hough transform take practically the 100% of the computational cost of this module. The two final steps are really quick to process. **Fig. 7. 4** brings out this fact.

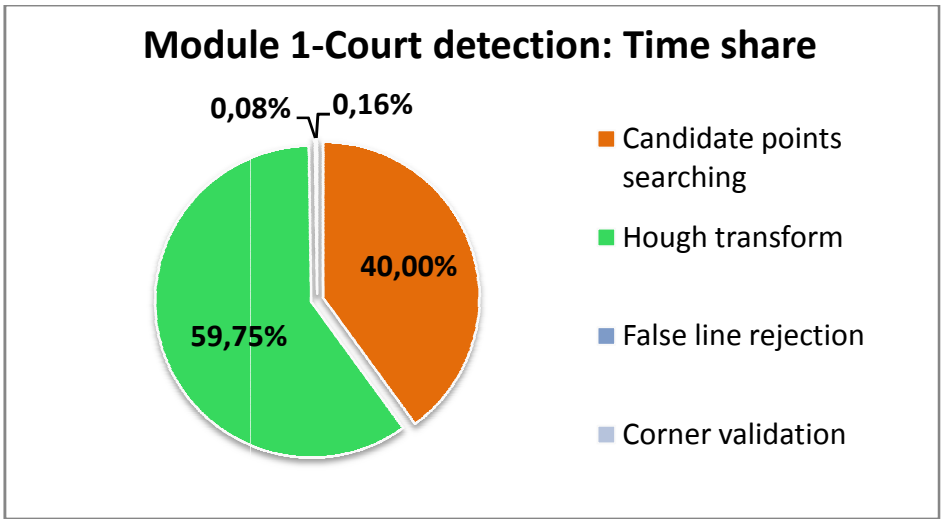

Fig. 7. 4: Time share between Module 1 sub-stages.

7.4.3 Player detection

In the second module, the player detection, two types of frames can be processed. The first frame which belongs to the court camera view will not have information about previous player localizations.

Therefore, it is easy to think that these frames will spend more time to find the players, as it is reflected in **Fig. 7. 5**. The following ones, in tracking mode, will be faster due to save in calculations.

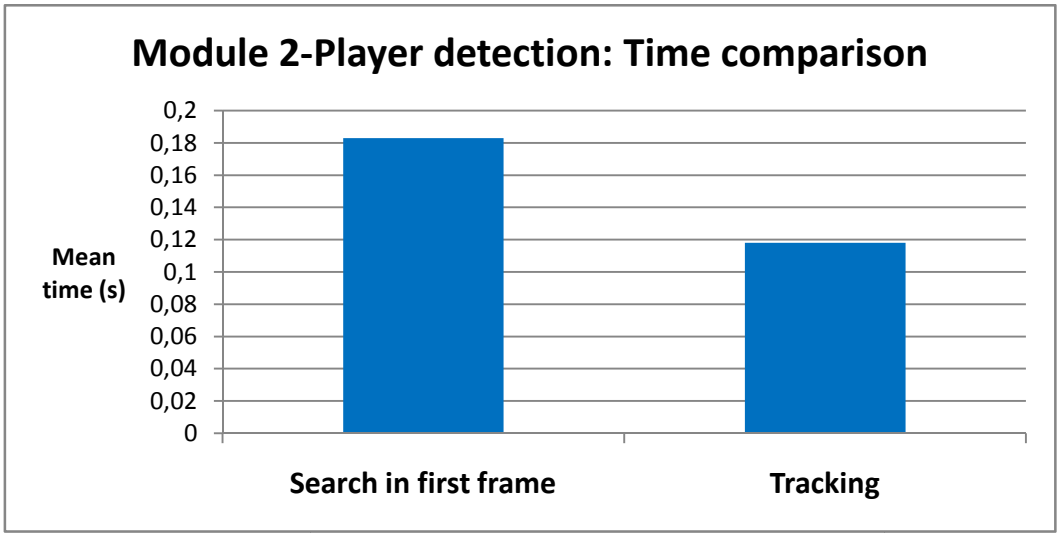

Fig. 7. 5: First frame vs Tracking frame. Processing time comparison.

7.4.4 Coordinates conversion

Coordinates conversion module is formed by two sub-stages. Whereas the vanishing point calculation only takes the 31% of the module time, the translation to two-dimensional plane spends almost the 69% (
Fig. 7. 6). The reason is that in the second sub-stage, more intersections have to be computed than in the prior sub-stage.

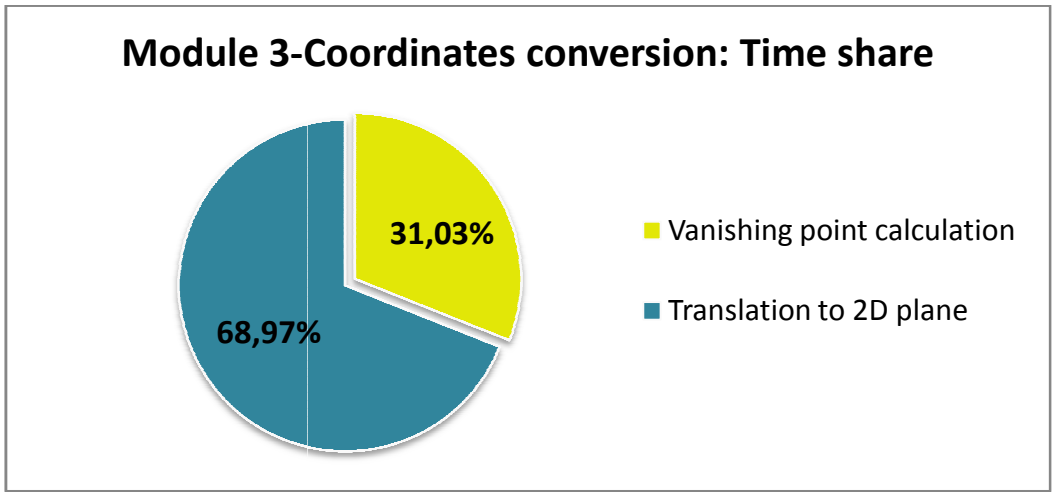

Fig. 7. 6: Time share between Module 3 sub-stages.

7.4.5 Full system

Finally, the whole system time share is shown in **Fig. 7. 7**. The most part of the time is dedicated to detect the court. It is important to explain that this module has been programmed to calculate the position of the court in each frame. In tournaments where the camera movements are practically null (as US Open and Roland Garros), this step can be computed once and the information can be re-used in subsequent frames, saving too much in computational cost.

However, in other courts as Wimbledon, the camera movements are so large and frequent, forcing to re-calculate the positions constantly. With the aim of proposing a software solution as more general as possible, the court position is calculated in all the frames, although it is not the best option in terms of efficiency.

The player detection time share reaches almost the 29%. This percentage would grow until 38% if the tracking mode was disabled. The coordinate conversion module only covers the 0.01% of the whole system time.

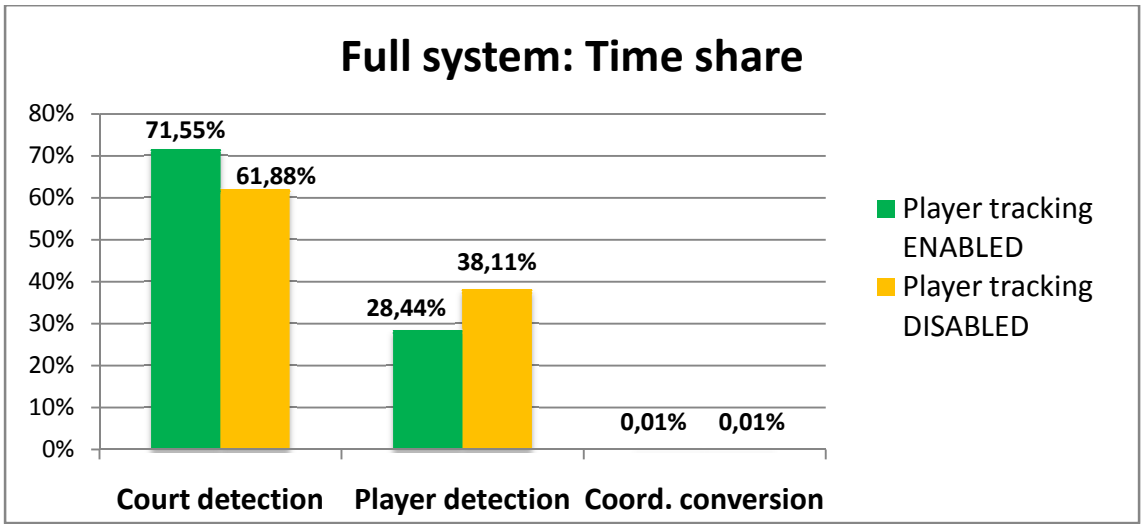

Fig. 7. 7: Time share between the three modules of the full system.

In summary, the software implemented during the thesis takes around 900 ms for processing a frame. Obviously, this computing speed is not enough for using the program in real-time. For achieving this goal, the frame time may be around 40 ms, which can be perfectly got optimizing the code and using a better computer.

Analysing the results in performance as well as in computing times, it can be deduced that the court detection module is the most critical part. Its lower efficiency causes that the overall performance decreases considerably. This is a point that may be considered in future enhancements.

CONCLUSIONS

The aim of the project was implementing an algorithm able to detect certain events in tennis broadcasting videos. It must work properly in any court surface, in order to be independent from the tournament where the match has been played.

The final proposed software provides useful information to classify three kinds of events: baseline rallies, net approaches and serve actions. Although the serve actions can be divided in some sub-events (aces, faults, double faults...), this distinction cannot be considered without using the audio signals. Therefore, it would be necessary to work in this way in future and improved versions.

The results obtained in terms of players positioning, distance travelled and shot time are consistent in the database used, which is limited (around seventy shots have been tested). However, it would be interesting to check a wider data-base that covers more special cases, as changes of illumination, other kind of surfaces, etc.

In terms of computational efficiency and processing time, nowadays the algorithm cannot be used in real time, but it was not the initial goal of the thesis. During the most part of the time, the program is computing operations related with the court line detection. Obviously, it is not necessary to look for them every frame, because the camera is almost fixed.

Nevertheless, in some tournaments the camera is constantly moving from left to right while following the action. Therefore, it is necessary to re-calculate the corner positions frequently. Because of that, the algorithm has been configured to work independently from the camera movements, resulting in some efficiency losses. For future lines, if the code is correctly optimized and the computer features are suitable for doing this task, the software will be perfectly able to process the videos in real time.

It is important to clarify that the project has worked through the three feature level defined in **Fig. 1.1**. First, some low-level features have been extracted from the images, as colour characteristics, line detection, etc. From of all them, mid-level models have been obtained (player trajectories and positioning). Finally, a high-level semantic is got, specifically the basic event differentiation. Other high-level semantics concepts would be used, as highlights extraction or tactic analysis. However, they have not been considered during the development of this thesis.

For getting more information from the shots, an additional functionality has been done, the serving speed calculation. The detailed information about this part of the algorithm is attached in the appendix A.1. Finally, the visual user interface of the program is enclosed in appendix A.2.

REFERENCES

[1] Changsheng Xu, “Sports video analysis: Semantics extraction, editorial content creation and adaptation”, *Institute of Automation, Chinese Academy of Sciences* (2009).

[2] P. Salembier, “Overview of the MPEG-7 standard and of future challenges for visual information analysis”, *EURASIP Journal on Applied Signal Processing,* Volume 4, pp. 1-11 (2002).

[3] MPEG-7 Overview
http://mpeg.chiariglione.org/standards/mpeg-7/mpeg-7.htm

[4] Ming-Chun Tien, “Event detection in tennis matches based on video data mining”, *Department of CSIE, National Taiwan University* (2008).

[5] Ekin A. and Murat A., “Robust dominant color region detection and color-based applications for sports video”, *Department of Electrical and Computer Engineering, University of Rochester, NY* (2003).

[6] Rea N., Dahyot R. and Kokaram A., “Classification and representation of semantic content in broadcast tennis videos”, *Department of Electronic and Electrical Engineering, Trinity College Dublin* (2005).

[7] Han J., Farin D., With P. and Lao W., “Automatic tracking method for sports video analysis”, *Eindhoven University of Technology*.

[8] Han J., Farin D. and Peter H., “Multi-level analysis of sports video sequences”, *Eindhoven University of Technology*.

[9] MathWorks Product Documentation
www.mathworks.com/help/toolbox/images/ref/edge.html

[10] Image processing learning resources
http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm

[11] Conaire C., Kelly P., Connaghan D. and O’Connor N., “Tennis sense: A platform for extracting semantic information from multi-camera tennis data”, *Centre for Sensor Web Technologies, Dublin City University.*

[12] Farin D., “Robust camera calibration for sport videos using court models”, *Embedded Systems Institute, Eindhoven University of Technology*.

[13] Kijak E., Gravier G., Oisel L. and Gros P., “Audiovisual integration for tennis broadcast structuring”, *Thomson multimedia R&D, France.*

[14] Kolonias I., Christmas W. and Kittler J., “Automatic evolution tracking for tennis matches using an HMM-based architecture”, *Centre for Vision, Speech and Signal Processing, University of Surrey, Guildford.*

[15] Dahyot R., Kokaram A., Rea N. and Denman H., “Joint audio visual retrieval for tennis broadcasts”, *Department of Electronic and Electrical Engineering, Trinity College Dublin*.

APPENDIX

A.1 Serving speed calculation

As additional information, the serving speed of the player that is serving could be useful information to be indexed in a video characterization. Although this information will not be used to classify the kind of shot, it can be helpful to look for the fastest serves of a certain player and to obtain some statistics.

A.1.1 Ball detection and tracking

In order to compute the serving speed it is necessary to detect the ball. For doing it, a search window is defined. The position of this window is defined in **Fig. A. 1**.

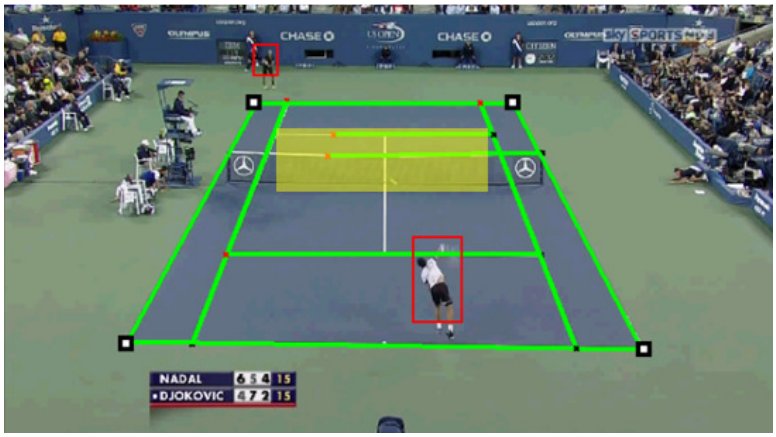

Fig. A. 1: In yellow: Search window for detecting the ball.

From the beginning of the shot, the software will be waiting until the ball enters by first time in the yellow area. The method to know when this issue happens is the subtraction, a method which has already been used in other parts of the code.

The subtraction among yellow regions of consecutive frames is computed. Thus, when the ball is not inside, the result is practically zero. It means that no change has been produced in this area between frames. When the result of the subtraction become higher, the ball will be entered and its location will be registered. During the following frames, the ball will remain in the yellow area but changing its position, therefore the subtraction result will be still high.

Once the ball leaves the region, the result will return to low values, so the serving speed module can be disabled until the next shot (**Fig. A. 2**).

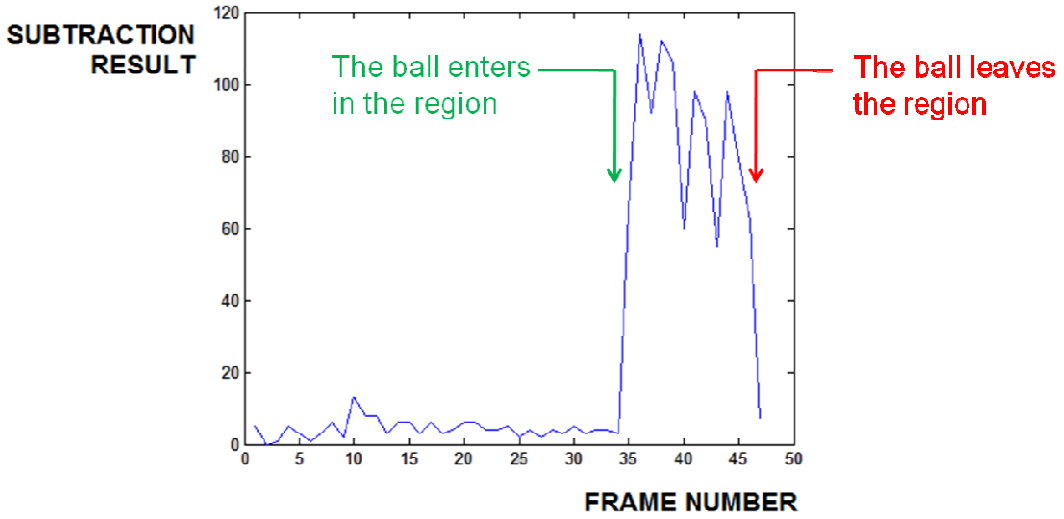

Fig. A. 2: Ball detection and disabling of the serving speed module.

A.1.2 Speed estimation

After knowing the positions where the ball was initially and finally detected and the frame numbers in which these events have happened, it is easy to compute the speed of the serve.

The difference in pixels among the initial and final positions of the ball has to be calculated and after that, translated into real meters. The speed is obtained following the (**Eq. A.1**) and (**Eq. A.2**).

$$\text{Time (h)} = \text{frame\_distance} \times \frac{1\text{ s}}{25\text{ frames}} \times \frac{1\text{ h}}{3600\text{ s}} \qquad \textbf{(Eq. A.1)}$$

$$\text{Speed}\left(\frac{\text{Km}}{\text{h}}\right) = \frac{\text{Ball\_distance (Km)}}{\text{Time (h)}} \qquad \textbf{(Eq. A.2)}$$

The method implemented for calculating the speed has been a first approach, so it has to be improved for giving credible information and for being robust independently of whatever condition.

A.2 User final interface

A.2.1 Non-court view frame

When the frame is labelled as non-court view one, the user interface shows the tournament logo and the current frame (**Fig. A. 3**).

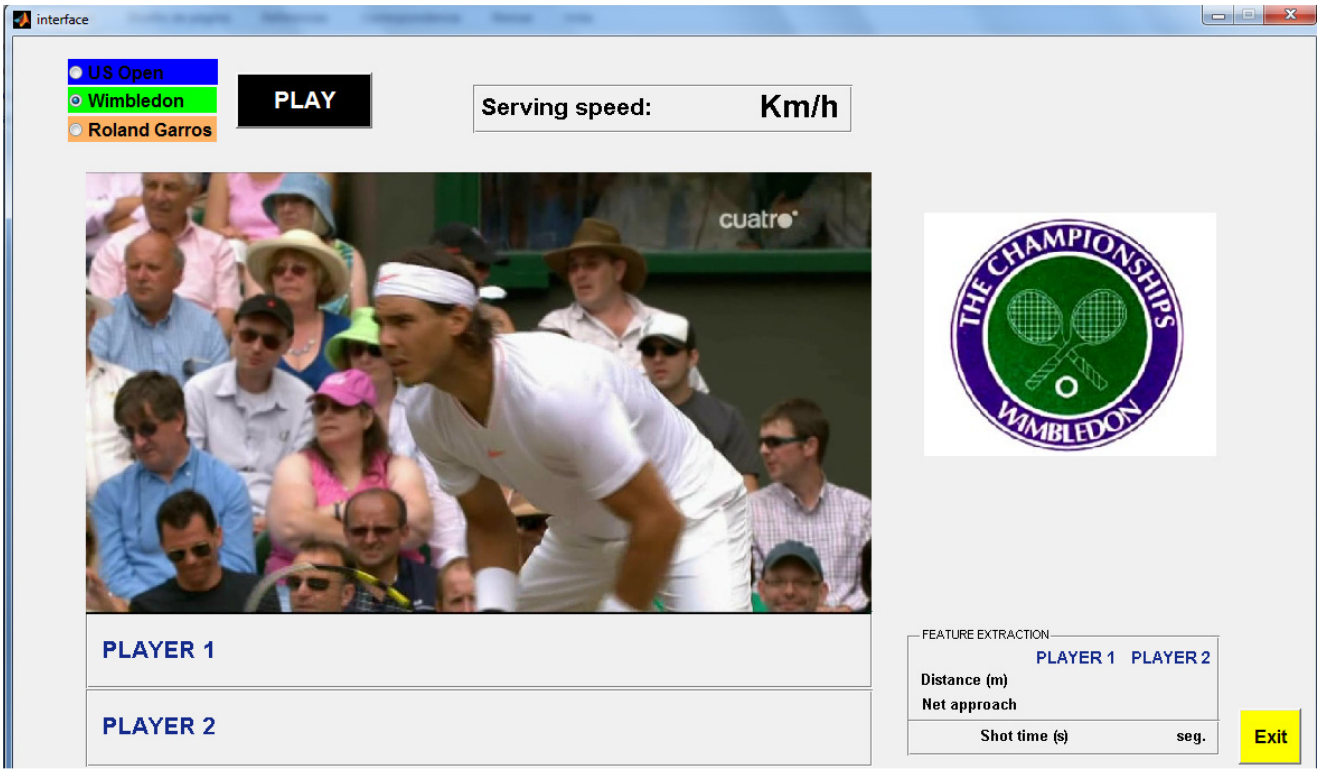

Fig. A. 3: Non-court view frame: Visual interface.

A.2.2 Court view frame

When the frame is labelled as court view one, several information is displayed at the user interface (**Fig. A. 4**).

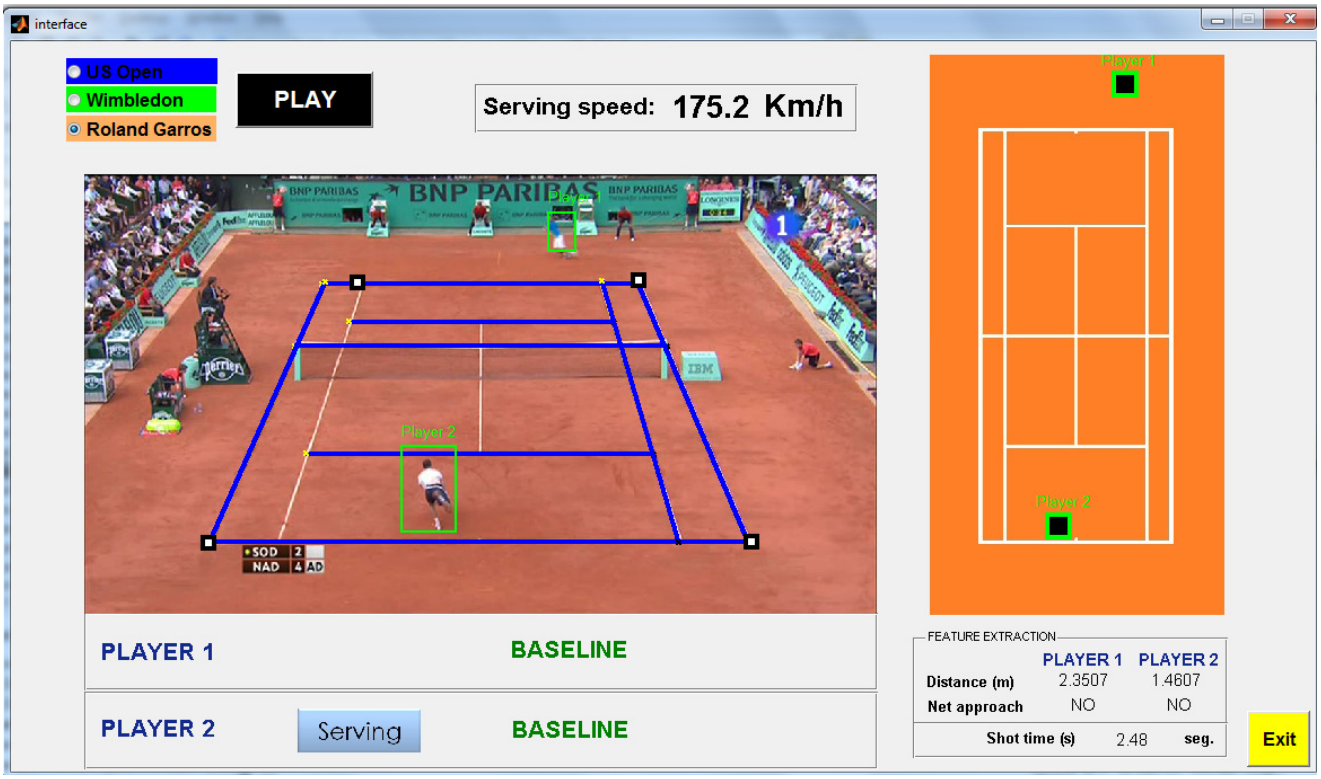

Fig. A. 4: Court view frame: Visual interface.

The box situated in the upper part informs about the speed at which the player has served. The middle part shows the current frame (highlighting the player and the court lines) and the two-dimensional plane where the real position of the players is indicated.

The bottom of the interface is formed by the useful information to classify the shot. On one hand, the left part tries to inform about live data of the current situation, for instance the player who is serving or where the players are located at this moment (baseline or close to the net).

Finally, the right-bottom side is composed by the final information that the person who is analysing the shot would have to look at the end of it. The distance travelled by each player and the total duration of the shot are shown here. Furthermore, it is specified if the player has ever approached to the net during the course of the point.

A.3 MPEG-7: XML label example

```
<Mpeg7>
 <Description xsi:type="SemanticDescriptionType">
  <Semantics>
   <Label>
    <Name> Net approach </Name>
   </Label>
   <SemanticBase xsi:type="AgentObjectType" id="Rafael Nadal">
    <Label href="urn:example:acs">
     <Name> Rafael Nadal </Name>
    </Label>
   </SemanticBase>
   <SemanticBase xsi:type="AgentObjectType" id="Roger Federer">
    <Label href="urn:example:acs">
     <Name> Roger Federer </Name>
    </Label>
   </SemanticBase>
   <SemanticBase xsi:type="EventType">
    <Label><Name> Net approach </Name></Label>
    <Definition>
     <FreeTextAnnotation> Nadal approaches to the net and knocks the
     ball for winning the shot
     </FreeTextAnnotation>
   </SemanticBase>
  </Semantics>
 </Description>
</Mpeg7>
```