uc3m | Universidad Carlos III de Madrid

University Degree in Telecommunications Technologies

2021-2022

Bachelor Thesis

"Multi-player tracking and statistics for team sports activities"

Álvaro Arenzana Mínguez

Tutor:

Francisco Javier González Serrano

Leganés, June 2022

SUMMARY

Video analysis is a technique that is increasingly present in our day to day. Whether it is to analyze sports, traffic or behaviors, it is a tool that allows great flexibility when it comes to application.

The main objective of this project is to develop a tool capable of detecting, tracking and positioning an athlete in the training field. Once located, a series of parameters will be generated. This tool will work using video analysis techniques and will have a maximum time to execute.

For this, the work has been divided into four modules: detection and classification, tracking, camera calibration and homography, and data representation. Each of these blocks will be studied independently and the different available methods will be analysed.

Next, the results for each of these blocks will be evaluated and the best methods will be defined. Once the modules have been tested separately, they will be put together and general tests of the system will be done.

This paper is a description of the definition of the problem, state of the art, design and results of the complete system.

Keywords: Video analysis, detection, classification, tracking, homography, statistics, system, person.

DEDICATION

I would like to thank in this section all those who have supported and accompanied me during this years.

First of all I would like to thank Francisco Javier González Serrano for his support and dedication during this project and for guiding me in the last stage of this degree.

Secondly, I would like to thank my family for all the support received during these years, Alberto, Elsa, Judit, Sara... You are everything to me. I would especially like to highlight my brother Pablo, the cornerstone of my life and the main reason for the development of this project.

Thirdly, to Jose Luis Devesa for trusting in this project and for putting me in contact with different people from the world of football.

Finally, to my classmates and my friends for having been part of this stage of my life.

To all of them, thank you.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. MOTIVATION AND INTRODUCTION

1.1. Motivation

The influence of new technologies such as Big Data or data analysis have changed drastically people lives in the last few years. Social networks or e-commerces are two clear examples of how data are used in order to optimize processes. This optimization is becoming extremely important in sports. Massive data collection is a must almost in every professional athlete in order to reach its top.

At the beginning of this project, several conversations with different people from sports world have been held. These people, who belong to different football clubs, have shown a big interest about this type of applications. Specially, they feel the lack of a program that can help trainers when practising tactics, focusing on real time analysis.

This project is also encouraged by the open problem that exists nowadays in this area. Sports teams are able to collect a wide range of data and actually, it is one of the industries that is making an special effort toward this. But on the other side, the main problem is that all of these data take a lot to be analysed and are not applied correctly. So the aim to get things changed in this field is something that pushed this project to be done.

The main motivation of this project is to make a new tool able of locating a person in a certain area by video analysing. The field of knowledge to develop during this project is video analysis. Several techniques about homography and person detection will be studied. After that, they will be joined for a same goal in order to get the real application and get stats from the images. Specifically, this project will use football as it is the easiest to get data from, even though it could be applied to many other sports.

This type of video analysis is already being used by other devices such as drones, self-driving cars and others. Although this project has being developed in the field of sports, it could be useful for many other applications like traffic flux, supermarkets, etc.

1.2. Introduction

Statistics in sports have been determining in the last few years in order to improve athletes performance. The world of journalism and sports is continuously trying to find new ways of obtaining data and generate new stats. These stats can be used either for athletes to improve or to be published.

Traditionally, analysts dedicated to sports took notes manually on paper, and after that, data were analyzed very briefly. During the last decade, many different programs have been developed. It was possible to registere different events while watching a video of the match such as passes, shots, tackles... After collecting all these data, it is possible to add it to a database and generate different graphics, estimations or even draw over the video.

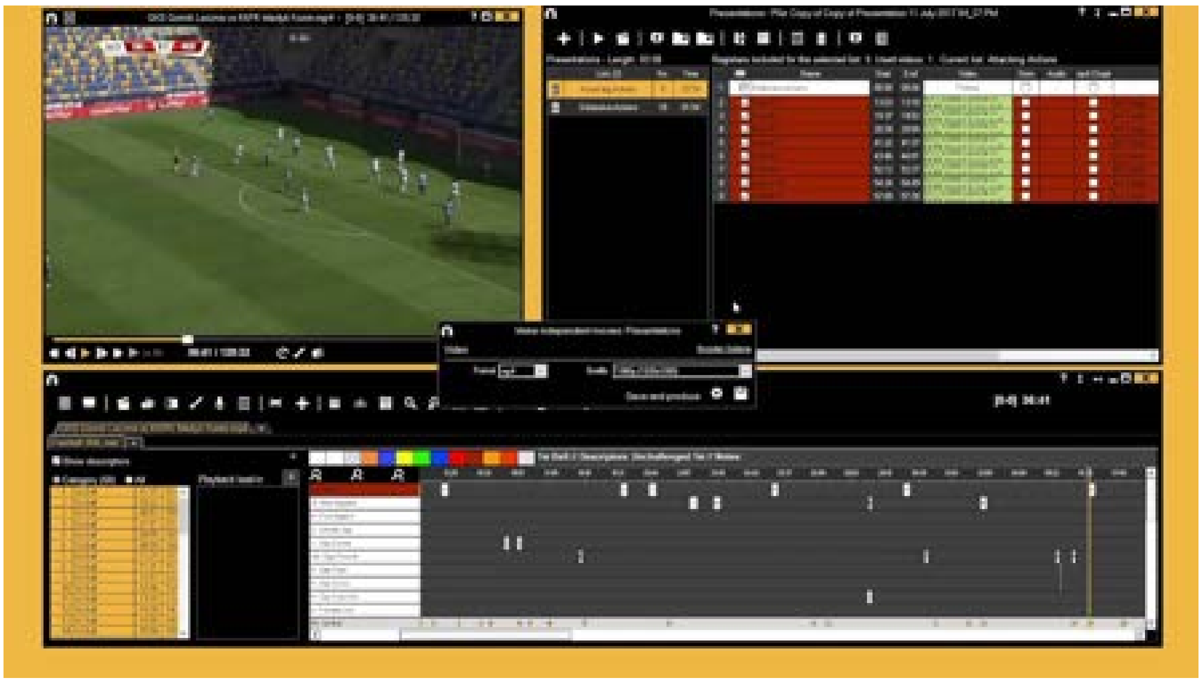


Fig. 1.1. Screenshot of the data analysis software NacSport [1]

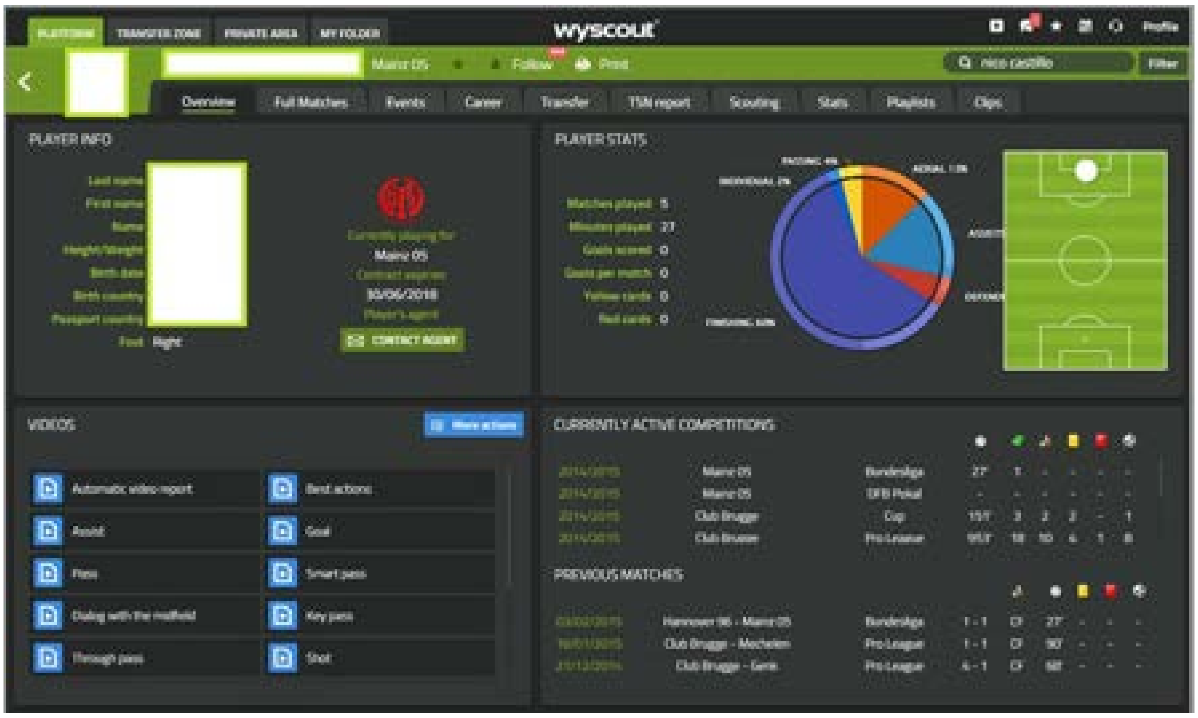


Fig. 1.2. Screenshot of the data analysis software WyScout [2]

This type of software does not use any video processing. Usually, it uses a timeline to mark down events while the user is watching a video. Because of that, this type of software requires a lot of handmade work, being inefficient and spending lots of hours.

Some examples of this type of program are:

- NacSport
- LongoMatch
- InStats Sport
- Soccer Lab
- Smart Soccer Coach
- WyScout

It is important to mention that most of them are payment programs, making them not accessible to everyone. Furthermore, some of them are not open to general public but are only open for professional teams.

In the last five years, new tools and methods have been developed. With the growing interest of people in sport stats, it was necessary to collect even more data in a faster way. Spectators expected to get more data almost in live time. In addition, it was necessary to show all these statistics in an easy and interactive way, so most of the public could understand it.

At that moment, researchers started to think about a way of applying technology to this type of problem. Several methods with different type of sensors were tested. One of those that became interesting, specially for football, was GPS tracking.

GPS tracking started as a way of locating players on the field. After the first versions, it got evolved by the adding of heartbeat and oxygen measurement. For this system to work properly, it is necessary to use it in an environment with great GPS satellite coverage. This GPS is put in the players back with a vest and data can be downloaded to a mobile app by WiFi connection.

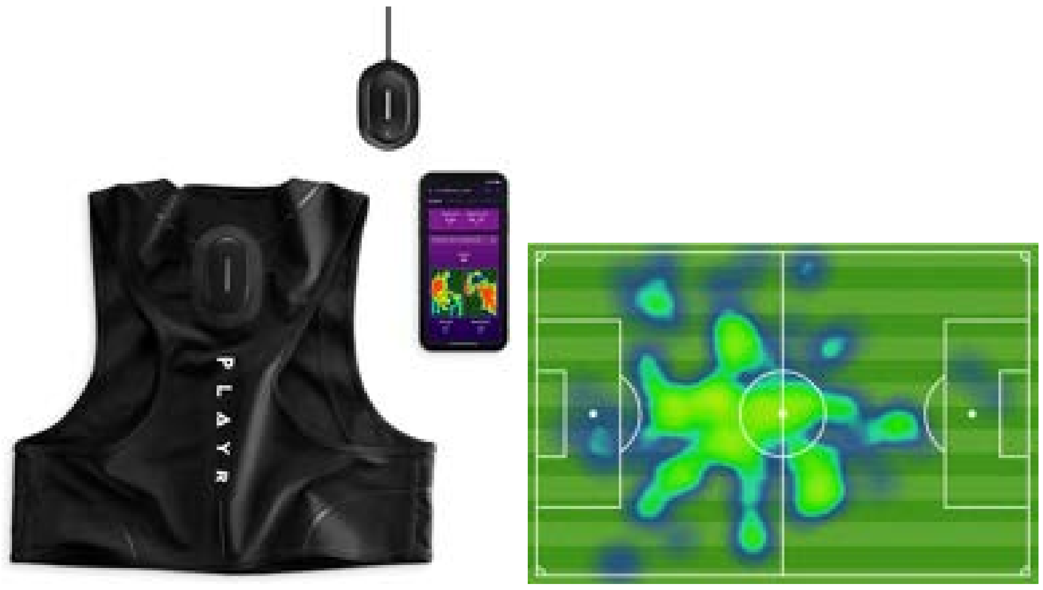


Fig. 1.3. GPS system images

This system is still being used today, and it is getting even more popular, but has passed to a second plane due to video analysis. Even though, every big football clubs use it every day. Some examples of this type of system are:

- Catapult Sports
- STATSoprts

- Oliver

Video analysis programs for sports started to be tested for American football. The main goal was to mark down different areas of the pitch in order to help study tactics for future matches. The main method used was colour tracking and it took few days to be analyzed.

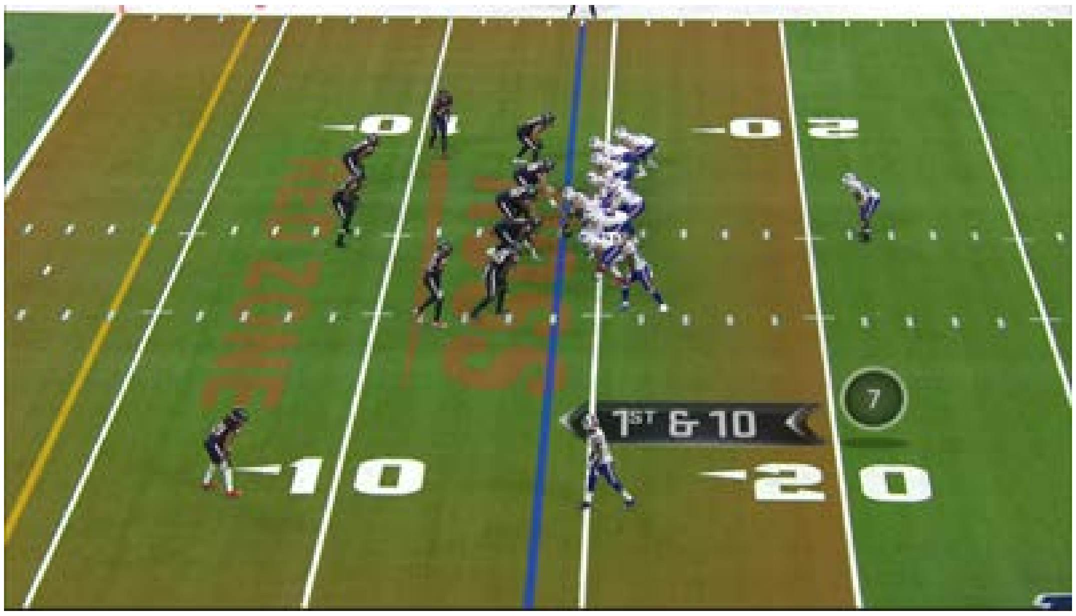


Fig. 1.4. Screenshot of the American football analysis software [3]

However,years after with the evolution of technologies and in particular, graphic cards, it was possible to locate the players on the pitch. The processing capacity of graphic cards multiply almost by ten processing capacity of traditional processors. This significant improvement make possible to compute more data by frame, which allows to use neural networks and other complex methods to detect and track athletes.

This evolution led to the type of software used nowadays. These programs can detect and follow several players all over the field by using neural network and tracking. Aditionally, processing time has decreased a lot compare to previous generation of programs. It is possible to get stats live and after a few minutes get even more complex cross data and graphics. For making this possible three main steps can be distinguished. Firstly, several cameras must be positioned around the stadium in order to cover all the field area. Secondly, videos are analyzed individually by using video detection techniques. Finally, these videos are processed and merged with the goal of getting a better estimation about the player position on the pitch.

If talking about systems capable of obtaining stats by video analysis, it must be mentioned Mediacoach software. Mediacoach was first presented in the year 2011 as a manual stats tool created jointly by LaLiga and Mediapro group, but it was after 2016 when it started to make a difference. Its use is very restricted since only it is offered to professional clubs and its prices are quite high.

Mediacoach uses wide angle cameras placed in the middle of the stadium to cover all the field. These cameras can stream video and save it for post processing. They are carefully calibrated depending of the stadium as well as they need periodical revisions.

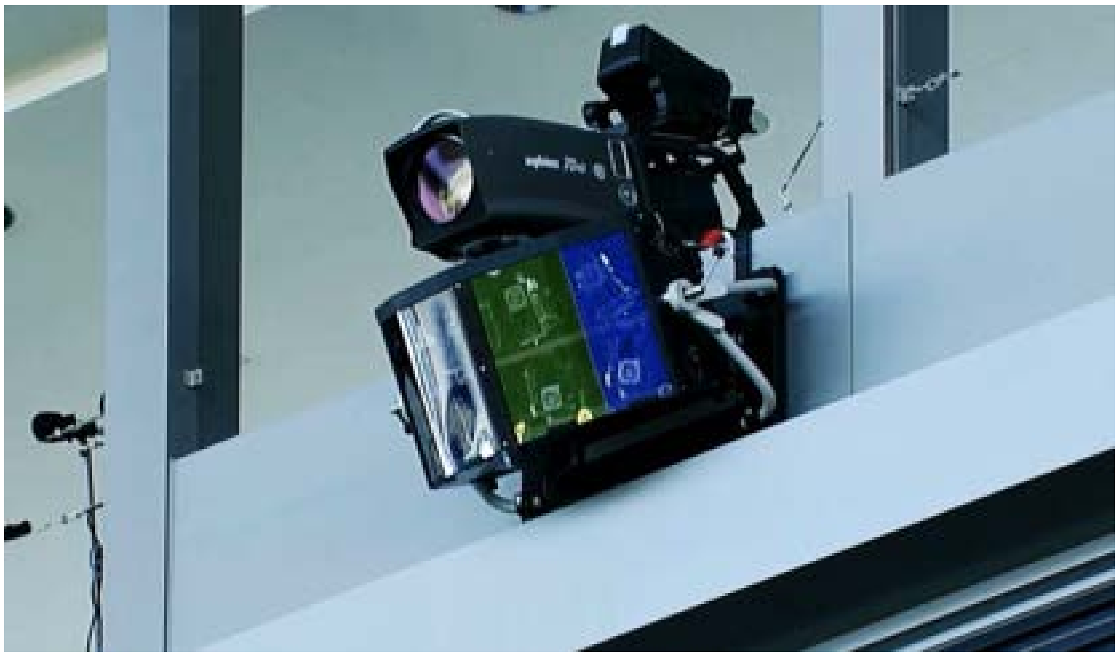


Fig. 1.5. Mediacoach camera system [4]

At the beginning of this system, Mediacoach had one server for each club, which was placed at each team office. One day after the match, analysts had to download data from that server, which was quite troublesome. Nowadays, these computations are held in Azure cloud, making work easier, faster and more accessible from any platform. Each club has its account and it is possible to use the software from the day the season starts until the day it ends.

As every technological development, it suffered lots of bugs in its begin. The program could not detect occlusions and tracking got lost in the majority of cases. Due to the implementation of neuronal networks and AI (Artificial Intelligence) these problems got solved over the years.

The stats that Mediacoach can provide go from position, speeds, and distances to team strategy and where shots go. These are offered as graphics and edited video. This allows teams to watch and analyse its own matches and other teams matches in a very precise way. Also this tool is very helpful in order to prevent lesions and keep athletes performance at its best.

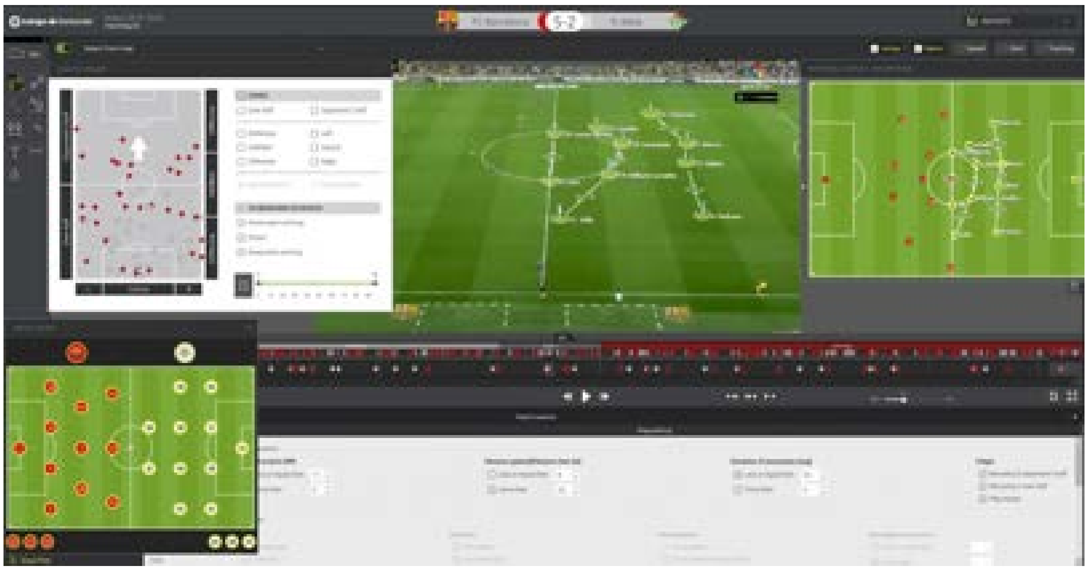


Fig. 1.6. Mediacoach screencapture software [4]

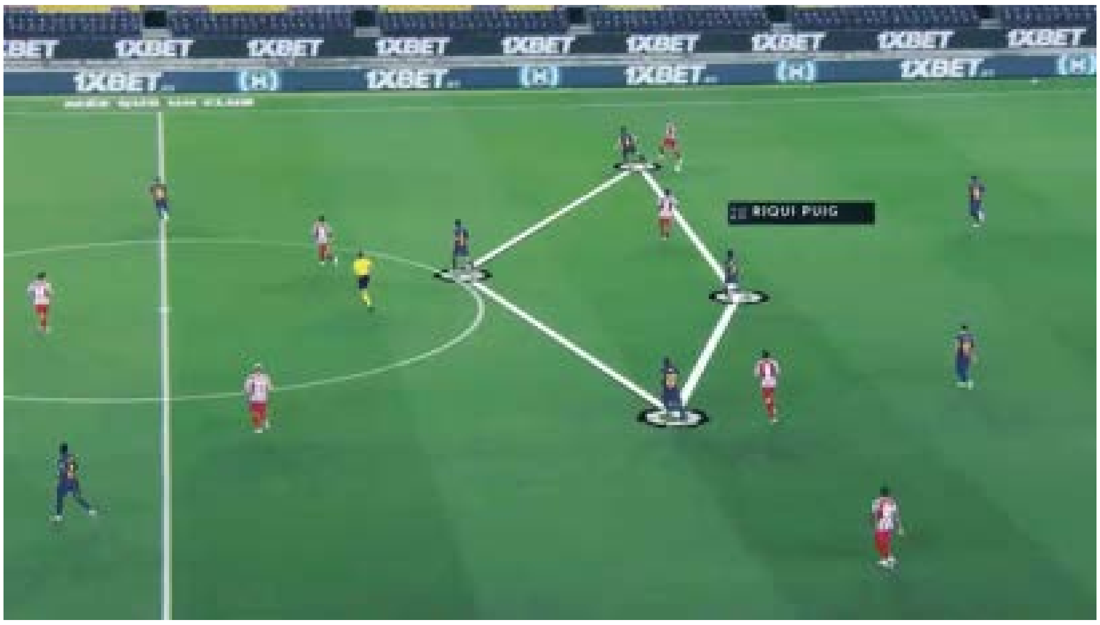


Fig. 1.7. Mediacoach screencapture [4]

1.3. Social and economic impact

Since this project is in the early stages of development and research, a larger team of programmers and researchers with knowledge of video analysis is needed. In addition, an initial investment will be necessary in different video cameras as well as in servers with a large number of graphics cards (GPUs) so that the project can be scaled to monitor an entire computer.

Likewise, this project covers an initial development of the system, and therefore, only the following minimum requirements will be necessary:

- Computer with the following characteristics:
  - Intel i7 processor
  - Nvidia graphic card with at least 2Gb of memory
  - At least 8 RAM
- Camera or mobile with tripod with a minimum quality of 1080p. It is also possible to use a drone as it allows more flexibility when taking better angles.

It should be remembered that this project is motivated to provide athletes without many resources with an automatic video analysis tool. The sales strategy would be focused on selling cheaper and for a wider audience than the current existing tools such as Media-coach, which is only offered to first and second division teams.

The fact that this system is oriented to the training of the general public. This will provide teams and coaches with few resources a work tool with which to work regardless of the situation in which they find themselves. It also improves the quality of the coaches' work, and as it is a portable and adaptable tool it can be applied to lower categories, thus improving the learning of children and young people.

1.4. Regulatory framework

As far as legislation is concerned, this project is not very limited. The biggest obstacle that could be found is related to video recording and data protection law (LOPD). In this case, the regulations of data protection law must be followed. Furthermore, any data processed by the system must not be shared with any type of external library.

To solve this problem, a series of preliminary measures have been defined. The first one consists of signaling that there are cameras recording. This will be done through the use of posters at the entrance of the installation. For the second measure, it will be necessary to generate an authorization that need to be signed by the players in order to allow recording them.

Furthermore, it is possible to implement a distortion mask outside the recording region of interest. This could facilitate recording of third parties as the only area left without distortion is the field, and thus, these people can not be recognized.

Another problem that can be found is if the video is recorded by drone. If recording is done in a private area, it is possible to fly up to ten meters high in any case. However, above these ten meters, care must be taken with the flight restrictions set by AESA. Below is a map of Enaire drones restrictions.

Fig. 1.8. Enaire drones restrictions map [5]

Finally, and since the project is an original idea, it is necessary to study similar patents in such a way that none of them prevents its development. Some companies such as MediaPro or LaLiga have patented similar ideas that could interfere with the development of this project.

2. DEFINITION OF THE PROBLEM

The essence of this project is to be able to provide a tool with which players and coaches can work and analyze movements made on the training field. This idea is driven by the knowledge of the author of this project about the lack of free use automated audio-visual statistical tools in the world of sports, and more specifically in football.

With the intention of continuing investigating about statistical tools applied to sport, various interviews were held with people belonging sports world, and more specifically, football. Most of the people showed the necessity of having an accessible tool which can handle video analysis without lots of handmade job. This tool should represent the results as a tactical board. A zenith view from at least one player position in the pitch would give trainers a better understanding of the game.

Another necessity that most of these interviewed people showed was that a short processing time is require. This is due to the fact that trainers want to study tactics and show it to players during training. Also, this tool must be easy to install at any sports field, by using a camera tripod positioned with some height from the field and even to be able to record videos with a smartphone once it is calibrated.

As this system it is defined as a tool for training, two different cases will be analyzed during this project. The first example will be about tracking one person on an specific area delimited by cones and handling different camera angles. The second example will take place in a similar area but two people will be tracked.

These previous examples are base cases which can be scaled in order to achieve all team monitoring. This scaling is out of bounds from the scope of this project, as much more processing capacity is needed. Other solutions such as servers or cloud processing could be studied in the future.

2.1. System requirements

In this section it will be explained all the necessary basic requirements that the system have to fulfill. Following they are mentioned and explained:

- **Camera:** A minimum 1080p camera is required. In this way it will be possible to obtain images with a minimum of quality, and therefore, will improve the performance of the tools used. The software must be flexible enough to be able to analyze video sequences obtained from cameras of all kinds, whether they are traditional video cameras, mobiles, drones... In addition, these cameras may be subject to changes in plane and position, so an initial calibration will be required before recording any video sequence.

- **Functionality:** The system has to be able to detect a person by itself, without receiving any input. This person should then be tracked through a specified rectangular area of the field with an error of less than 5%. The program must be robust enough to handle partial occlusions. In addition, this project will analyze the same base case but for two people.

- **Processing time:** The system to be developed has to meet certain requirements regarding the time needed to process the video. After the interviews carried out, a processing time limit of 10 minutes has been set for a one minute long video in which a person is followed. A 15 minute limit was also set to analyze two people in a video of equal length.

- **Data representation:** The data obtained must be represented in a friendly way after processing. This representation must be done mostly by 2D representation of the area selected for monitoring. These graphics must include at least a map with the player's position on the field and a heat map.

2.2. System diagram at user level

The following diagram shows how the functionality of the project looks like from a user point of view.

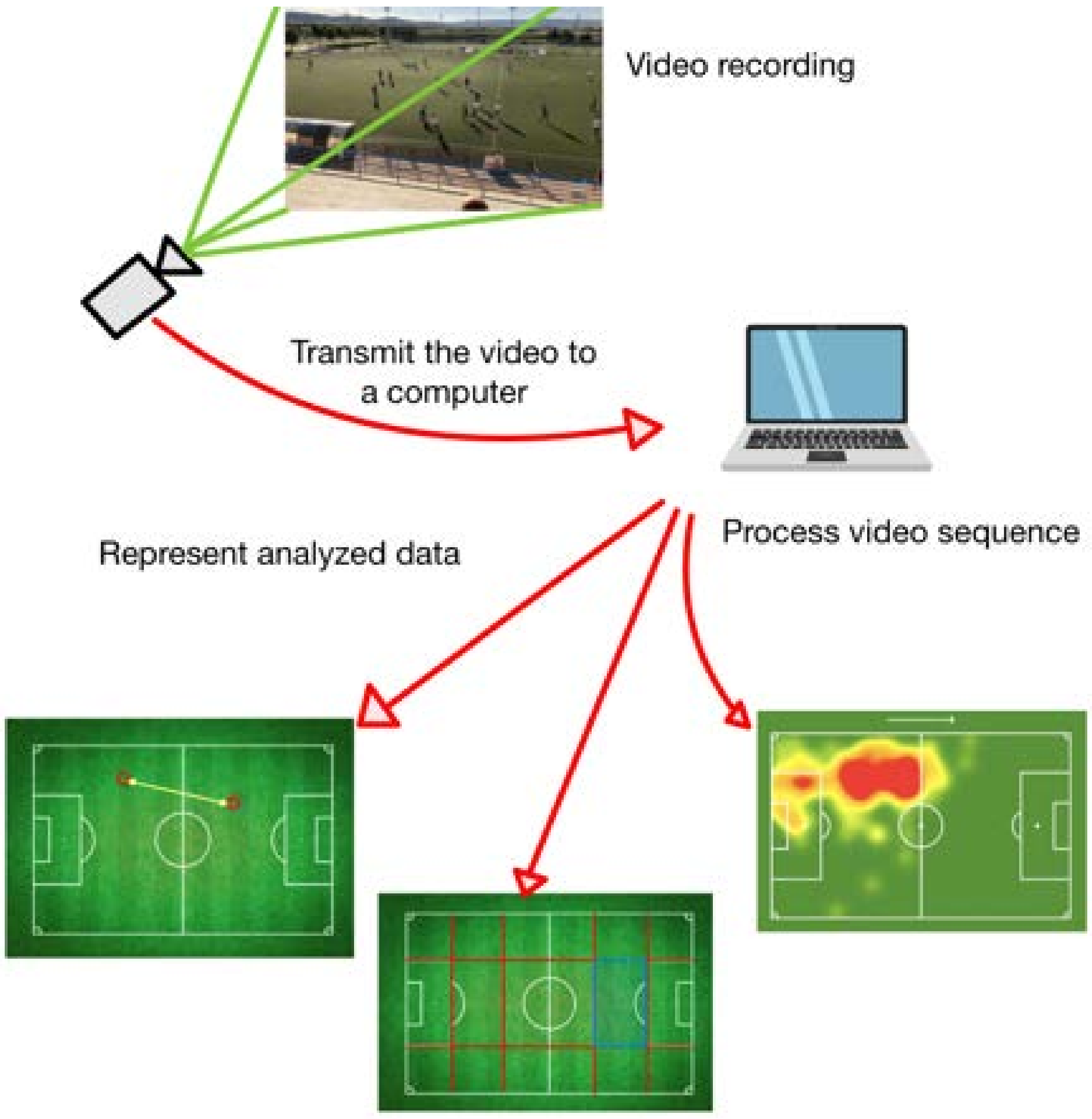


Fig. 2.1. User diagram

2.3. State of the art

In this section it will be analyzed diverse applications and methods in the field of video analysis and detection. As this theme is very wide and it is still growing, some techniques will be deprecated for this explanation and this part will focus on the basis for developing this project. In addition, it will be mentioned how statistics take place in sports and some cutting edge tools will be studied. As mentioned before, this section will focus on football, as is the most accessible nowadays.

2.3.1. Video analysis and detection

The task of detecting and following someone or something automatically from a video sequence consists on determine this object in time and space. The applications of these techniques are very wide nowadays. They go from automotive, drones, and fire detection to health-care, terrain detection and pet caring. This type of analysis has infinite applications and a big room for improvement, that is why it is continuously being developed. For this reason, concepts that will be explained during this section are general and well established.

A video detection system consists of five main blocks that can describe almost every case: recording, determining the region of interest (ROI), detection, classification, tracking and application. Figure 2.2 describes the data flow between these blocks.

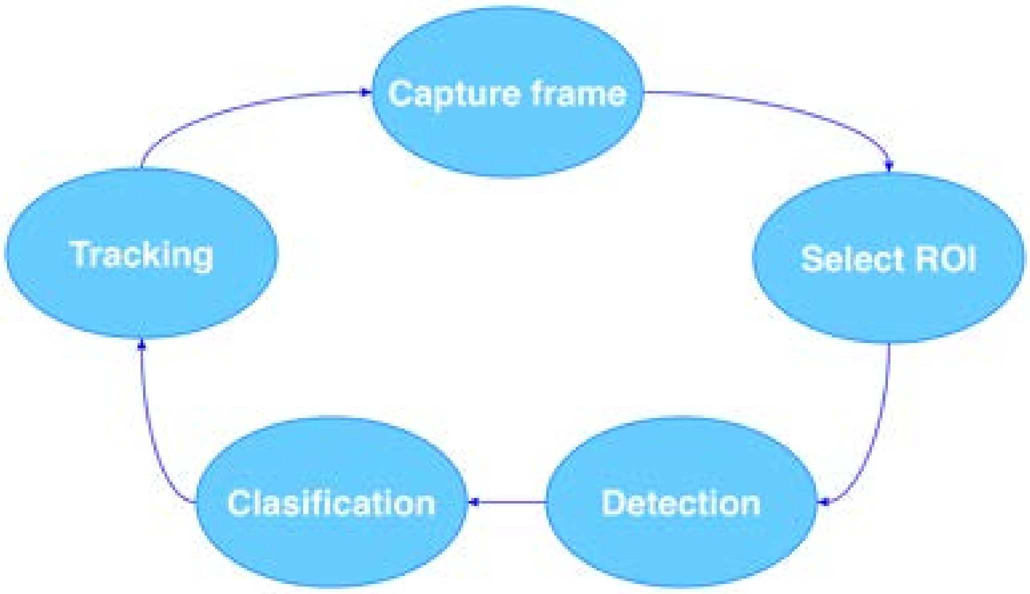


Fig. 2.2. Video detection block diagram

Capture frames

Frame capturing is the first step of a video detection system. The fact of getting usable images is crucial and challenging at the same time. The world of video recording, photography and cameras is so wide and large that it is not possible to explain it in this paper. Thust, only cameras used for this type of systems will be mentioned.

In the last few years, development of smartphones has changed our way of recording and taking photos. It is not a must to have a professional camera that achieves certain requirements. With any mobile device it is possible to take photos and videos with high enough quality to be processed by one of this video detection system. Also, there is another type of camera that differ from traditional cameras. Wide-angle cameras allow to cover more area in a same size frame by distorting the image. This make it possible to capture more information but they have a payload, without a well done calibration they are more imprecise.

Another relevant concept is the field of view (FOV). As its own name describes, the field of view is the amount of observable world that the camera can capture in one frame. This depends on the angle and distortion of the lens. Figure 2.4 shows a comparison between a photo taken with a smartphone and another one taken with a wide-angle camera.

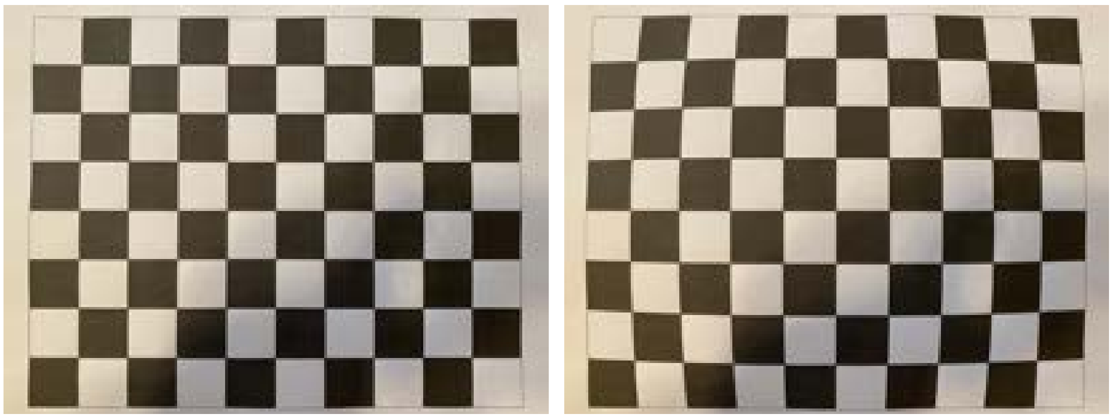


Fig. 2.3. Difference between normal and wide-angle camera

Region of interest (ROI)

The main goal of the region of interest is to define a data set that will be used for a particular purpose, in this case, for detection and classification. The use of a ROI usually helps to reduce the payload that the software has, focusing on the relevant data that is going to be analyzed. ROI is also used in other applications such as medicine and biotechnology.

In the case of video analysis, ROI is going to be used in 2-dimensions, using pixel matrix as coordinates. This ROI can be defined in two ways. The first one is manually, a region can be defined by a set of points that the user insert manually. The second one is done automatically by colour grading detection. By applying a set of colour filters, it is possible to reach a mask that covers only the colours chosen. Next frame shows an automatic ROI selecting the colour red.

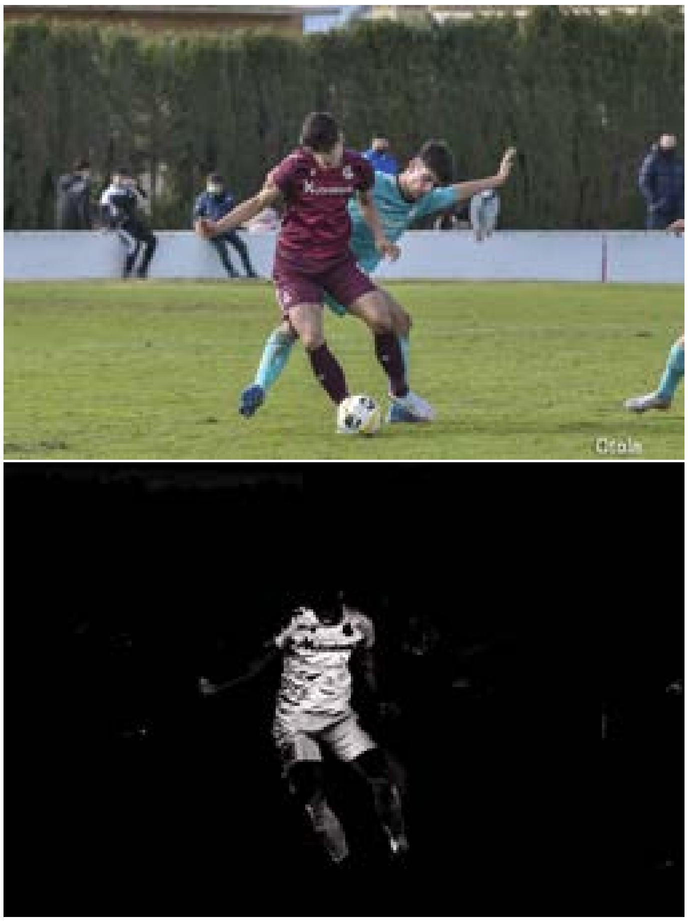


Fig. 2.4. Example of automatic ROI by colour

Detection and classification

Video analysis consists of extracting information from the objects of each video frame. This technique has been widely developed during the last years and researchers are still working on it. "Recent improvements in video analytics have been a game-changer, ranging from applications that count people at events, to automatic license plate recognition, along with other more well-known scenarios such as facial recognition or smart parking." [6].

There exist two main types of detection algorithms. The first one is the use of Support Vector Machine (SVM) algorithms, which are able to classify images because a pre-training has been done.

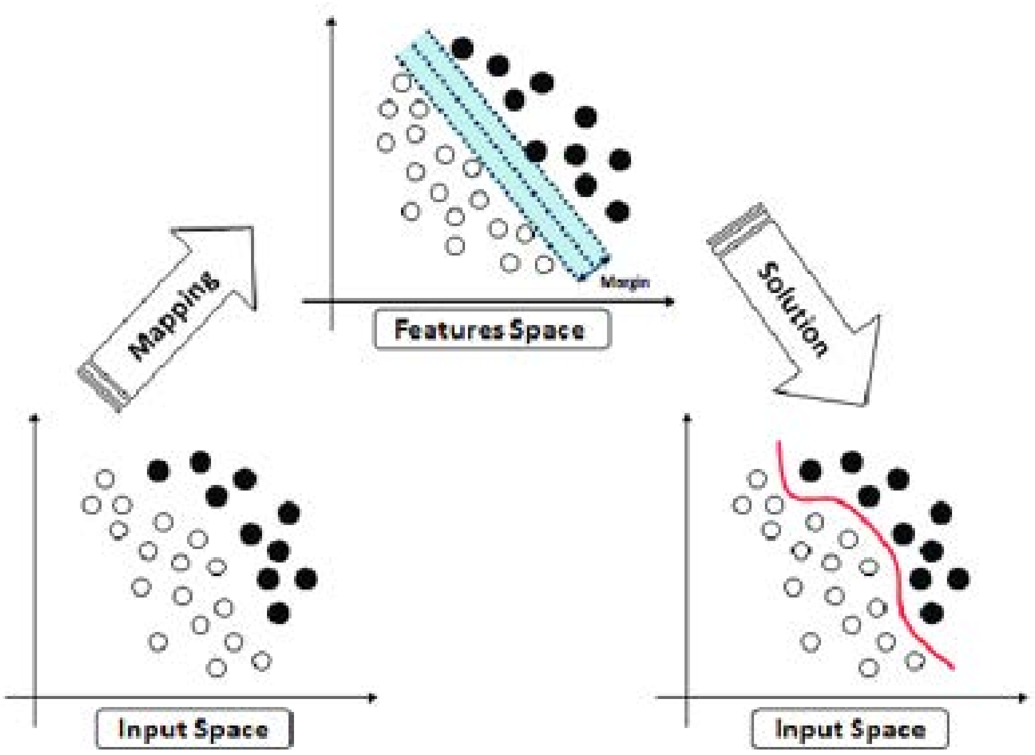


Fig. 2.5. Diagram for SVM classification [7]

The second method is Deep Learning based algorithms. These algorithms are implemented with Convolutional Neural Networks (CNN) which allows this software to learn as a human and be able to differentiate between different objects. Some of these Deep Learning detection algorithms are shown in the following image:

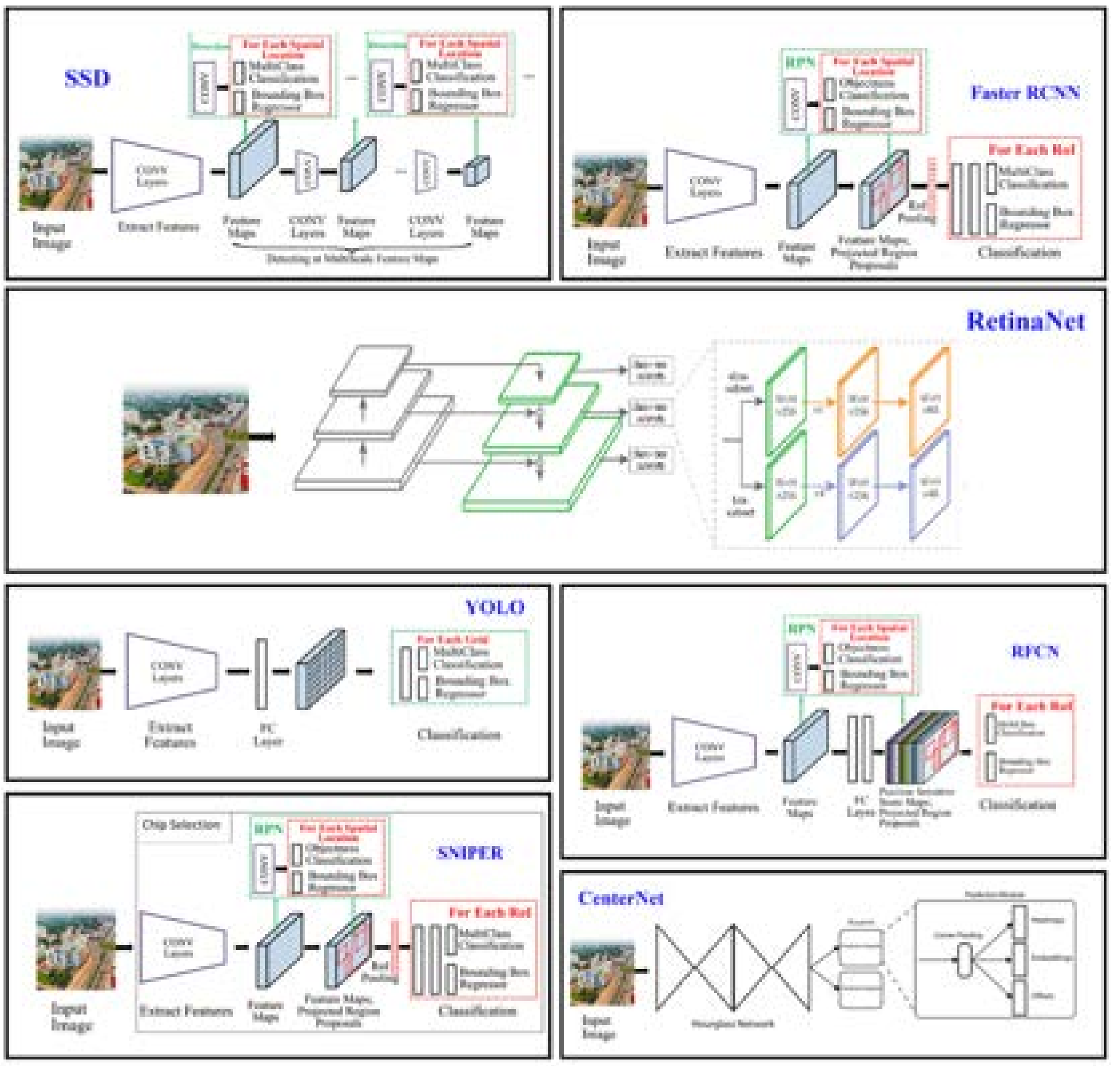
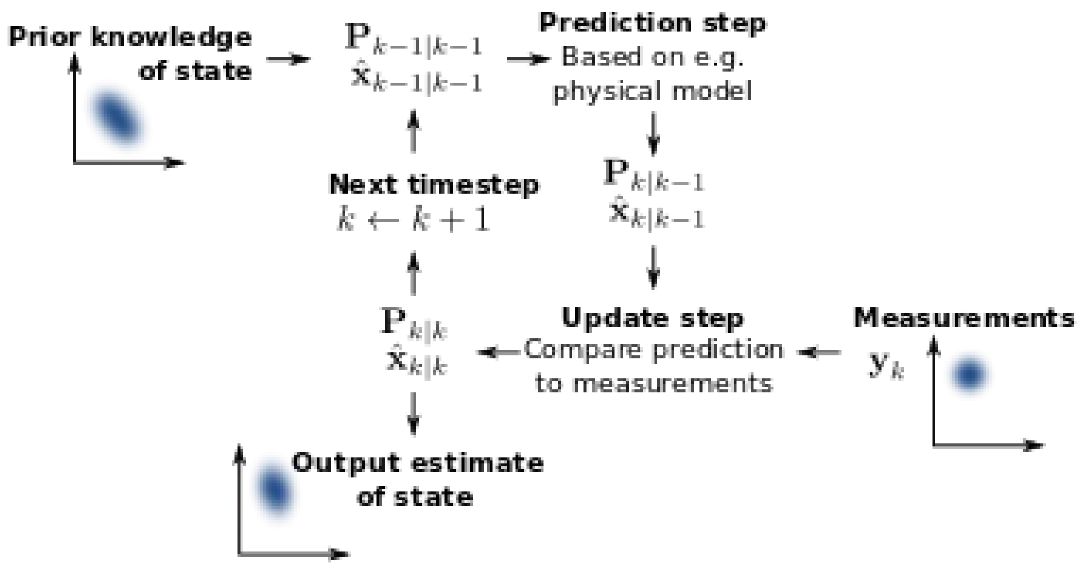


Fig. 2.6. Different structures from object detection neural networks [8]

Tracking

Tracking objects in a video consists of being able to locate during a video sequence any object that appears and moves through it. There exist several methods for tracking an object in a video. Tracking will be analyzed during this project in order to choose the one that adapts the most to system requirements. Some examples of them are MOOSE, CSRT or KCF trackers.

Most of these trackers have implemented Kalman filters."For statistics and control theory, Kalman filtering, also known as linear quadratic estimation (LQE), is an algorithm that uses a series of measurements observed over time, including statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone, by estimating a joint probability distribution over the variables for each timeframe" [9]. This type of filters generate a set of predictions from present and past locations, giving a set of possible likely next positions for the object.



Prior knowledge of state
$\mathbf{P}_{k-1|k-1}$ $\hat{\mathbf{x}}_{k-1|k-1}$
Prediction step
Based on e.g. physical model
Next timestep
$k \leftarrow k+1$
$\mathbf{P}_{k|k-1}$ $\hat{\mathbf{x}}_{k|k-1}$
$\mathbf{P}_{k|k}$ $\hat{\mathbf{x}}_{k|k}$
Update step
Compare prediction to measurements
Measurements
$\mathbf{y}_k$
Output estimate of state

Fig. 2.7. Kalman filter algorithm representation. [9]

3. DESIGN AND IMPLEMENTATION

During this chapter it will be described every functionality and particularity needed for implementing a system capable of detecting and tracking players on a defined area of a football pitch. Once the video sequence has been analyzed and the player has been tracked correctly, some statistics will be provide in order to help trainers and athletes to understand the game.

The main goal of this project is to develop a system that can provide with data to trainers and players while training, so it is possible for them to correct and improve different aspects of the game. Also, it is important to remark that it is necessary to obtain data in a short period of time, in order to be applied on-site.

As camera angles, perspective, light fixture or motion blur can be different due to the moment they where recorded, the system must be as robust as possible.

As most of the information and investigation about video analysis is done in Python, the project has been developed using this same programming language. It is important to mention that python is not as good as C++ working on real-time applications, but it is possible to obtain good results and can work with several libraries [10].

One of the libraries mentioned before is OpenCV (Open Source Computer Vision Library), an open source computer vision and machine learning software library [11]. Object detection, trackers and calibration are some of the functions that this library contains. Some of these functions related with video analysis will be used in this project. Also, the fact that OpenCV is an open source library, ease everything related with copyright and it is possible to look for official documentation and manual page.

3.1. General description

In the next figure it is shown a diagram of operation of the system. This diagram has been obtained from the block diagram of video analysis but it has been modified according to the requirements of this project.

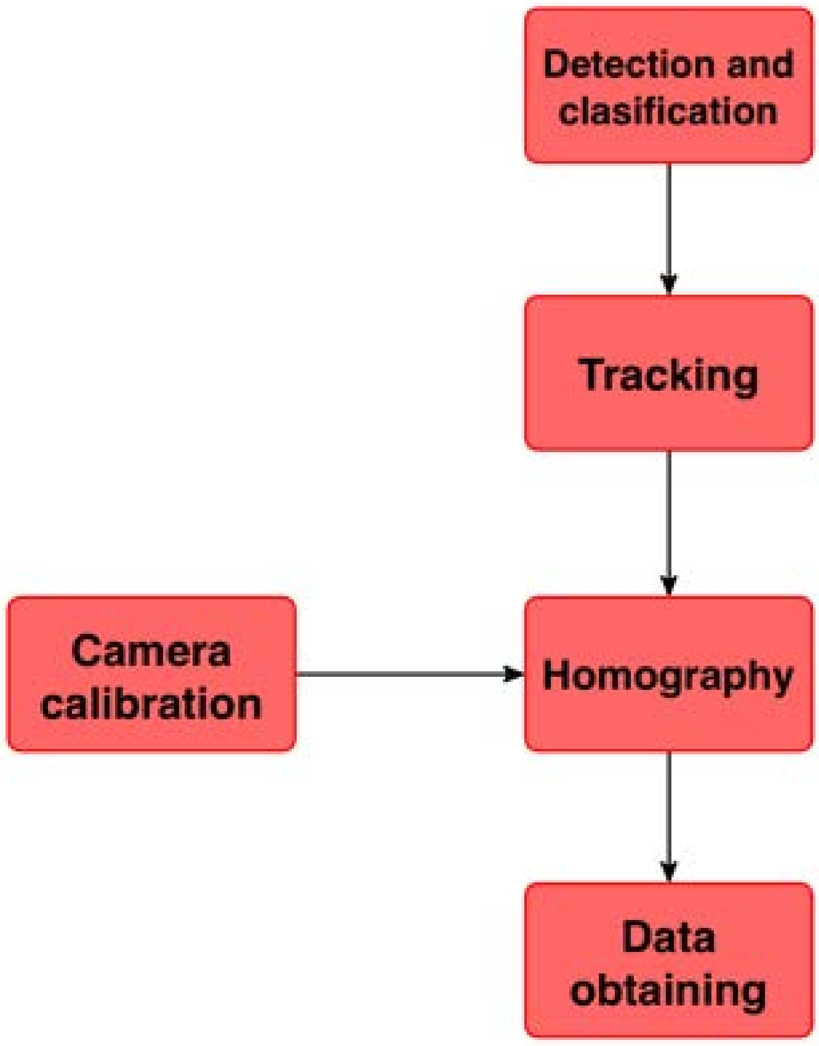


Fig. 3.1. System operation diagram

In the next sections, these blocks will be analyzed and its implementation will be explained one by one.

3.2. Detection and classification

Video analysis consists of extracting information from the objects of each video frame. Once this information is available, it is possible to use it for tracking, statistics, etc.. The process of video analysis can be divided in three main steps: detection, classification and recognition. Following these tree steps will be briefly explained.

- **Detection:** It is the first of this three steps. Object detection consists of processing an image and locating a set of object on it. In this step there is no distinction between each of this objects.

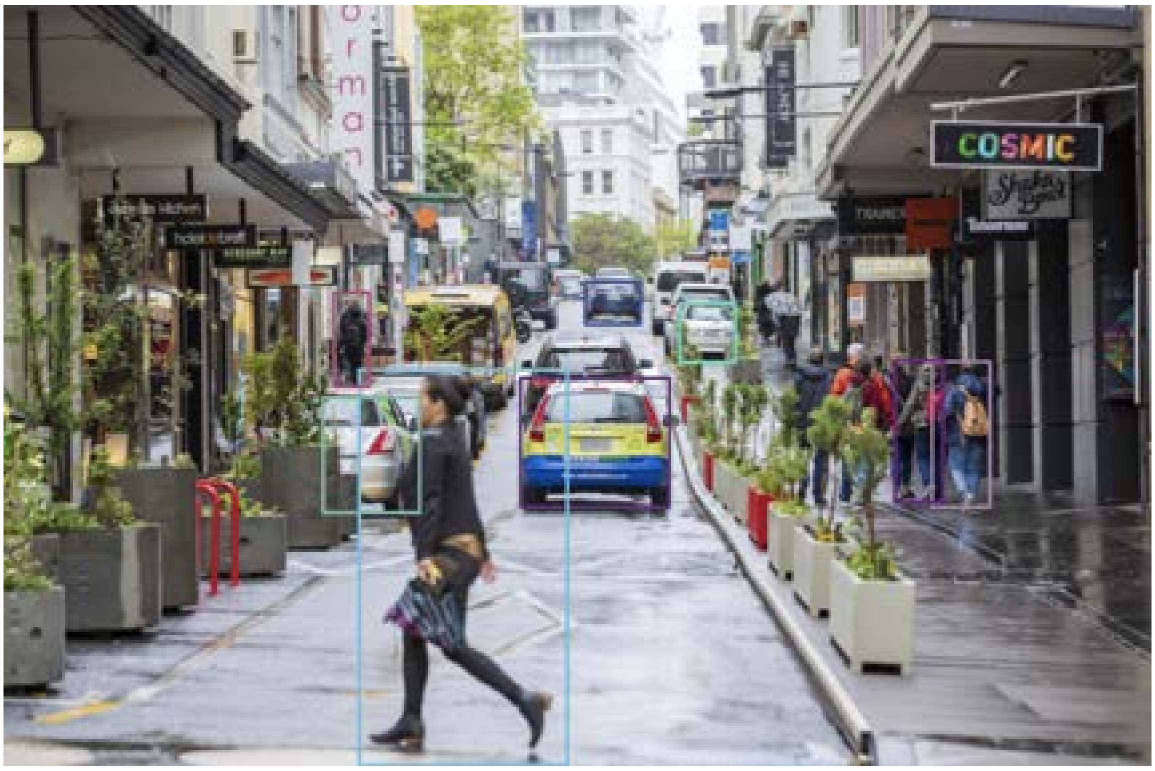


Fig. 3.2. Image after applying object detection

- **Classification:** Once the objects are located and a bounding box has been assigned to each of them, it is possible to proceed with classification.

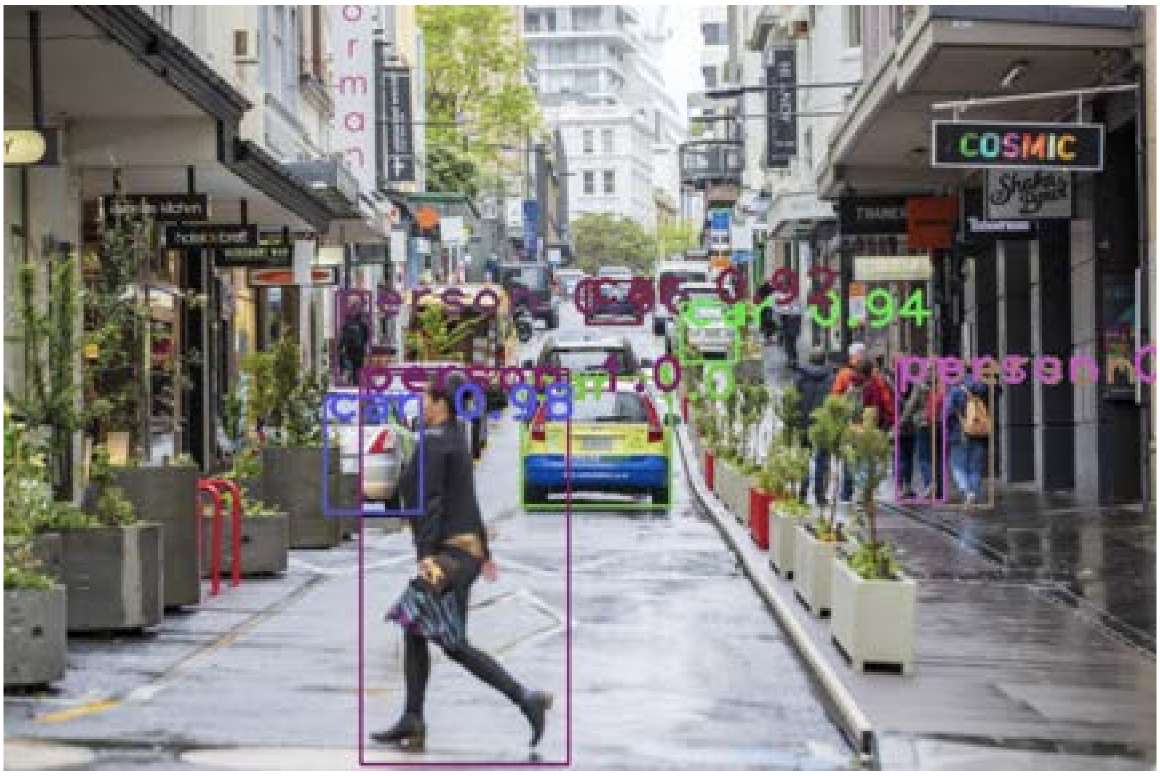


Fig. 3.3. Image after applying object classification

- **Recognition:** It is the last step of video analysis. Recognition is the capacity of the system of distinguishing objects of the same type. For example, face recognition, not only can detect and classify that there is a face but it can detect whose face it is. This step is not computed in all video analysis systems, as may be not necessary. In this particular project, recognition will not be applied by the moment, but could be applied for future development in order to enhance athlete differentiation.

There are multiple methods and techniques of video detection and classification. The main two types are SVM (Support Vector Machine) based algorithms and Deep Learning algorithms. The SVM based algorithm that will be analyzed for this project is HOG (Histogram of Oriented Gradients) and the Deep Learning based algorithm is YOLOv3 [12]. These two methods were chosen because they are supported by OpenCV library.

3.2.1. SVM+HOG

The method of HOG (Histogram of Oriented Gradients) is a video detection technique defined by Navneet Dalal and Bill Triggs in 2005 [13]. This method uses the intensity of pixels and the angle direction of the gradient so it is possible to detect edges and contours.

The steps for processing images with HOG method according to [14] are:

1. Divide the image into small cells and compute a histogram for each cell.
2. Determine each cell gradient orientation angle taking into account every pixel.
3. Group near cells into groups called blocks.
4. Normalized groups of histograms. These histograms represent the descriptor.

Next figure shows the algorithm scheme:

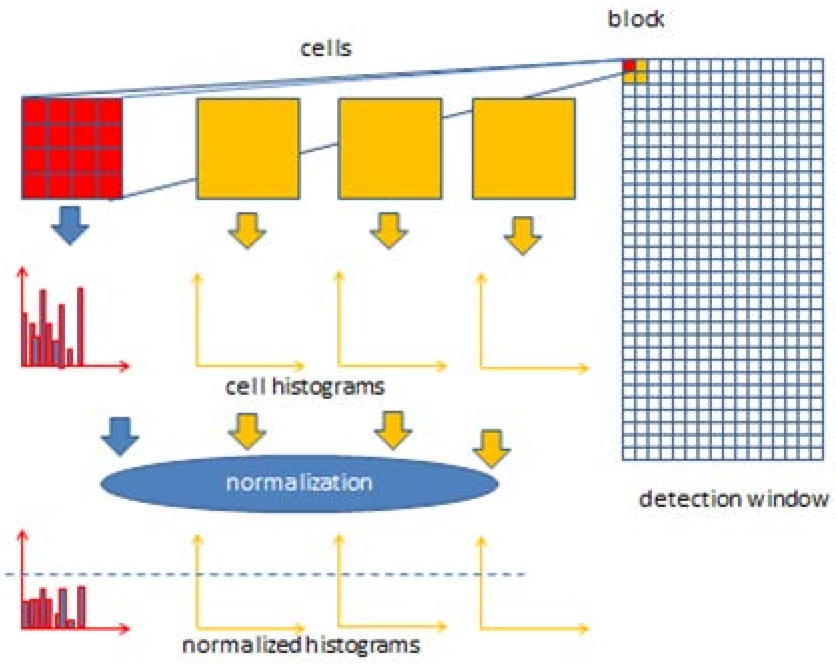


Fig. 3.4. HOG algorithm scheme [14]

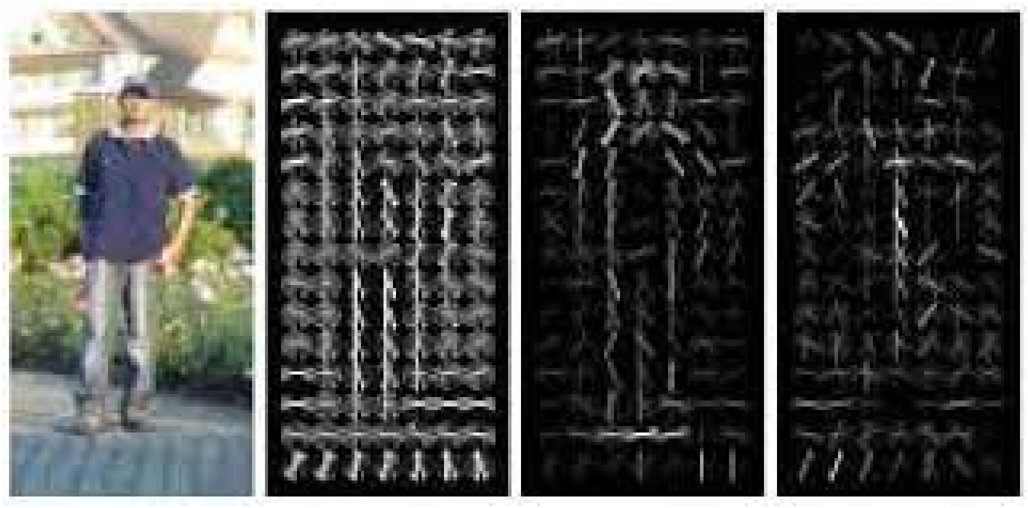


Fig. 3.5. HOG images example

Images come out from the HOG descriptor as shown in the previous figure and become an input for the SVM algorithm. This algorithm is based on Maximal Margin Classifier, based on hyperplane concept. In this particular case, the SVM algorithm is designed in order to classify people. In the next figure there is an example about how a SVM classificator looks like.

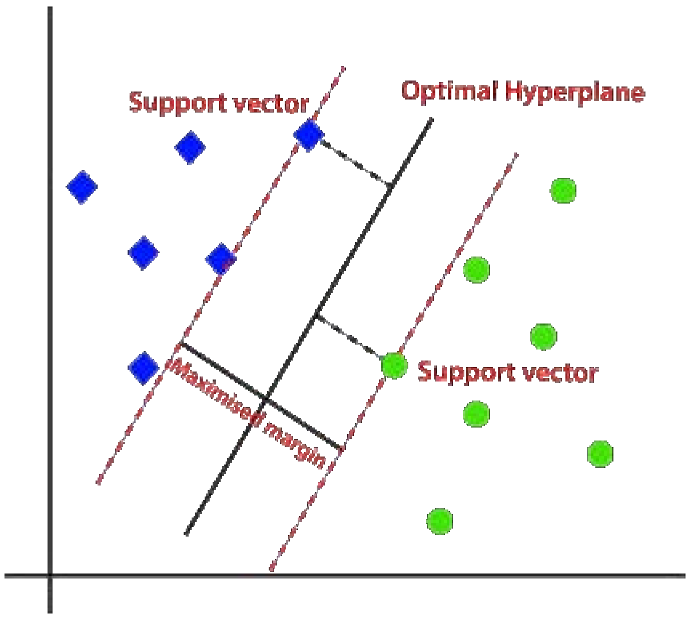


Fig. 3.6. HOG images example

This classification is implemented in Python with functions cv2.HOGDescriptor() and hog.setSVMDetector(). These two functions are already included in OpenCV libraries.

3.2.2. YOLOv3

YOLOv3 (You Only Look Once, Version 3) is a real-time object detection algorithm that identifies specific objects in videos, live feeds, or images. YOLO uses features learned by a deep convolutional neural network to detect an object. Versions 1-3 of YOLO were created by Joseph Redmon and Ali Farhadi from 2016 [15]. YOLO is named "you only look once" because its prediction uses 1×1 convolutions. This system belongs to the Deep Learning algorithm family.

YOLO is a Convolutional Neural Network (CNN) used for detecting objects in real time. CNNs can be trained with previous data and it is possible to obtain at least as good results as HOG+SVM method and x20 times faster. Architecture of this CNN is described in the next figure.

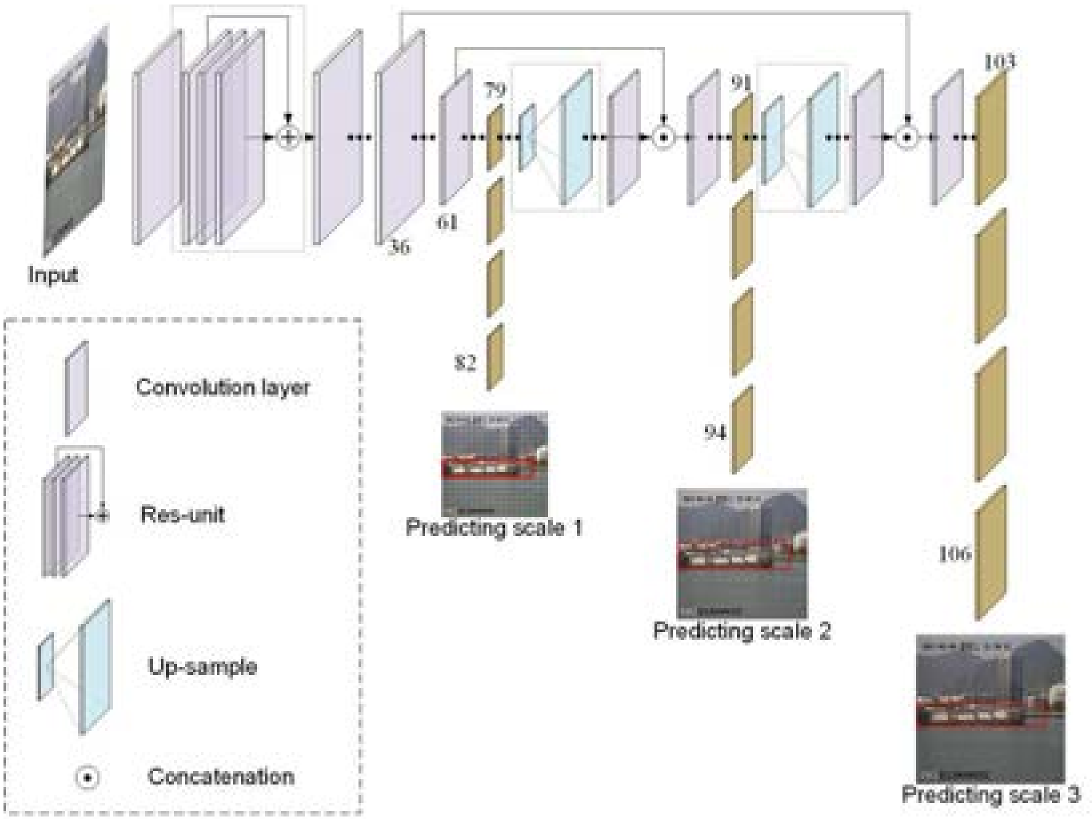
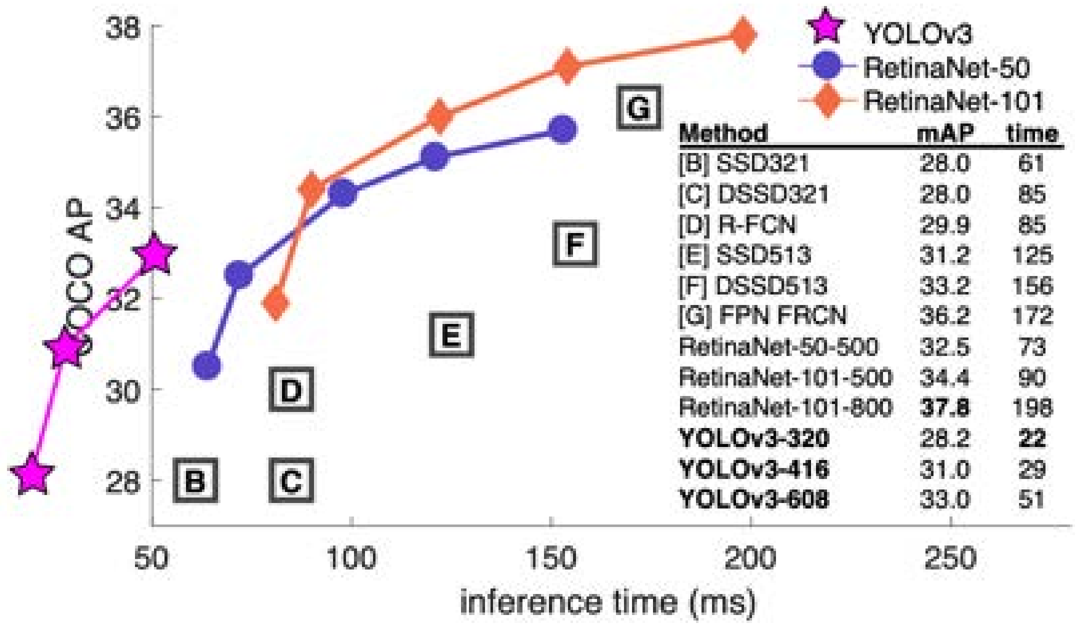


Fig. 3.7. YOLOv3 neural network [16]



Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

Fig. 3.8. YOLOv3 comparison with respect other detectors [17]

As shown in figure 3.8, YOLOv3 is faster an more accurate than previous detectors based on Deep Learning algorithms. The graphic is based on the list of objects *coco.names*, that provides a set of the most common objects to detect. This improvement caused because the redesign of the algorithm.

YOLOv3 algorithm computes an analysis of the image in three different scales in parallel. It computes detection and classification once for each scale and at the end, it merges the results, obtaining a combination of these three. There are few parameters that can be adjust in order to optimize the detection. These parameters are:

- **Length of the neural network:** Number of layers of the neural network. It is usually around 105.
- **Size of detection box:** The size of the detection box can be assigned and adapted to the situation. For this project, this variable will help because some of the people to detect might be far from the camera and thus, it size in pixels is small.
- **Image size:** It is possible to vary image size in order to optimize detection time.

YOLOv3 can be implemented in two different ways when talking about code. The first one is through Tensorflow. Tensorflow is an open source library developed by Google focused on training and using neural networks. This implementation require very demanding requirements, as it use GPU to process the code. The requirements for the GPU are two. Graphic card must be from Nvidia and a minimum 4 gigabytes graphic memory is required.

The second method is implemented by libraries from OpenCV. This method is easier to implement, but in the other side, it gets slower results. For this it is necessary to download configuration, names and weights files from Darknet GitHub repository [18].

For this project there is not available a GPU which fulfill the requirements. Because of that lack of resources and because this project is being developed with OpenCV libraries, the method which will be applied is the second.

Functions from OpenCV that are used:

- **net = cv2.dnn.readNet('yolov3.weights', 'yolov3.cfg')** This function initializes the neural network
- **blob = cv2.dnn.blobFromImage()** This function prepares the image to be readable by the neural network.
- **net.setInput(blob)** Image is inserted into the neural network.
- **outputlayersnames = net.getUnconnectedOutLayersNames()** Get the layers names.
- **layersOutput = net.forward(outputlayersnames)** Get the output result layers.

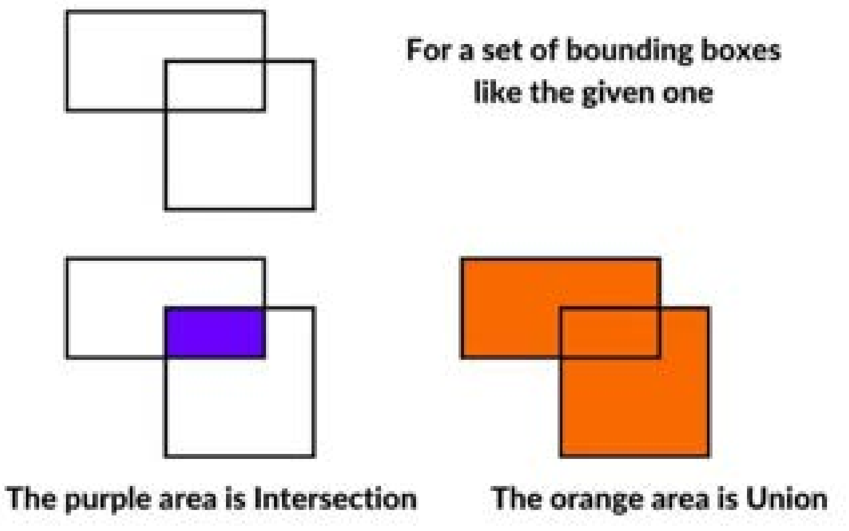
3.2.3. Non Maximum Suppression (NMS)

Non Maximum Suppression is a technique used in many video analysis tasks. The application for which will be used in this project is to get a unique bounding box from the several ones that the detection method give as an output. Following are explained the three steps of NMS algorithm [19]:

1. Select the bounding box with maximum confidence and add it to the final predictions.

2. Select any of the nearest neighbours bounding boxes and calculate IoU (Intersection over Union) ratio with the original bounding box. If IoU is greater than the threshold defined, this neighbour bounding box is added to the final prediction list. IoU can be computed as:

$$IoU = \frac{\text{Area intersection}}{\text{Area union}} \tag{3.1}$$



For a set of bounding boxes like the given one
The purple area is Intersection
The orange area is Union

Fig. 3.9. Graphic explanation of union an intersection of bounding boxes

3. If there are any neighbour bounding boxes left, repeat from step one.

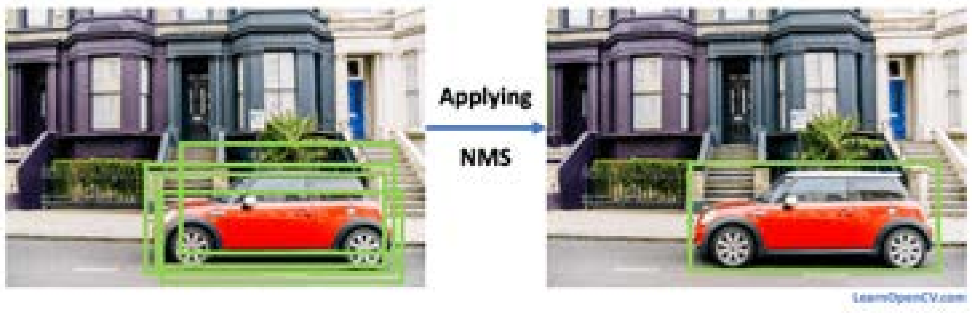


Fig. 3.10. Non Maximum Suppression example [19]

3.2.4. Implementation

Next figure describes the operation diagram for detection and classification. The detection method that will be implemented is YOLOv3 detecting people, as it is faster and obtain at least the same or better results. Even though, in the next chapter it will be justified with data.

- **Input:** Frame from a video sequence.
- **Output:** 2D coordinates in pixels of the centered base from the person detected

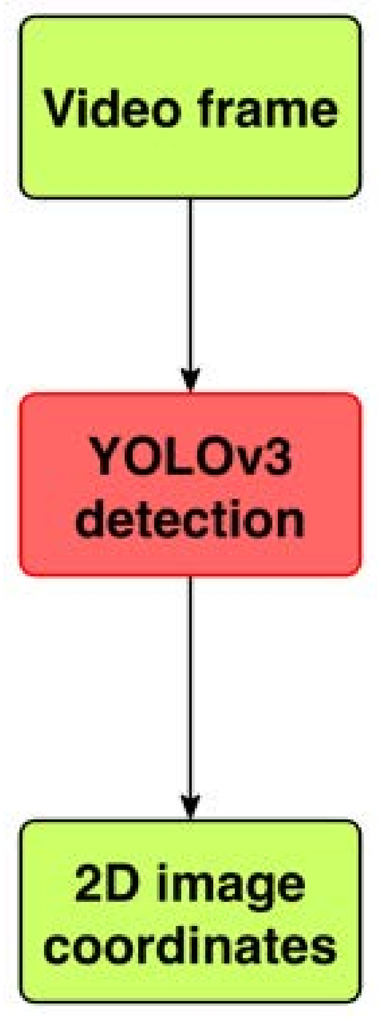


Fig. 3.11. Detection block diagram

3.3. Tracking

In this section it will be explained the importance of tracking into this system. Object tracking is one such application of computer vision where an object is detected in a video, otherwise interpreted as a set of frames, and the object's trajectory is estimated. [20].

Trackers are used rather than detectors as they are faster. This happens because when a tracker is started, it has much more information than a detector. For detectors, it is required to analyzed the full frame and find an objects. For trackers, this area is previously defined and so, it can use pixels colours as starting point information.

There exist two different approaches for trackers. One is Single Object Tracking (SOT), where only a single object is tracked during a sequence of video frames. The second one is called Multiple Object Tracking (MOT), used for tracking more than one object during a video sequence [20].

As every system it has some variables that are important for the performance and can be adjusted depending on the requirements of the applications. Some of these variables are:

- **Movement speed:** The speed at which an object moves can influence in object tracking as trackers compare next frame with the previous frame in order to keep track of a colour sequence.
- **Accuracy:** It might vary depending on the system. It is possible that for an specific application high accuracy of the margin box is necessary. In this case, the tracker will need more time to process.
- **Frames per second processed:** The frequency of frames analyzed by second that the tracker can stand will determine if it is suitable for real-time tracking or not.

Trackers implemented in OpenCV have already implemented Kalman filters and they are trained with several data sets in order to distinguish between object and background. Following, some of these OpenCV implemented trackers will be mentioned and its main characteristics will be explained.

TABLE 3.1. ADVANTAGES AND DISADVANTAGES OF DIFFERENT TRACKERS

Tracker	Advantages	Disadvantages
Boosting	• It is basic.	• Not precise. • Not reliable • It is old.
MIL	• Works well if there exists some occlusion.	• Can not recover from full occlusions. • Can not detect failure.
KCF	• Good accuracy and speed. • Detects failure.	• Can not recover from occlusions.
TLD	• Works the best under occlusions. • Works well with scale changes	• Has a lot of false positives.

Source: [21]

TABLE 3.2. ADVANTAGES AND DISADVANTAGES OF DIFFERENT TRACKERS

Medianflow	• Excelent failure reporting. • Works very well when movement is predictable.	• Fails under large motion.
GoTurn	• Robust for different angles and light changes.	• Do not work well with occlusions.
MOOSE	• Faster than any other tracker. It allow a high FPS throughput.	• Not very accurate.
CSRT	• High accuracy tracker.	• Not as fast as MOOSE

Source: [21]

The most interesting ones are CSRT, KFC and MOOSE trackers. These three will be tested in the next chapter and one of them will be selected.

3.3.1. Implementation

In this subsection it will be shown an example of a tracked video sequence and an operation diagram of this module.

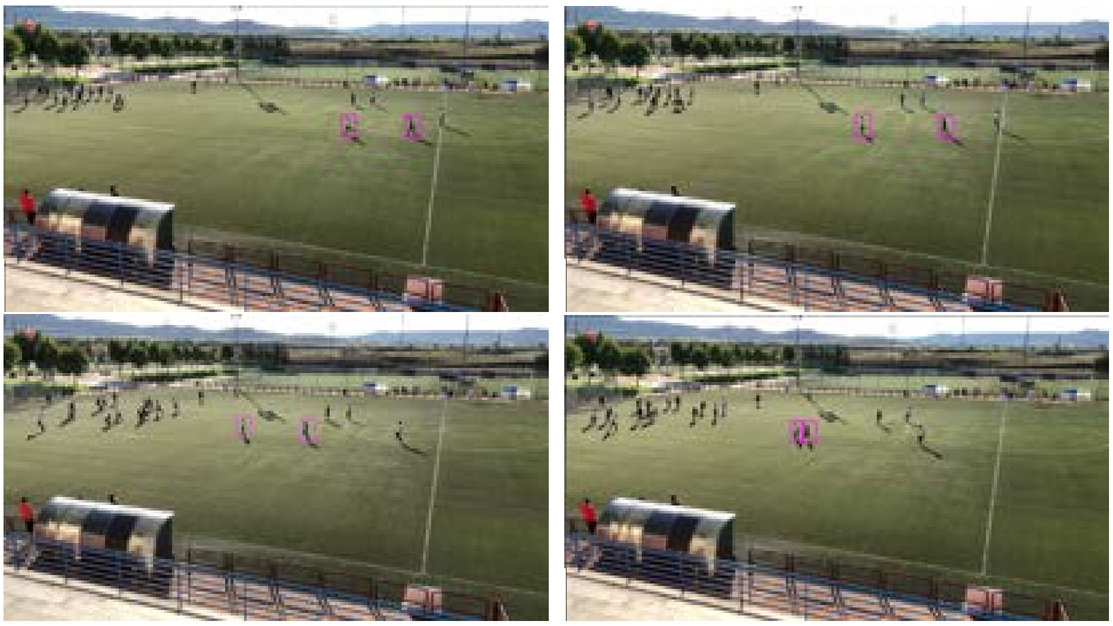


Fig. 3.12. Tracked video sequence example

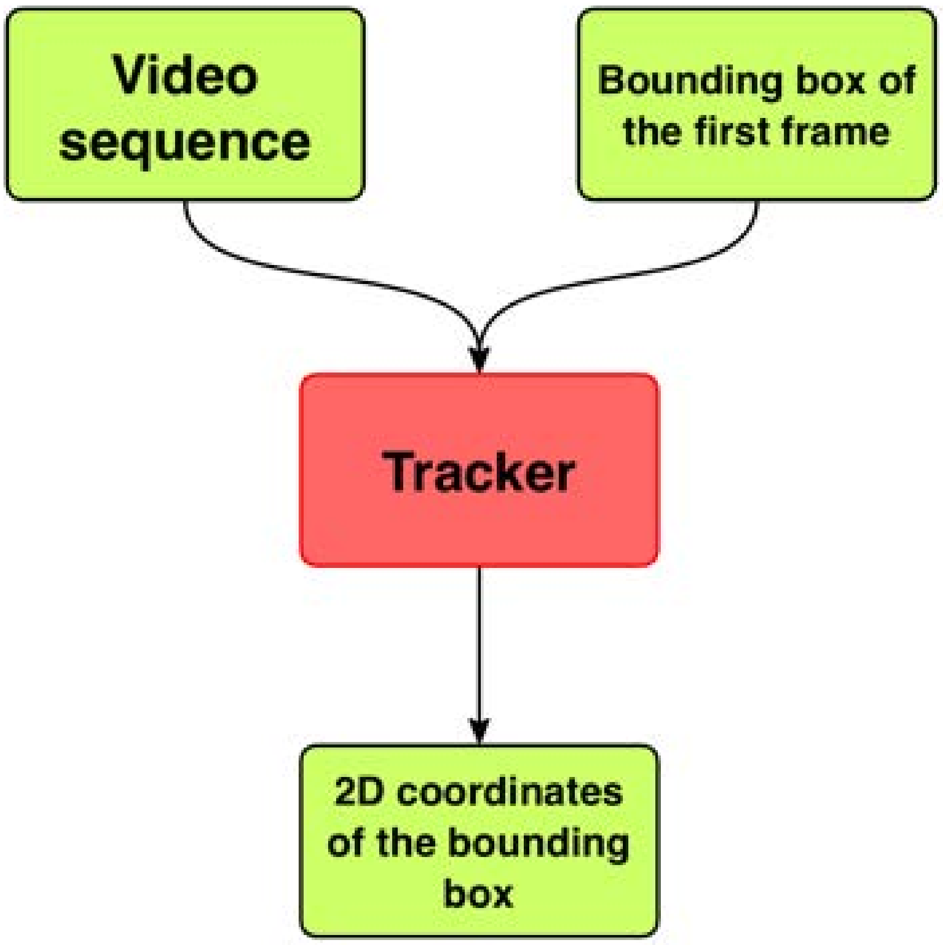

Video sequence
Bounding box of the first frame
Tracker
2D coordinates of the bounding box

Fig. 3.13. Tracked video sequence example

3.4. Camera calibration and homography

3.4.1. Camera parameters

Any camera system has a set of common parameters that describe its position. For the application of this project will be necessary to obtain some of these parameters in order to compute the homography. Next figure shows the parameters of a camera model and its relation between the three different coordinate systems.

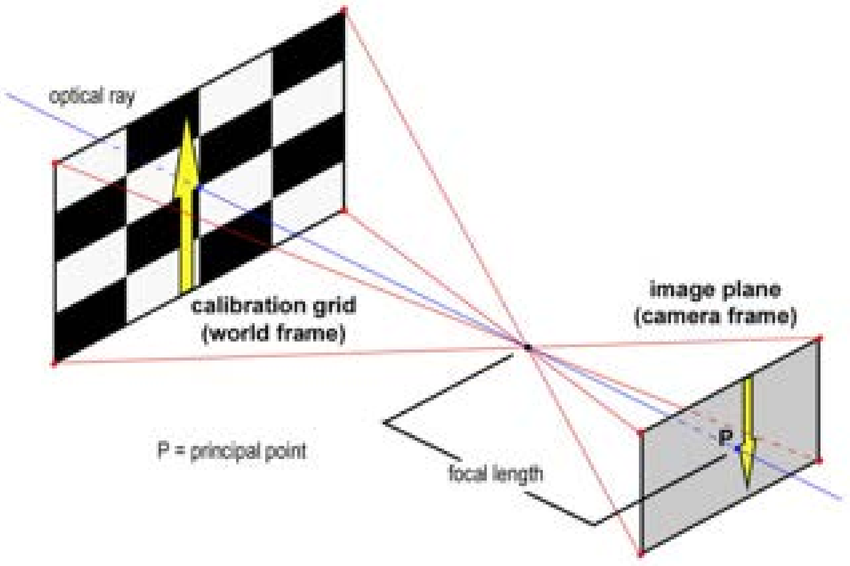


Fig. 3.14. Description of a camera system

When talking about camera parameters, it is possible to divide it in two clear groups [22]:

- **Intrinsic parameters:** Intrinsic parameters are defined by the own characteristic of the camera. They are represented as a 3x3 matrix **K**:

$$\mathbf{K} = \begin{bmatrix} f_x & \alpha & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

  where:

  - C_x**:**. it is the camera center of the x-axis measured in pixels.
  - C_y**:** it is the camera center of the y-axis measured in pixels.

- f_x**:** it is the camera focal length of the x-axis.
- f_y**:** it is the camera focal length of the y-axis.
- α**:** it is the skew angle if x-axis and y-axis of image coordinates are not perpendicular. During this project, this value is going to be always equal to zero, as images are not distorted.

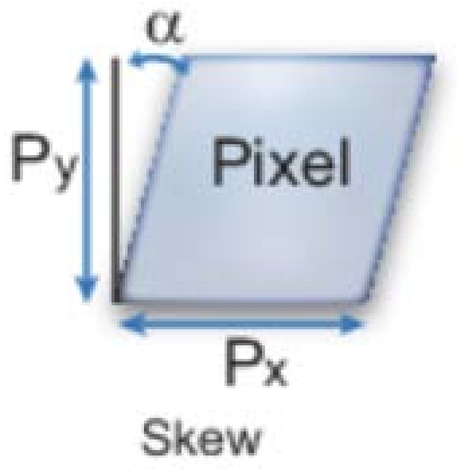


Fig. 3.15. Skew

- **Extrinsic parameters:** they depend on the position and orientation of the camera. These parameters are normally represented as **[R|t]** where:
  - *R* is a 3x3 rotation matrix of the camera on an specific position. It is obtained after calibrating the camera once it is positioned in the field.

$$\mathbf{R} = \begin{bmatrix} r_{01} & r_{02} & r_{03} \\ r_{11} & r_{12} & r_{13} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{3.3}$$

  - *t* is a 3x1 translation vector of the camera.

$$\mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \tag{3.4}$$

Knowing these two parameters we can define the extrinsic parameters matrix as:

$$\mathbf{[R|t]} = \begin{bmatrix} r_{01} & r_{02} & r_{03} & t_1 \\ r_{11} & r_{12} & r_{13} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \tag{3.5}$$

Real life coordinates will be defined as in the next figure.

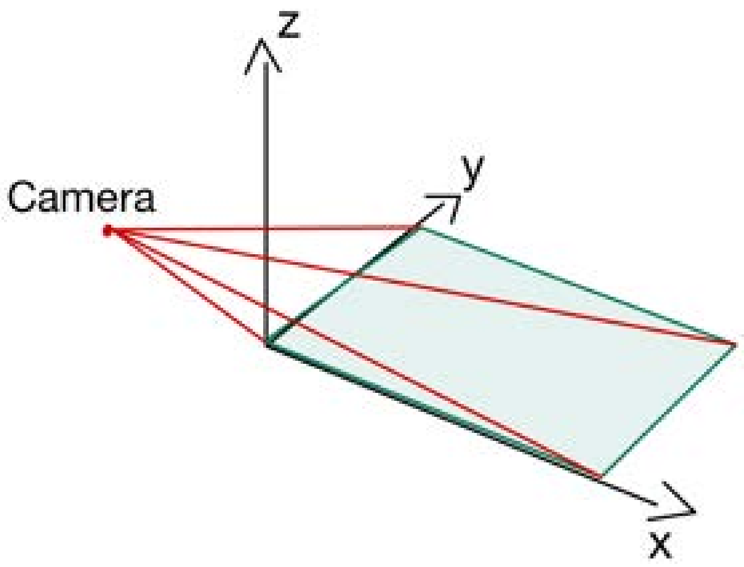


Fig. 3.16. World coordinates axis

3.4.2. Calibration

The definition of camera calibration is the process of determining specific camera parameters in order to complete operations with specified performance measurements [23]. The main goal of camera calibration consists of given a set of known points in the real world coordinates and its coordinates in the image plane, it is possible to compute the rotation matrix. Once the rotation matrix is computed, it is possible to project real world coordinates onto image coordinates.

There are several ways camera calibration can be done. Following are mentioned the most commonly used methods [24]:

- **Calibration pattern:** this method is the most commonly used. This method is applied when it is possible to have complete control about the calibration pattern is going to be used. Normally, the most popular calibration pattern is the chessboard, that will be mentioned later in this section.
- **Geometric rules:** calibration made with this method depends on the different lines and perspective of the image used. For this method it is possible to obtain dif-

ferent lines and points by subtracting the background and finding vanishing points that allow the software to compute the position of the camera. This method is analyzed in a deeper way in [25] and [26], and as they mention, it has high processing requirements.

- **Deep Learning algorithms:** this calibration method is the one that is evolving the most in the last few years. Neuronal networks are being trained in order to detect shapes and objects, without the necessity of capturing it totally in the frame. This kind of methods need lots of pre-processing images.

For this project, the option selected is the use of a calibration pattern, as is the one that requires less computational costs. Also, it is possible to calibrate the camera the first time it is used and save that calibration for future uses.

Inside this type of calibration, there are several intrinsic parameters methods that have been studied during the years. Some examples are:

- **Faugueras-Toscani:** Faugueras and Toscani proposed a calibration method based on an estimation of the Perspective Projection Matrix (PPM) using an image of a non-planar pattern. It uses both linear and nonlinear approaches. For this method it is necessary to set an input of six known 3D points [27].
- **Tsai:** Tsai calibration method consists of obtaining intrinsic camera parameters with a single photo of known points that belong to different planes.
- **Zhang:** Zhang described a method that can obtain the intrinsic parameters of the camera by at least two different view of a known pattern. This method increases its accuracy when more than twenty images are used [27].

In this problem the method that will be used is Zhang's method. The reason is because it is possible to achieve high performance results. Also, OpenCV has a camera calibration module that contains a function for calibrating a camera with Zhang's method.

3.4.3. Homography

Homography is defined as a projective transformation from a plane of aligned points to another that keeps this set of points aligned. This operation must be invertible. For applying homography it is necessary to have at least a set of four points.

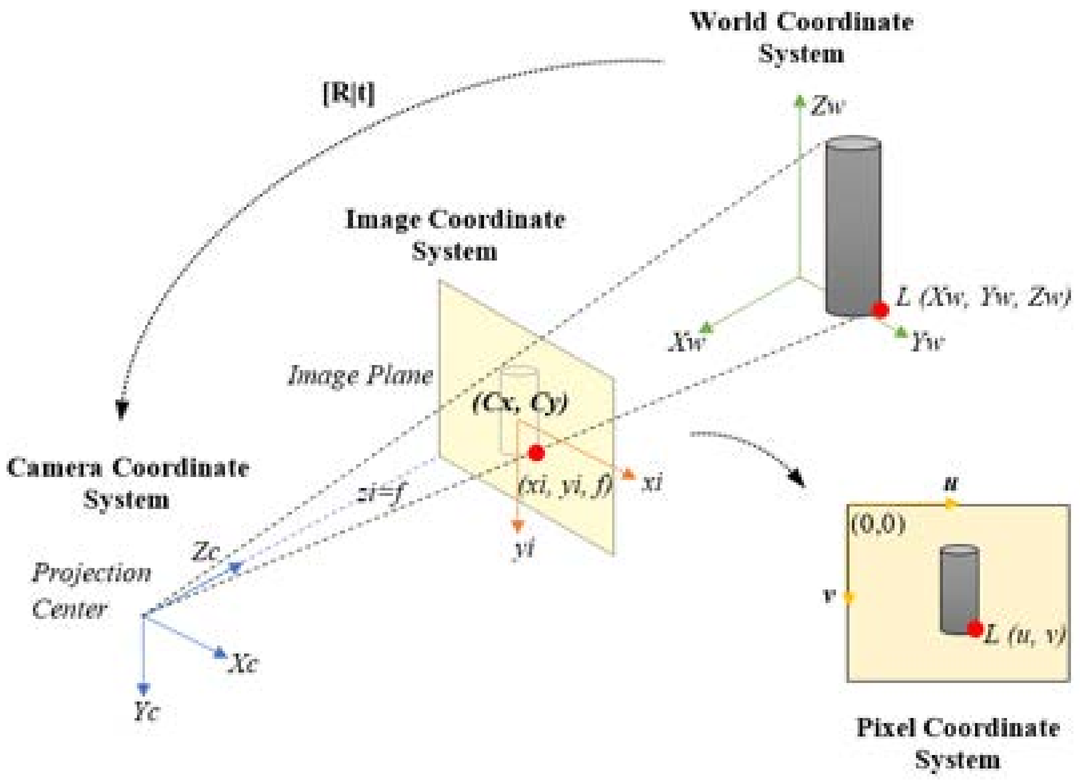


Fig. 3.17. Pinhole model camera system [28]

Firstly, it is important to determine how the vectors of coordinates look like. In the case of world and camera coordinates system are 3x1 vectors, while image coordinates is a 2x1 vector:

- $$P_{world} = \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} \tag{3.6}$$

- $$P_{camera} = \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{3.7}$$

- $$P_{image} = \begin{bmatrix} u \\ v \end{bmatrix} \tag{3.8}$$

In this particular case, it is needed to obtain from a real life 3 dimension coordinate system a 2 dimensions vector that refers to the pixels of the image. To do this and as it shows figure 3.17, it is necessary to compute two transformations.

The first one is from world coordinates system to camera coordinates system. This is a 3 dimensions to 3 dimensions transformation. In order to compute the camera coordinates it is necessary to multiply real world coordinates by the extrinsic parameter matrix.

$$P_{camera} = \mathbf{[R|t]}P_{world} \tag{3.9}$$

Where R is the rotation matrix and t the translation vector.

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} r_{01} & r_{02} & r_{03} & t_1 \\ r_{11} & r_{12} & r_{13} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{3.10}$$

The fourth parameter of the world coordinate point is a one. It is added to make possible the multiplication. This factor is called homogeneous coordinate.

The second one is from camera coordinates system to image coordinates system. This is a 3 dimensions to 2 dimensions transformation. In order to compute the image coordinates it is necessary to multiply camera coordinates by the intrinsic parameter matrix.

$$P_{image} = \mathbf{K}P_{camera} \tag{3.11}$$

Where **K** is the intrinsic parameters matrix.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{3.12}$$

In this case it is also necessary to add the homogeneous coordinate to be able to compute the multiplication of matrices.

If these two transformations are equaled, it is possible to obtain a full transformation from real world coordinates to image coordinates.

The left side is multiplied by s , the scaling factor, in order to fix the distance to the object in the real world coordinates.

$$sP_{image} = \mathbf{K[R|t]}P_{world} \tag{3.13}$$

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} r_{01} & r_{02} & r_{03} & t_1 \\ r_{11} & r_{12} & r_{13} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{3.14}$$

3.4.4. Implementation

In this subsection is going to be described how all these previous concepts and parameters are merged and applied to reach the goal of this project. In order to reach that goal, the requirement needed from this block is to obtain the coordinates of the real world of a point from a specific pair of coordinates of the image. Figure 3.18 shows an operational diagram of how this main block works at coding level. Also, sub-blocks will be briefly explained individually.

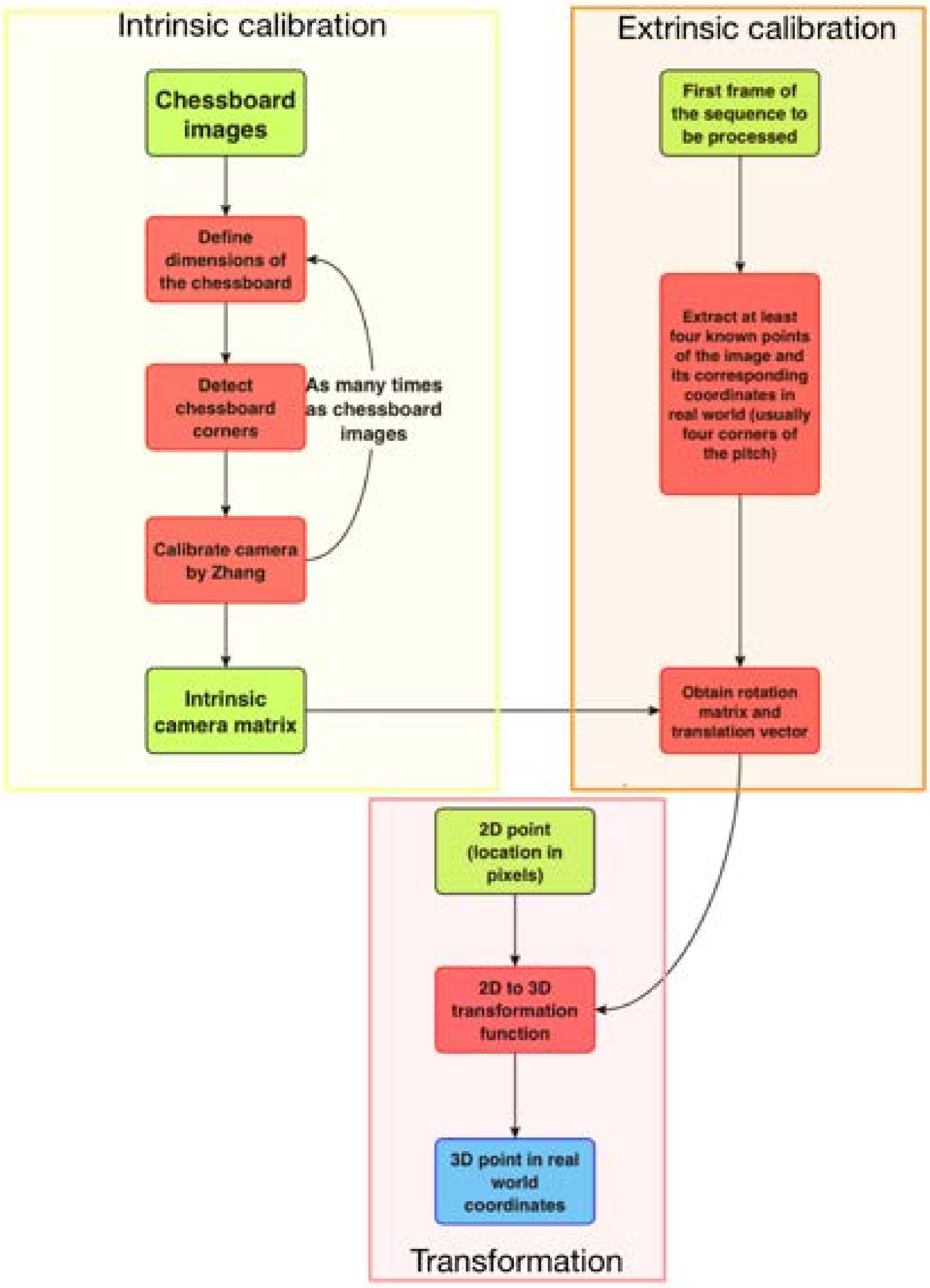

Intrinsic calibration
Chessboard images
Define dimensions of the chessboard
Detect chessboard corners
As many times as chessboard images
Calibrate camera by Zhang
Intrinsic camera matrix
Extrinsic calibration
First frame of the sequence to be processed
Extract at least four known points of the image and its corresponding coordinates in real world (usually four corners of the pitch)
Obtain rotation matrix and translation vector
2D point (location in pixels)
2D to 3D transformation function
3D point in real world coordinates
Transformation

Fig. 3.18. Calibration and homography block operation diagram

Intrinsic calibration

The intrinsic calibration process has as an input few chessboard images. And it has as output the intrinsic parameters of the camera given as the intrinsic camera matrix. Following, each block will be explained:

1. **Input chessboard images:** Firstly, few chessboard pattern images are taken from different angles and input in the program. Figure 3.19 is an example of the images to use.

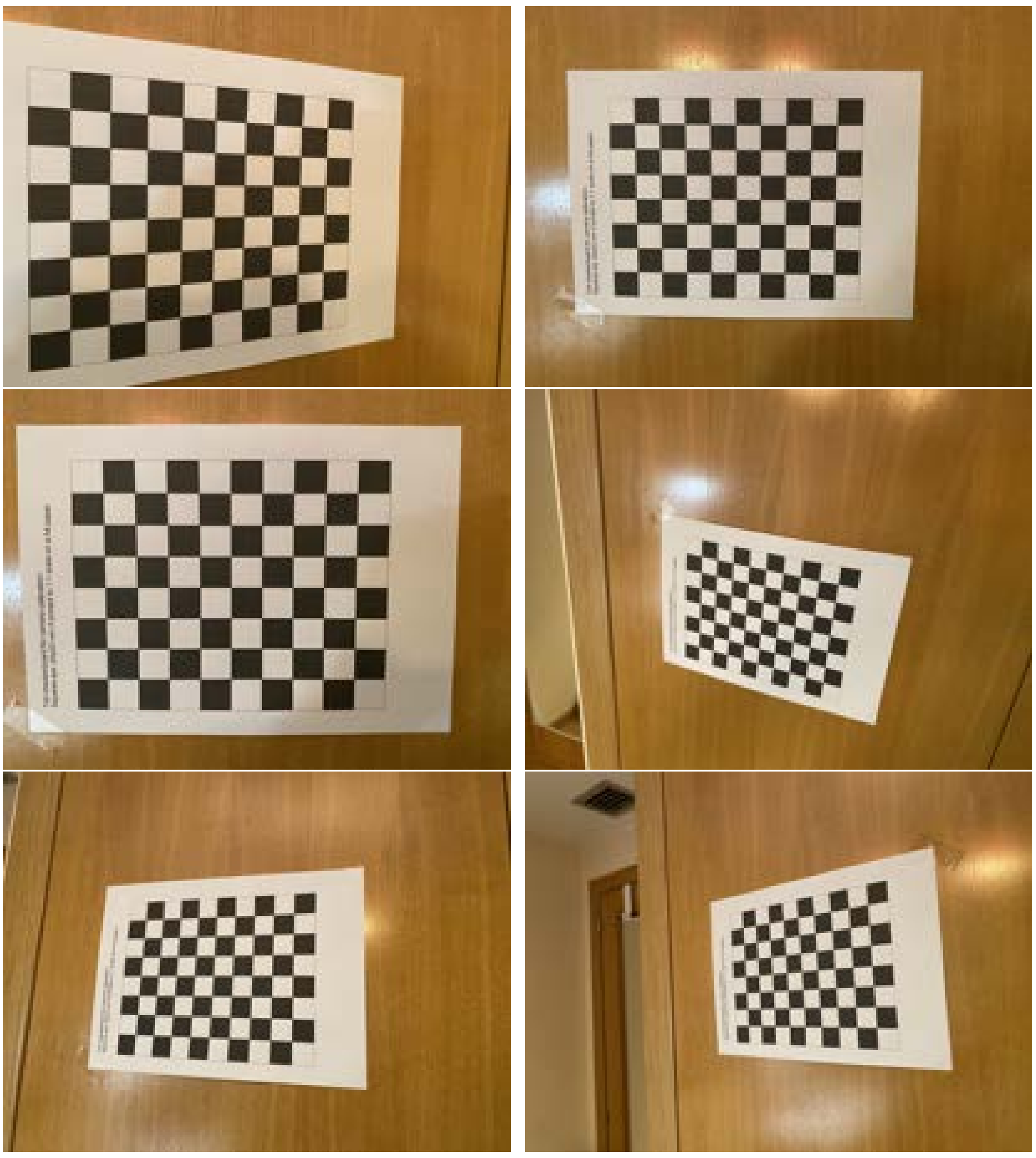


Fig. 3.19. Set of images from the Chessboard Pattern used as input

2. **Definition of chessboard size:** After importing the images to our program, it must be defined the size of the chessboard and the dimensions of each square. With this measurements, we are defining real world coordinates of the chessboard.

3. **Detection of chessboard corners:** Function cv2.findChessboardCorners belonging to OpenCV library detect the corners of the chessboard by using colour thresholds.

4. **Apply Zhang's method for intrinsic parameters:** Once all corners from all images has been obtained, they are all introduce into function cv2.calibrateCamera. This function is also from OpenCV library and applies Zhang's calibration method for obtaining the intrinsic camera matrix and the distortion vector. This function has as an input a set of object points in real life and a set of image points. As an output, it returns the camera matrix and the distortion vector, which will be used as input for obtaining the rotation matrix and translation vector. The outputs for the images inserted in step one are:

$$\mathbf{K} = \begin{bmatrix} 1088 & 0 & 624 \\ 0 & 1086 & 367 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.15}$$

$$\mathbf{dist} = \begin{bmatrix} 0.262 \\ -0.823 \\ 6.02e-03 \\ -5.58-04 \\ -0.685 \end{bmatrix} \tag{3.16}$$

Extrinsic calibration

Extrinsic calibration depends on the angle and the position of the camera with respect to the object we want to locate. For this project, camera is defined as static, in order to avoid computing continuously the rotation matrix.

1. **First frame as input:** The first frame of the video sequence to analyze is selected and used for calibrating extrinsic parameters of the camera. It is important that the field of view (FOV) of the camera can not vary at any frame of the sequence of

this video. If this happens, the rotation matrix and the translation vector will not be correct to execute the translation.

2. **Identify real world points in the image:** After importing the first frame into the program, it is possible to select onto the image the pixel coordinates of the known points and after that enter the coordinates in real life through console. It is assumed that the environment where this project takes place is planar, so coordinate Z_w will always be zero as an input for this step.

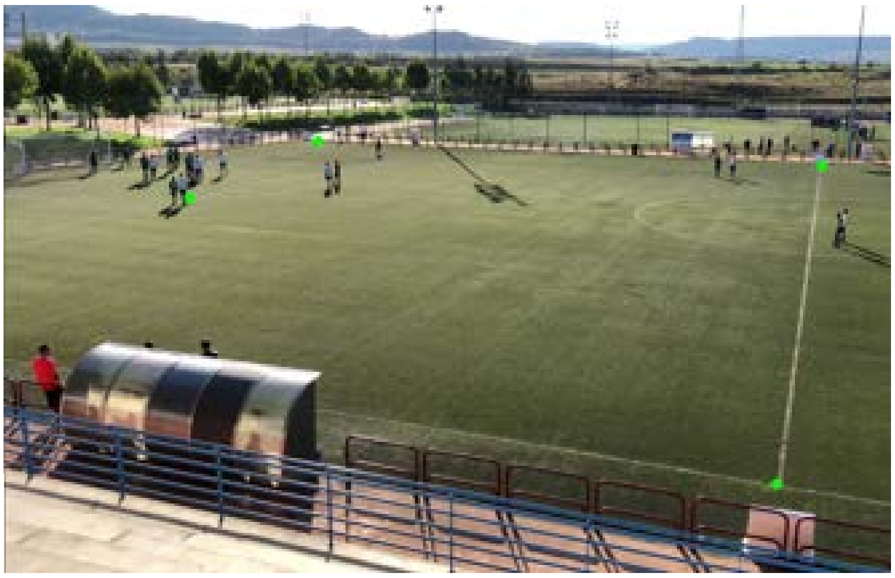


Fig. 3.20. Point selection system onto the first frame

3. **Rotation matrix and translation vector computation:** To compute the rotation matrix and the translation vector, the function cv2.solvePNP is executed. This function has the next inputs and outputs:

   - Inputs:
     - **3D points:** Points in real world coordinates obtained from the second step.
     - **2D points:** Points in image coordinates obtained from the second step.
     - **Camera matrix:** Intrinsic parameters camera matrix obtained from intrinsic calibration.
     - **Distortion vector:** Distortion vector obtained from intrinsic calibration.
   - Ouputs:

- **Rotation vector:** Rotation vector of the system that will be converted to a matrix via Rodriges method. The function cv2.Rodriges is also included in OpenCV library.
- **Translation vector:** Translation vector of the system. It defines where the camera is positioned.
- **Return:** Variable that indicates if the function has been executed correctly.

Transformation

In this step a 2 dimensions to 3 dimensions transformation is done. To explain how this function operates, homography equations will be used. To compute the 3D points, it is necessary to compute first the scale factor. Formula 3.12 is taken as start for this problem, as the goal is to pass from image coordinates to real world coordinates [29].

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} r_{01} & r_{02} & r_{03} & t_1 \\ r_{11} & r_{12} & r_{13} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix}\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{3.17}$$

$$sP_{image} = \mathbf{K[R|t]}P_{world} \tag{3.18}$$

$$sP_{image} = \mathbf{K}(\mathbf{R}P_{world} + \mathbf{t}) \tag{3.19}$$

$$sP_{image}\mathbf{K}^{-1} = \mathbf{R}P_{world} + \mathbf{t} \tag{3.20}$$

$$sP_{image}\mathbf{K}^{-1} - \mathbf{t} = \mathbf{R}P_{world} \tag{3.21}$$

$$(sP_{image}\mathbf{K}^{-1} - \mathbf{t})\mathbf{R}^{-1} = P_{world} \tag{3.22}$$

From this step, it is possible to compute s as $Z_w = 0$ and constant for all points.

Once s has been computed it is possible to obtain the real world coordinates of a point given its position in pixels from the image with the next formula:

$$P_{world} = (sP_{image}\mathbf{K}^{-1} - \mathbf{t})\mathbf{R}^{-1} \tag{3.23}$$

This function will be executed in loop, once for each frame of the video sequence. The input will be a 2x1 vector with the coordinates in pixels and will come from the tracking module. This point will represent the base of the object tracked. The output will be a 3 dimension vector with the value Z_w going to zero. Relevant data from this vector will be X_w and Y_w, that will be useful when obtaining data, as it is the position of the athlete on the field.

3.5. Applications

The final goal of this project is to provide athletes and trainers with statistics. The fact that this project helps to a real problem gives a great value to it. The main objective of this module is to compute different diagrams and data with many applications. To do this it is necessary to execute all previous blocks in order to obtain 2D position of one or more players observed from a zenith plane. This view, allows trainers to studied players position as a tactical board.

3.5.1. Field area detection

A great tool that can help trainers is to be able to detect which player is out of its position. In order to calculate that, it is possible to define a region of interest ROI and detect if a player is located inside it. Once the player is inside this area, its representation point can change its colour or any warning can appear into the graphic.

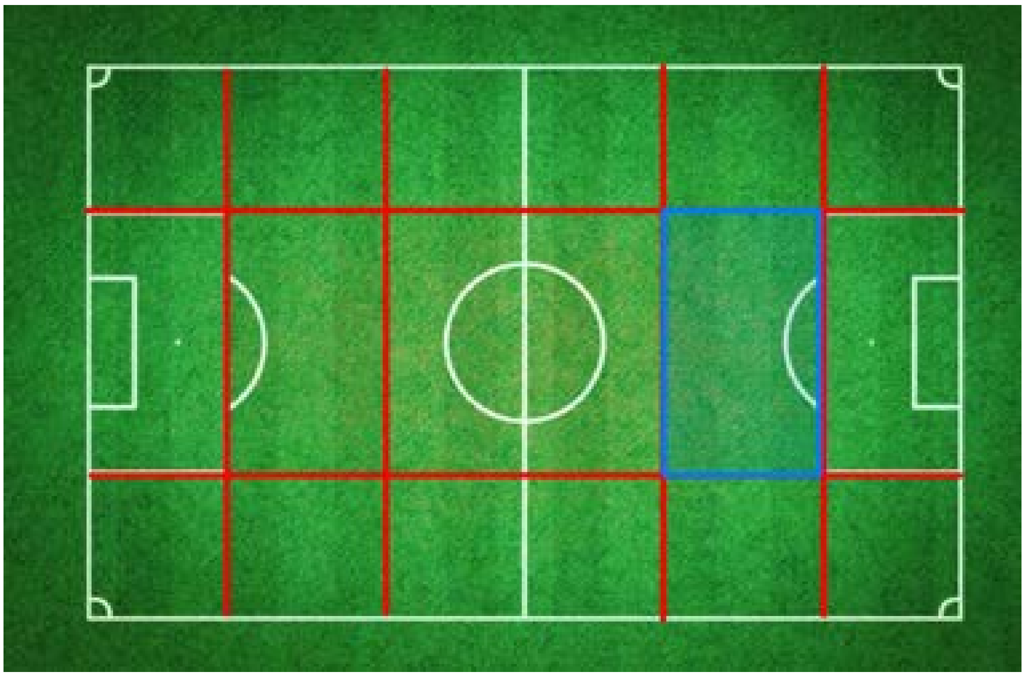


Fig. 3.21. Field area

3.5.2. Heat map

A heat map is a graphical tool that represent every position where the player has passed during a period of time. Heat maps are widely spread through others sports such as basketball, rugby, tennis, etc.. This map is computed by generating a matrix of size *lengthxwidth* of the field initialized to all zeros. Each time player position is tracked, a 1 must be added to the position (x, y) of the array. At the end of the sequence, this array is normalized and its intensity is represented with a colour range.

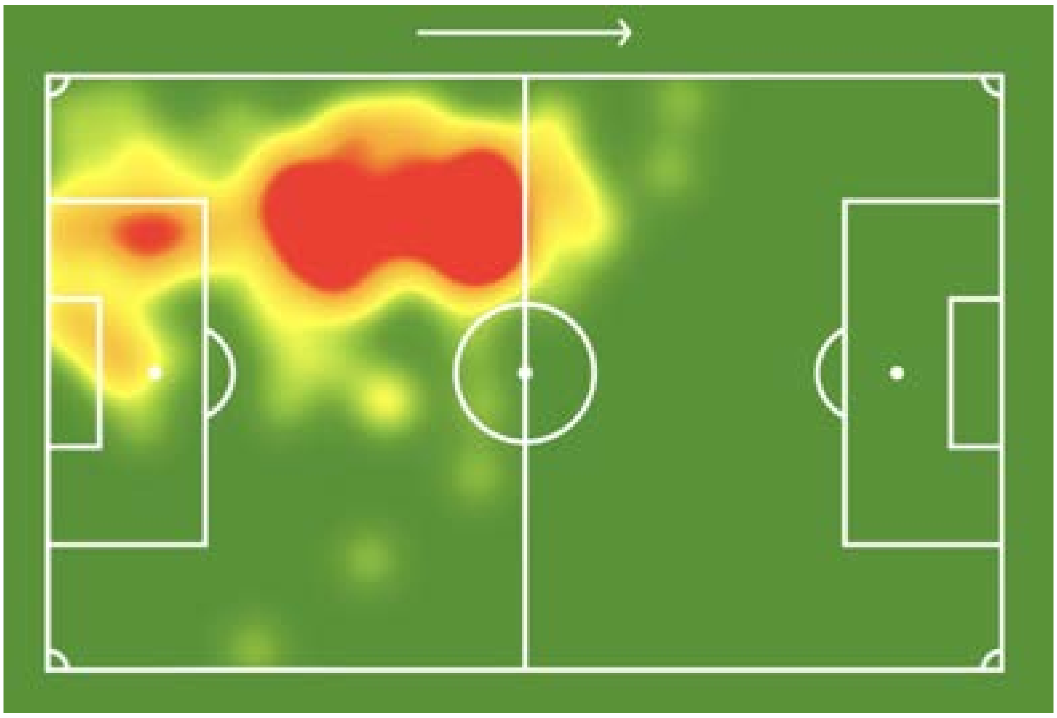


Fig. 3.22. Heat map [30]

3.5.3. Distance between players

Distance between players can help to detect bad positioning. If this application is taken to the edge, it is possible to optimize strategies in a very precise way. This variable is computed by calculating the module of a vector between two points in 2 dimensions, where these points are given as pixels coordinates.

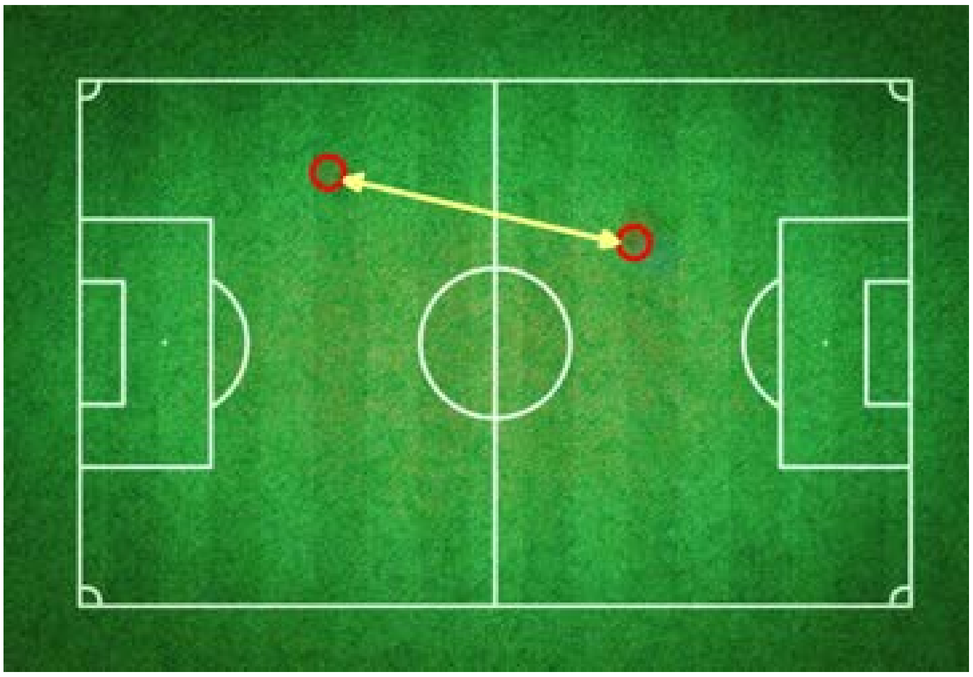


Fig. 3.23. Distance between players

4. STUDY AND EVALUATION OF RESULTS

During this chapter, each of the three main technical modules of the system will be analyzed on its own. Video sequences used in this chapter are the base cases described in chapter two, a square of size 10x10 meters with one and two people. The final application with these videos will be shown at the end of this chapter.

In the previous chapter, it has been explained different possibilities for developing each module, while in this chapter, it will be decide which of these options are the ones that suits best the necessities of this project. Some of the decisions that are going to be made for each module are:

- **Detection and classification:** At this point,the two options are detection with SVM+HOG and detection with YOLOv3 neural network. In order to evaluate these methods, it will be measured the percentage of success and detection time.
- **Tracking:** There are three trackers that are mostly used nowadays. The first one is MOSSE tracker that is the fastest one but with low accuracy. The second one is KCF tracker which provides a balance between precision and speed. The last one is CSRT tracker which is slower but gets better accuracy results.
- **Homography:** For this block it is necessary to evaluate the precision obtained from transformation and calibration. The way to evaluate this, will be by transforming a set of known points from the image to real world coordinates and computing the percentage of error.

4.1. Detection and classification

In this section it will be compared and evaluated the two main methods analyzed in the previous chapter, SVM+HOG and YOLOv3 convolutional neural network.

For evaluating this module it will be analyzed the whole video sequence frame by frame analyzing if there exist a person in the image. By doing this, it will be possible to obtain a set of data which will help to decide which method is better.

The first parameter that will be evaluated during this section is the percentage of frames where the program detects a person over all the total of frames. This parameter will help to determine the precision of the detector and it is relevant because the detector an be activated at any frame. The way to evaluate the accuracy will be done by classifying images into two types of frames, detection OK or detection MISS. This percentage of accuracy is computed by:

$$\text{Percentage of accuracy} = \frac{\text{Detection OK frames}}{\text{Total frames}}100 \qquad (4.1)$$

The second parameter to evaluate is the detection mean time. This parameter is crucial because when the detector is activated it is required to detect as fastest as possible. Mean detection time will be computed as the mean of the seconds used by the detector for each frame.

To make this evaluation in equal conditions, the input video sequence will have a size of 1280x720 pixels, which is the same size used for calibration. Following figures shows one example of each method:

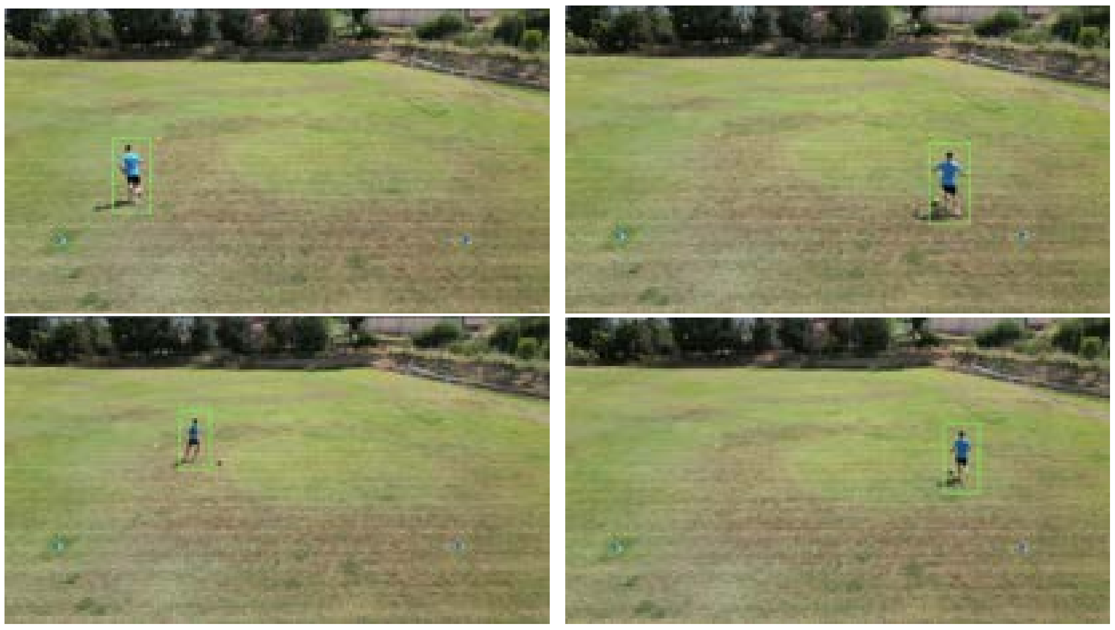
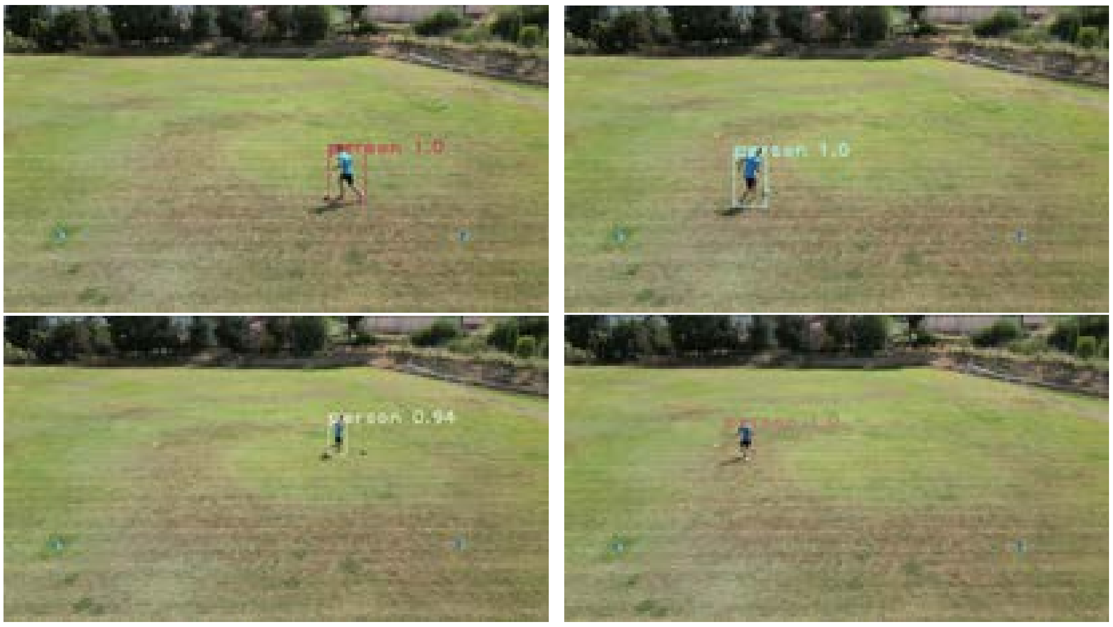


Fig. 4.1. Detection example using SVM+HOG]



Detection example using YOLOv3]

Fig. 4.2.]

After executing the code results are analyzed and compared. This comparison is shown in the following table:

TABLE 4.1. DETECTION RESULTS

TRACKER	SVM+HOG	YOLOv3
MEAN DETECTION TIME [s]	0.283	0.352
TOTAL DETECTION OK FRAMES	681	1675
TOTAL DETECTION MISS FRAMES	1008	14
TOTAL FRAMES	1689	1689
PERCENTAGE OF ACCURACY [%]	40.32	99.17

Source: Experimental results

After having compared the different results, it can be concluded that YOLOv3 gets better results than SVM+HOG. In this case accuracy and time are both relevant, but the gap between both when talking about accuracy is so differential that it is worth to carry this time delay payload.

It is significant to mention that YOLOv3 can improve its detection time by implementing this neural network with Tensorflow instead of OpenCV. This method can not be implemented for this project since it is a very broad topic and the required hardware is not available.

4.2. Tracking

In this section it will be decided which of the three trackers will be used depending on the characteristics of the project. The three trackers are MOSSE, KCF and CSRT. These trackers are implemented in OpenCV and thus its implementation becomes easier.

For evaluating this trackers in equal conditions the same starting bounding-box has been determined for the first frame of the sequence. This bounding-box is adjusted to the limits of the body and its size is 77x184 pixels.

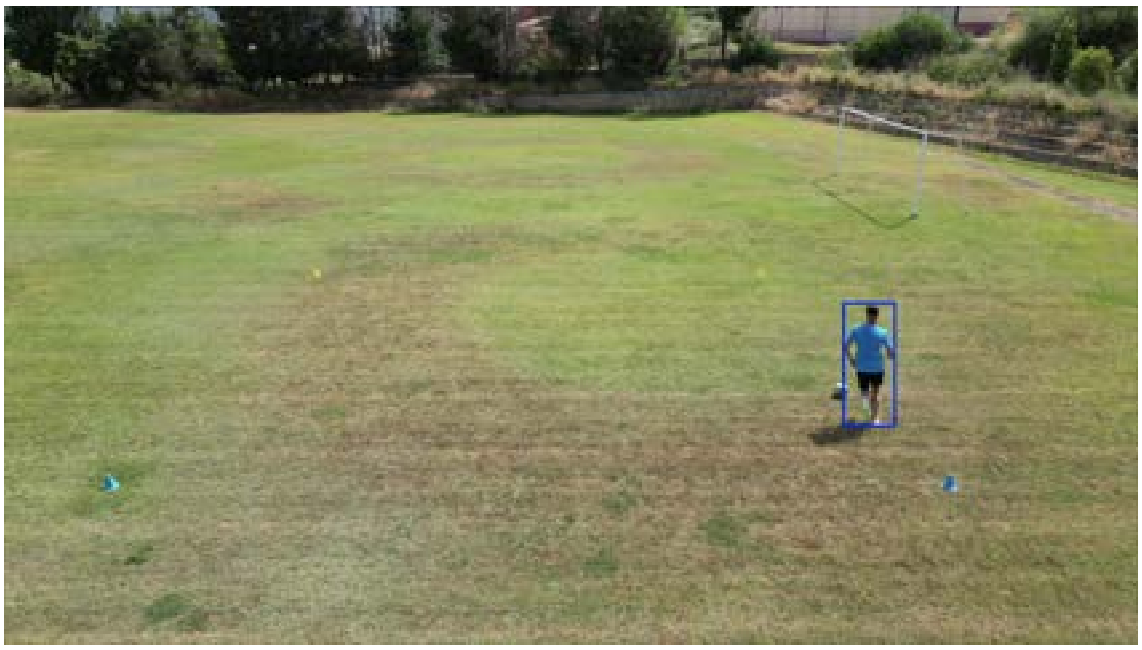


Fig. 4.3. Bounding-box of the fist frame (common for the three trackers)

After initializing these trackers, the tracking of the video sequence is started and stopped in the frame number 600. When choosing this frame, it is possible to evaluate how the tracker evolves in time.

For evaluating these trackers some parameters must be defined in order to quantify the results. These parameters are:

- **Frames per Second (FPS):** It will measure how many time it will take to track a video sequence of an certain time.
- **Percentage of accuracy of the bounding-box:** This parameter will evaluate the percentage of accuracy of the bounding-box respect to the perfect box. The way of computing this parameter will be done by the next formula:

$$\text{Percentage of accuracy} = \frac{\text{Area of the perfect bounding-box}}{\text{Area of the resulting bounding-box}} 100 \tag{4.2}$$

- **Number of losses:** This parameter counts the amount of losses a tracker suffers during the same video sequence.

In the next three figures it is shown the bounding-boxes of each tracker evaluation.

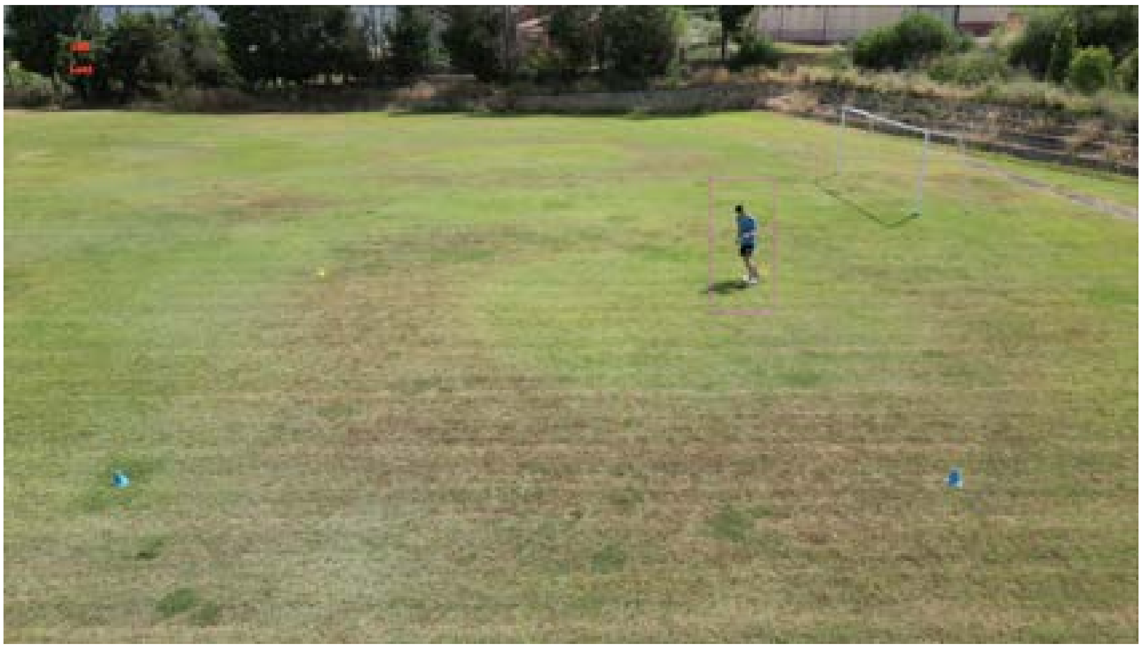


Fig. 4.4. Resulting bounding-box from MOOSE tracker

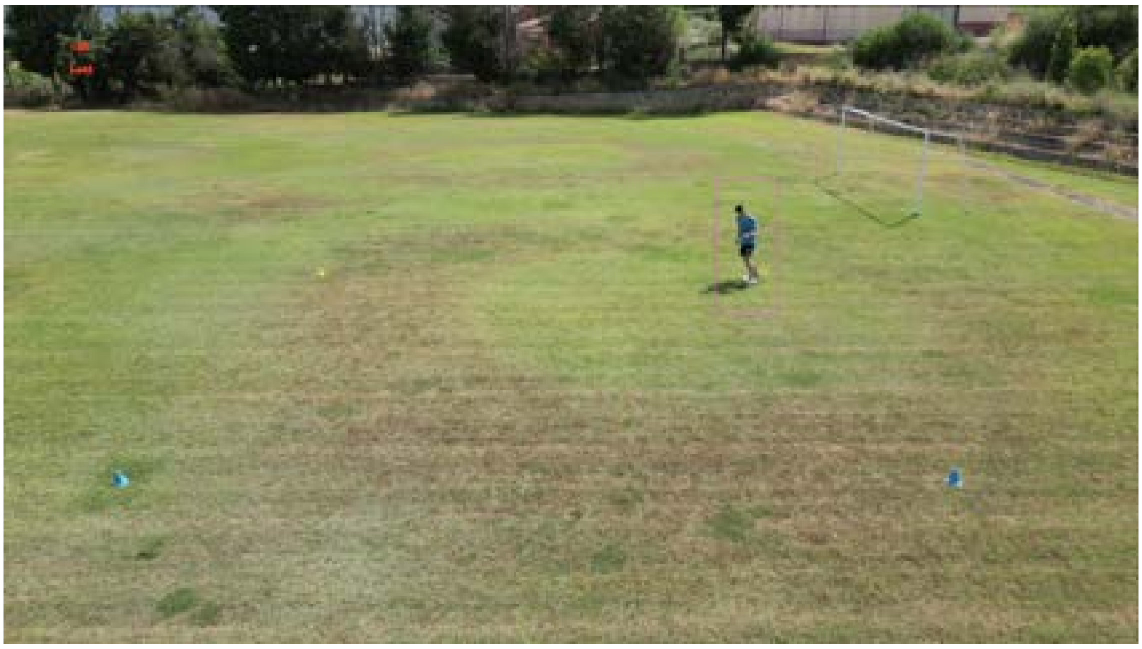


Fig. 4.5. Resulting bounding-box from KCF tracker

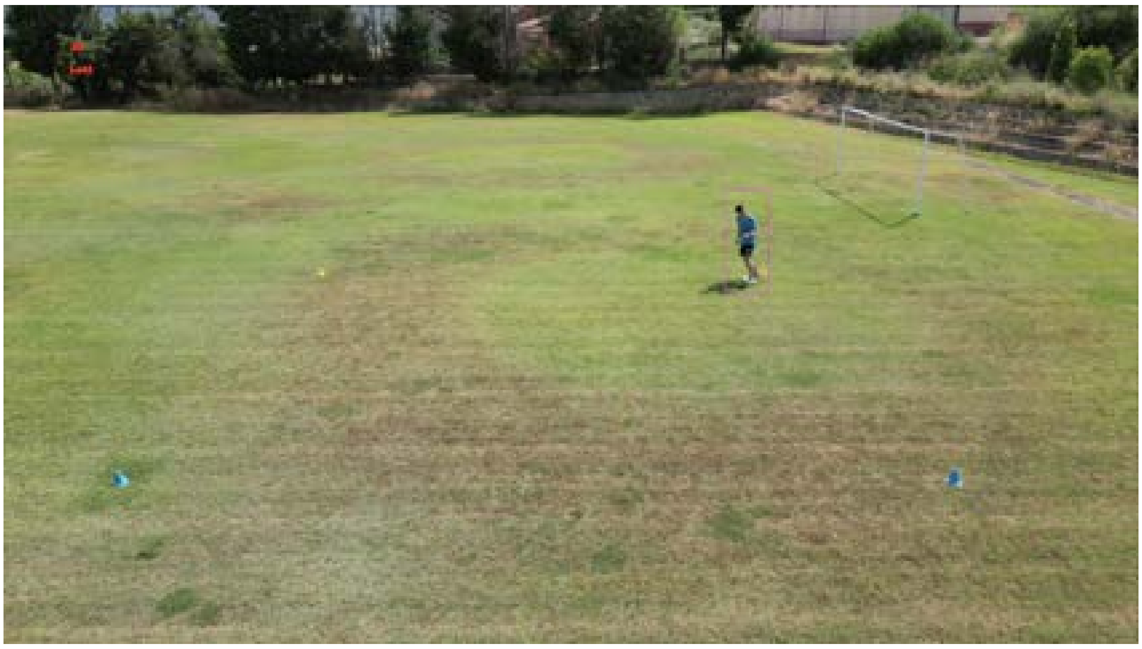


Fig. 4.6. Resulting bounding-box from CSRT tracker

After executing this code, each of the bounding-boxes is measured in pixels and the number of mean FPS is obtained. The results are exposed in the next table.

TABLE 4.2. TRACKING RESULTS

TRACKER	MOOSE	KCF	CSRT
MEAN FPS	41.4	37.4	17.7
MEAN LOSSES	3	1	0
RESULTING BOUNDING-BOX SIZE [pixels]	$95X198 = 18810$	$93X202 = 18786$	$65X155 = 10075$
PERFECT BOUNDING-BOX SIZE [pixels]	$38X121 = 4598$	$38X121 = 4598$	$38X121 = 4598$
PERCENTAGE OF ACCURACY [%]	24.44	24.47	45.64

Source: Experimental results

Mean FPS and mean losses are computed from the repetition of 10 times analyzing this video. The percentage of accuracy represents the amount of pixels inside the resulting bounding-box that match with the perfect bounding-box.

According with the requirements of this project, it is necessary to obtain a certain degree of accuracy and time. For this module, time has enough margin to be placed a secondary parameter and it is necessary to prioritize accuracy.

The two parameters that will influence in accuracy are mean losses and percentage of accuracy. If talking about these two aspect the best tracker is CSRT. This tracker will make possible to locate people more precisely and will make an improvement not needing to execute the detection so often.

4.3. Homography

During this section it will be analyzed how accurate is the transformation between image coordinates and real world coordinates in order to fulfill the requirements for this project. For computing homography, it is indispensable to calibrate the camera by calculating intrinsic parameters.

The first step for calibrating the camera is to compute the intrinsic parameters matrix of the camera. Zhang's method is applied to calibrate the camera. It is necessary to take some chessboard pattern images from different angles in order to obtain the matrix.

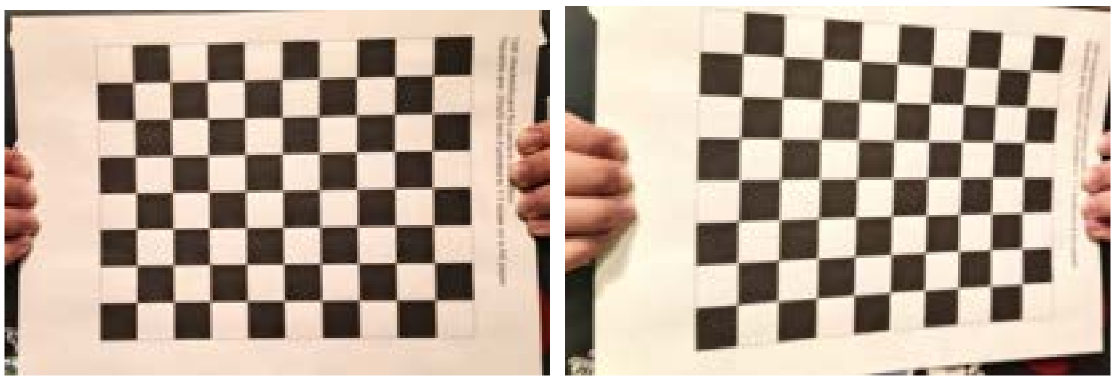


Fig. 4.7. Detection example using SVM+HOG]

The calibration matrix for this specific case (DJI mini 3 pro drone camera) is :

$$M = \begin{bmatrix} 924 & 0 & 650 \\ 0 & 926 & 346 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

And the distortion vector is:

$$dist = \begin{bmatrix} 0.350 \\ -2.60 \\ 5.66e-03 \\ 6.12e-03 \\ 6.31 \end{bmatrix} \tag{4.4}$$

Once M and $dist$ has been computed, the next step is to get the extrinsic parameters of the camera. This parameters depend on the position of the camera respect to the real world. For doing this it is required to have a set of known points in both, image and real world coordinates system.

For evaluating calibration and homography, the process which will be used is the inverse. Once the camera is fully calibrated, the goal will be to obtain the real world coordinates of a set of known points given its image coordinates. These results will be compared with known real world coordinates and extract error margin.

Next figure will a set of known points selected in the image with its real world known coordinates and used for computing extrinsic calibration.

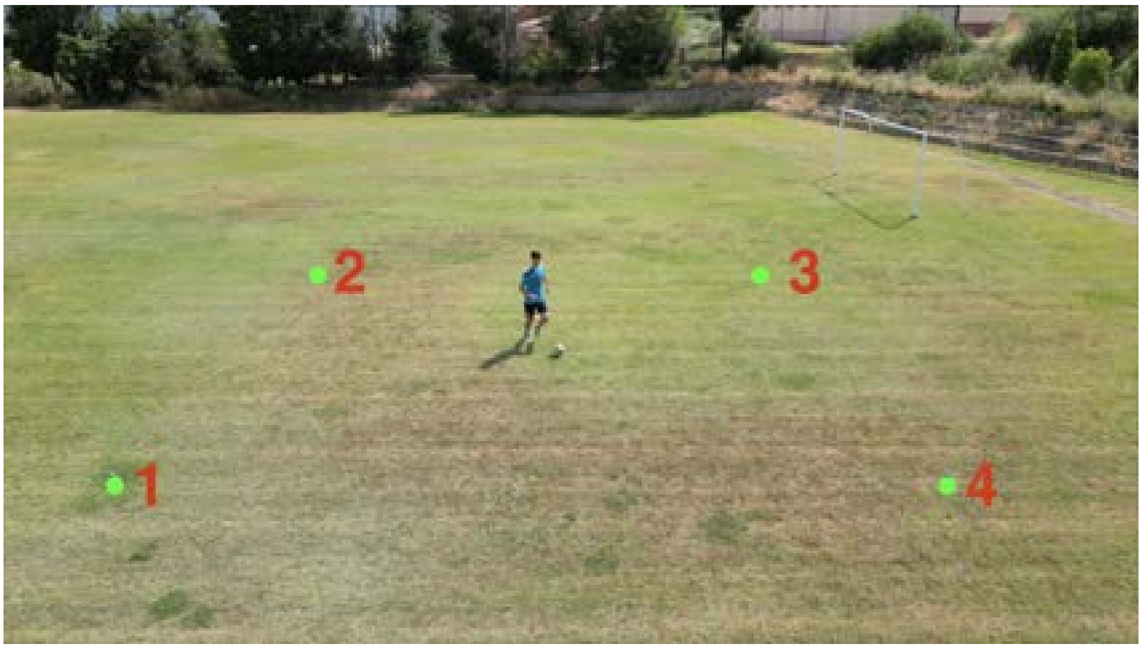


Fig. 4.8. Used points for obtaining extrinsic parameters)

In this case, distance is represented as centimeters. As the distance between corners is 10 meters, it will be used 1000 centimeters as the unit used for this case.

For calculating the percentage of error it is necessary to compute the distance between the real world point and the resulting point as the module of a vector. As z resulting coordinate is negligible, it will be assumed as zero and this module will be computed in a 2D coordinate system. The percentage of error is computed as:

$$\text{Percentage of error} = \frac{\text{Distance between points}}{\text{Total distance between corners}} \cdot 100 \tag{4.5}$$

$$\text{Percentage of error} = \frac{\text{Distance between points}}{1000} \cdot 100 \tag{4.6}$$

TABLE 4.3. HOMOGRAPHY RESULTS

POINT	1	2	3	4
IMAGE CO-ORDINATES [pixels]	(125, 541)	(354, 304)	(853, 304)	(1064, 541)
REAL WORLD CO-ORDINATES [centimeters]	(0, 0, 0)	(0, 1000, 0)	(1000, 1000, 0)	(1000, 0, 0)
RESULT CO-ORDINATES [centimeters]	(-15.17, -21.91, -1.42·10^{-14})	(-22.58, 963.02, -1.14·10^{-13})	(1041.05, 1029.49, 0)	(1002.15, -18.74, -5·10^{-14})
PERCENTAGE OF ERROR [%]	2.66	4.33	5.05	1.87

Source: Experimental results

This module meets the requirements specified for this system but a minimum angle for the camera is needed. This angle has been calculated from this video sequence and camera so it can vary for any other situation. In this case, it is possible to compute an estimation of this angle by using the point number 3, with an error margin of near to 5%. The following figure represent a top and a lateral view of the position of the camera respect to the rectangle. Units are represented in centimeters.

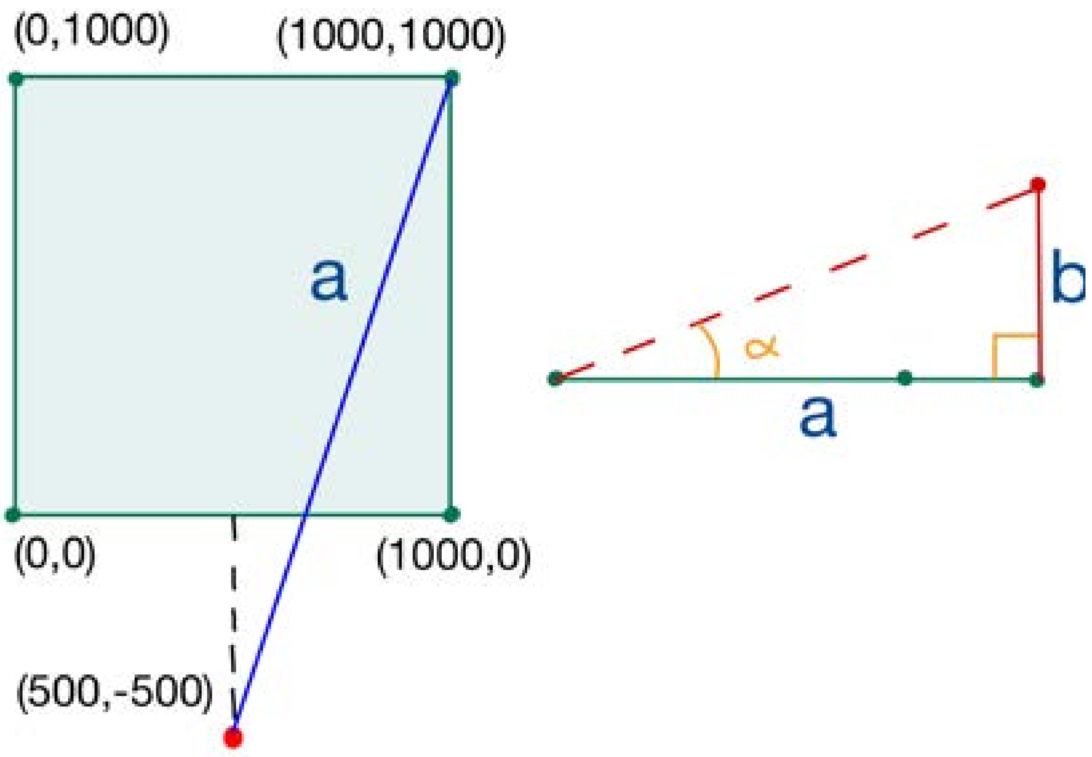


Fig. 4.9. Representation of the relative position of the camera

Firstly, the distance *a* is computed as the module of a vector and *b* is measured while recording:

$$a = \sqrt{(1000 - 500)^2 + (1000 + 500)^2} = 1581,14cm \tag{4.7}$$

$$b = 420cm \tag{4.8}$$

α is computed as:

$$\alpha = \arctan \frac{b}{a} = 15^\circ \tag{4.9}$$

So it is possible to determine a minimum angle of 15° in order to obtain an error of less than a 5%.

4.4. Final results

In this section it will be exposed the two main cases described in this project.

4.4.1. Case 1

In this case it is used a DJI mini 3 pro as camera at a height of 4.2 meters and 5 meters away from the square. The duration of this video is one minute and the processing time is 1 minute and 51 seconds. Following images show the results.

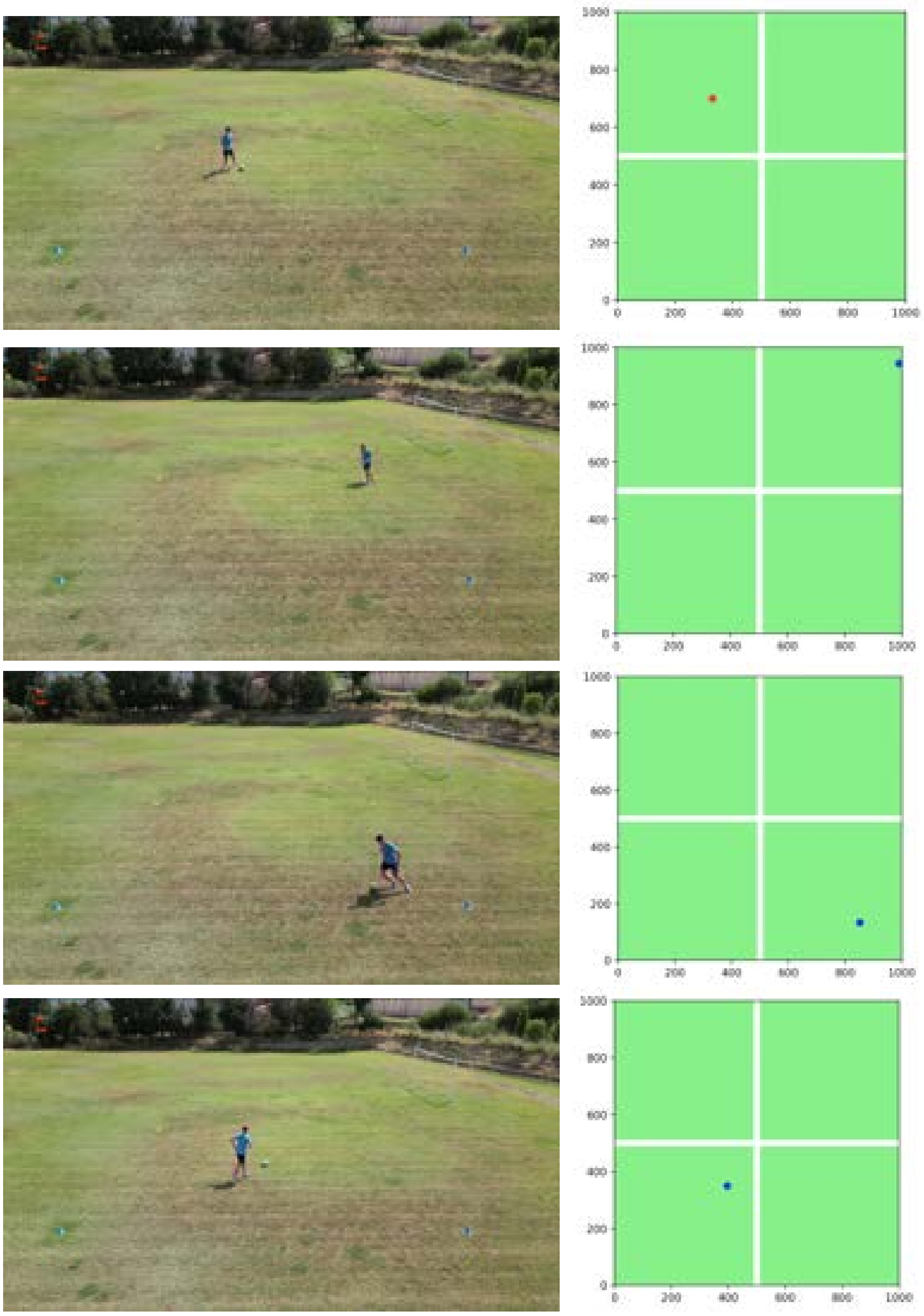


Fig. 4.10. Case 1 screenshots

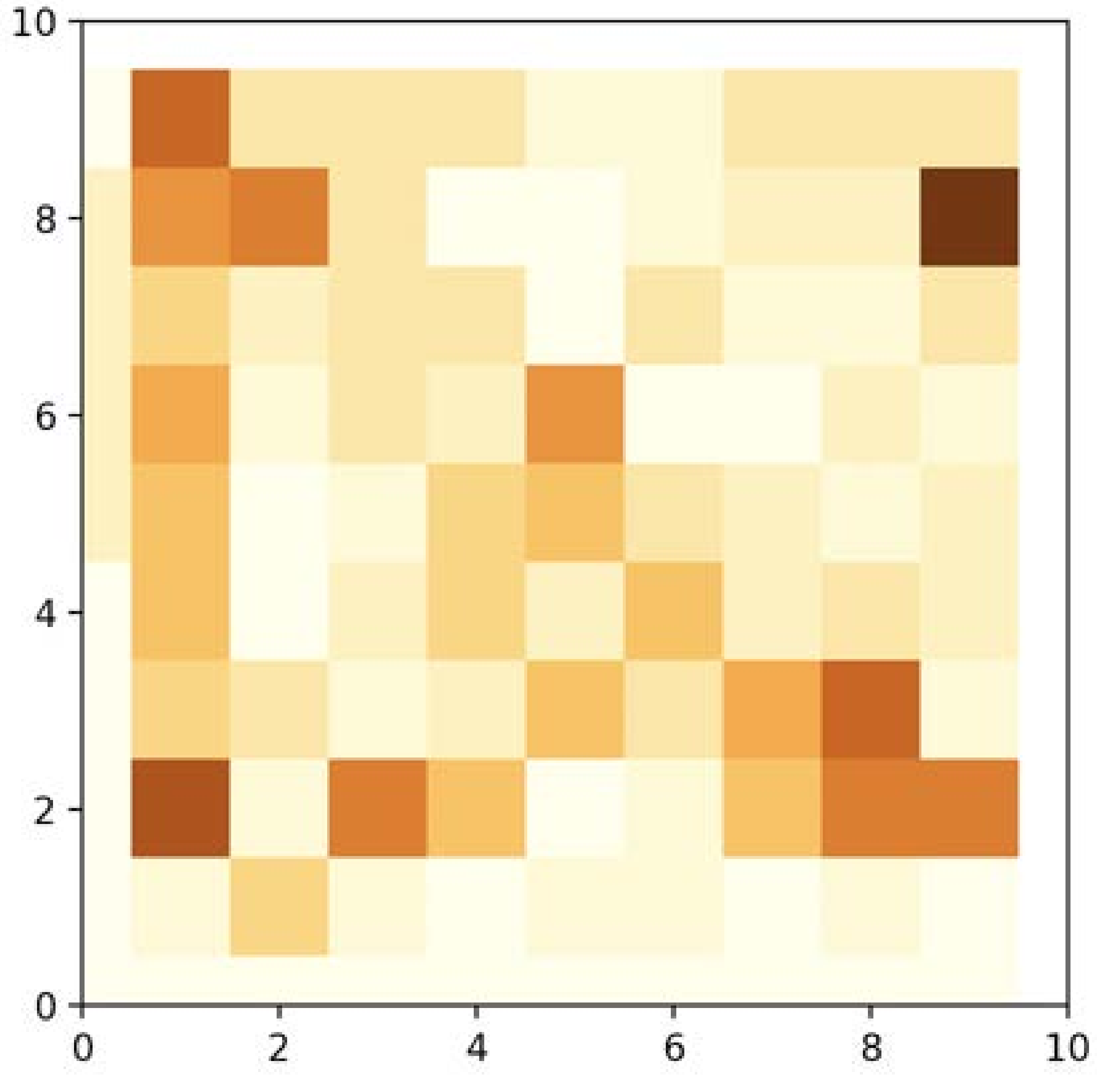
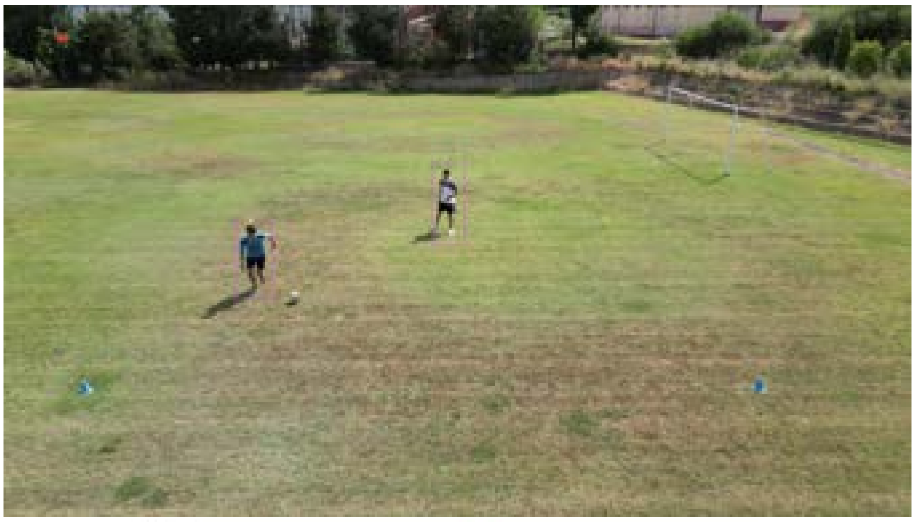


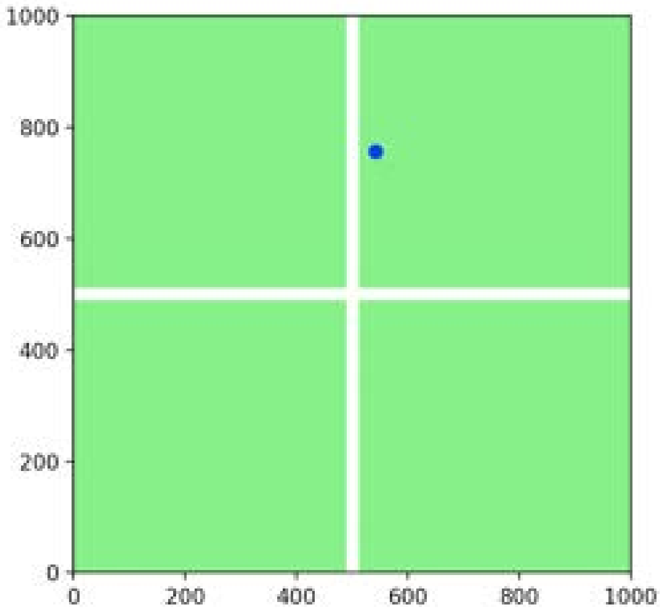
Fig. 4.11. Case 1 heatmap

This video sequence fulfill the requirements of the system.

4.4.2. Case 2

In this case it is used a DJI mini 3 pro as camera at a height of 4.2 meters and 5 meters away from the square. The duration of this video is one minute and the processing time is 3 minutes and 20 seconds. Following images show the results.





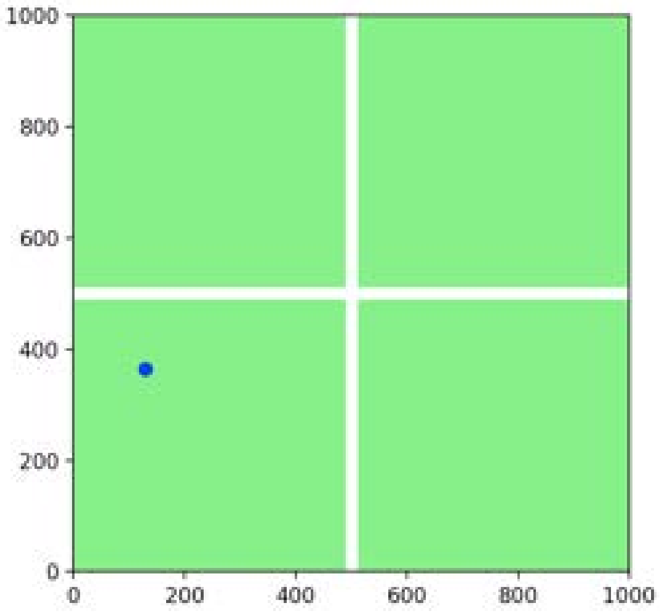


Fig. 4.12. Case 2 screenshots (1)

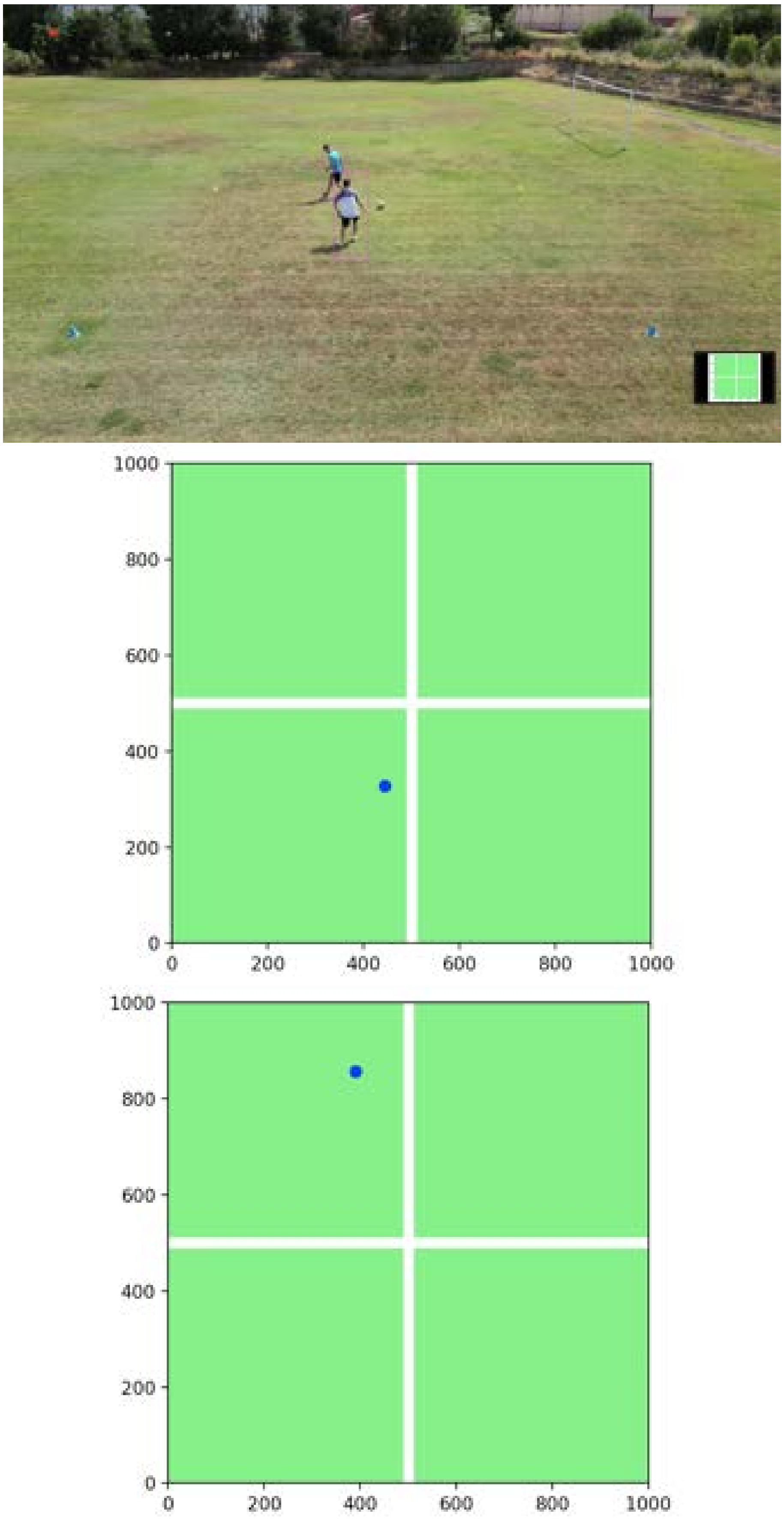
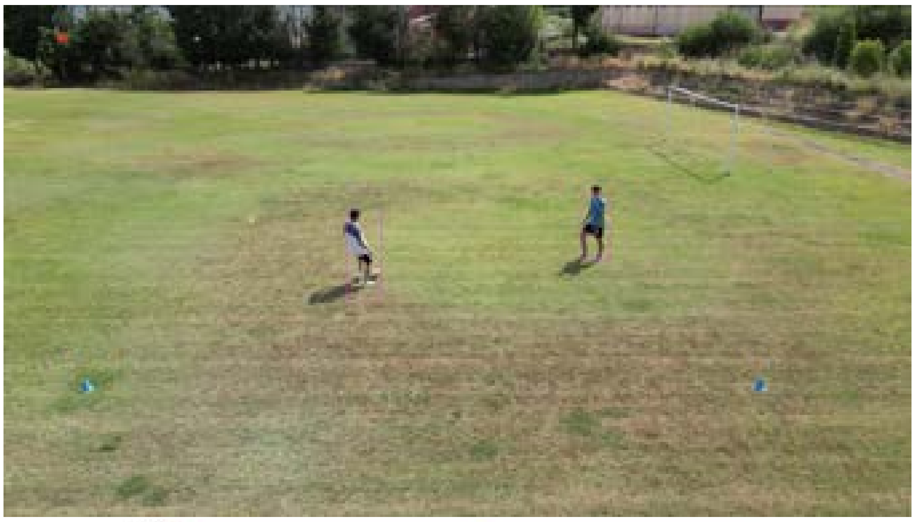
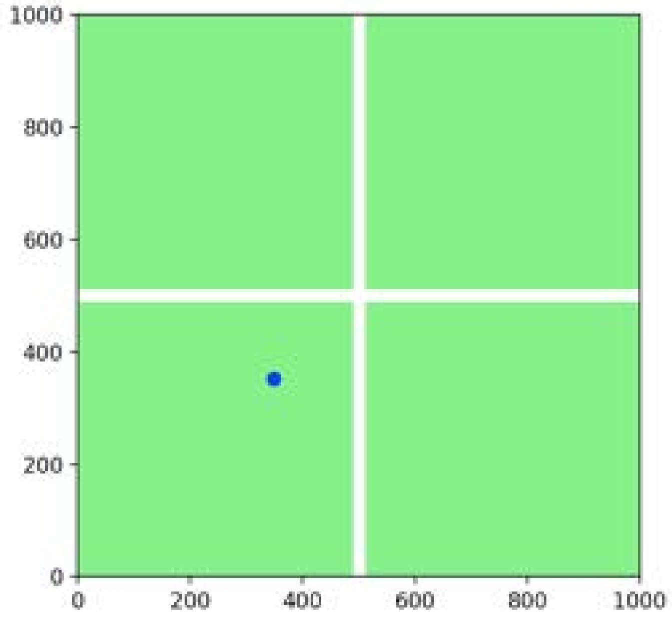


Fig. 4.13. Case 2 screenshots (2)

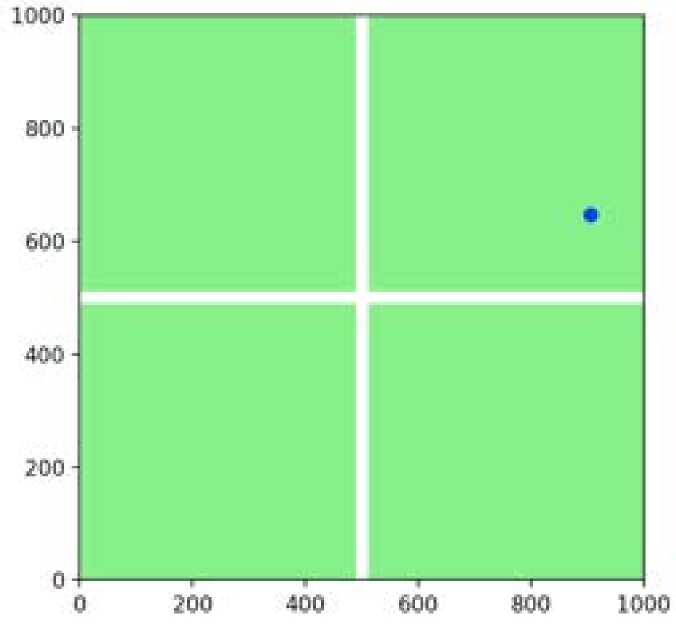


Fig. 4.14. Case 2 screenshots (3)

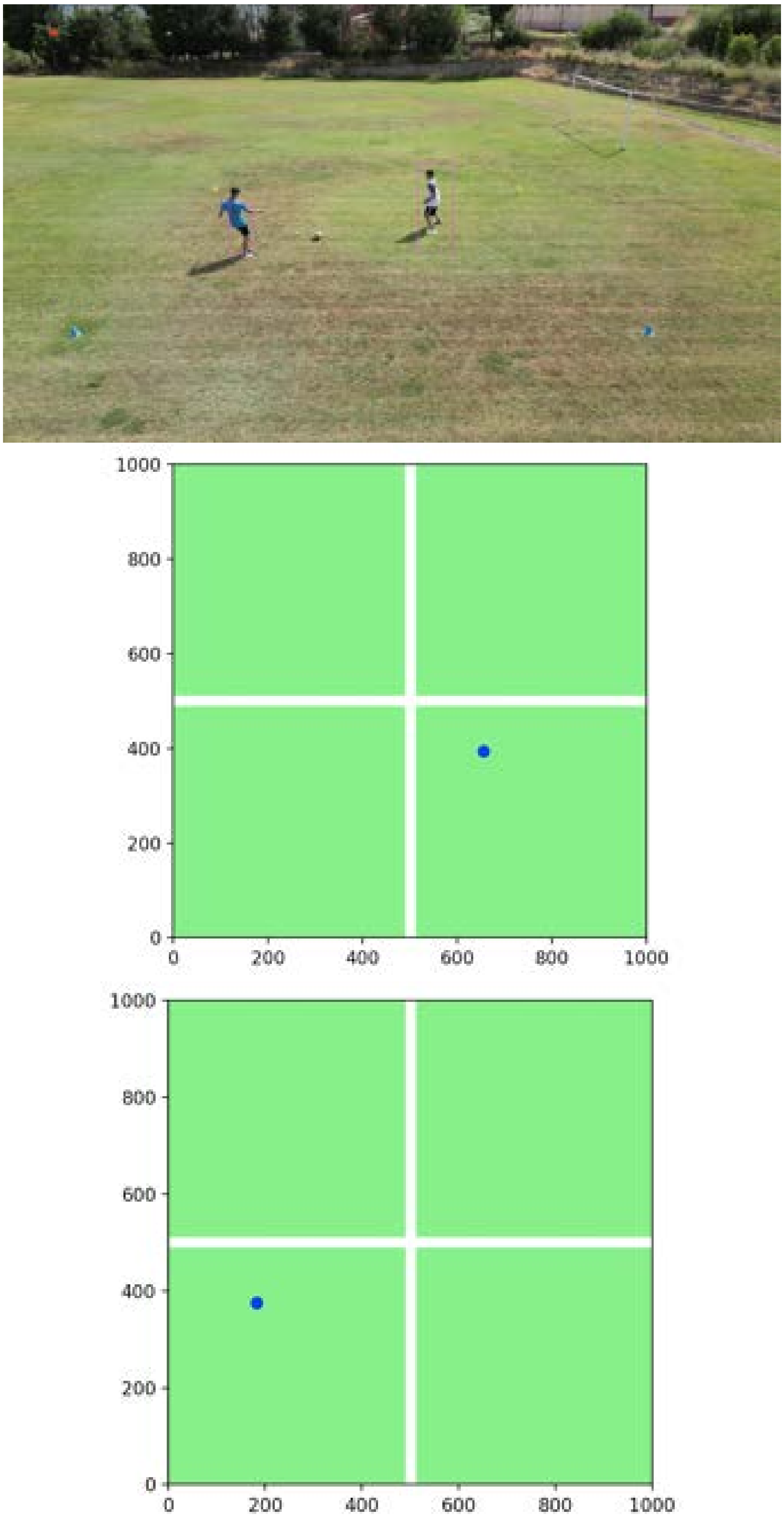


Fig. 4.15. Case 2 screenshots (4)

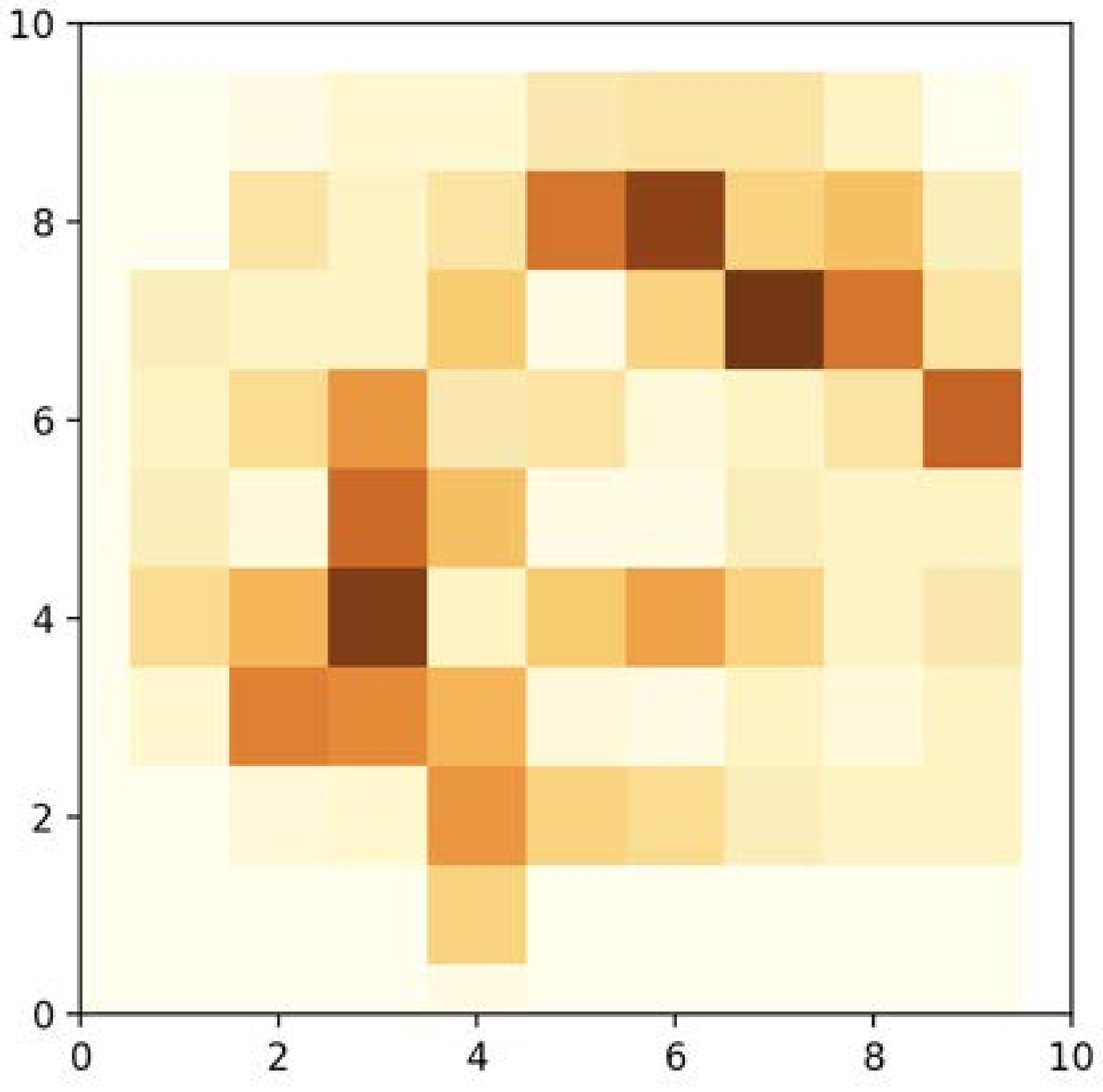


Fig. 4.16. Case 2 heatmap

This video sequence fulfill the requirements of the system.

5. CONCLUSION

During this project a tool capable of detecting, tracking and transforming coordinates from 2D to 3D has been developed. The initial intentions of this project were to provide athletes and trainers an affordable video analysis tool to work with, facilitating their work and improving the quality of training.

At the beginning of the project, a series of minimum technical characteristics were defined with which to meet the practical requirements. From them, the four basic blocks of the system were defined: detection, tracking, homography and data representation. Each of this blocks have been studied individually and different alternatives have been proposed to solve the proposed problem.

The first of the functions to be developed has been the detection and classification of people in an image. For this, different detection methods have been investigated, which has allowed learning about the world of object detection and neural networks. The final decision for the implementation of this block has been to use YOLOv3 as object detector, resulting in a bounding-box surrounding the person.

The second block is object tracking applied to video. During this module, different trackers offered by the OpenCV library have been investigated. After performing different tests, it has been concluded that since the most difficult requirement to meet is accuracy, the CSRT tracker has been chosen as the tracking method.

The third block of the main ones is the transformation from image coordinates to real world coordinates. For this, it has been necessary to study different camera calibration methods and it has allowed to acquire knowledge about different camera parameters and how they affect the image. After this, a transformation from 2D to 3D has been carried out in such a way that given a point in an image, it is possible to calculate the point in real life.

The last of the blocks is the representation of the data through different graphs. For this it has been necessary to find out which were the most relevant graphics for the coaches and for this reason they have been implemented.

As a final conclusion it can be said that this project has allowed to acquire a greater knowledge about video analysis and open source libraries such as OpenCV. It has also allowed us to have an approach with different people from the world of football, who have shown great interest in the possible evolution of this project. Therefore, it can be said that this project has laid the foundations for possible future work.

5.1. Future lines

The idea addressed in this project is in a first phase of development and research. That is why the system has several lines of evolution and different approaches. In addition, the different applications that this project has are many and varied, which opens up another wide range of possibilities when it comes to research in the future.

Some of the main future lines are listed and explained below:

- **Live analysis:** during the development of this project, live video analysis tests have been done, obtaining some positive results. Live tracking has been achieved by lowering the transformation and detection frequency. In this way it is possible to estimate the position of a player on the field three times per second.
- **Tensorflow:** the implementation of this tool would improve the speed of analysis, making live video analysis more feasible. Testing would be necessary but at first glance, it would slow down the processing speed.
- **Full-team analysis:** another future implementation to be able to turn this project into a product is to be able to analyze a complete team. In this way it would be possible to obtain more data and it would be more attractive for trainers.
- **Recording with drones:** one option when recording video is from a drone. A good line of research might be to develop a way to stream video directly to a computer and analyze it in real time.
- **More applications:** this software can be used for different goals and functions. Some examples are traffic management, flow of people in airports, etc. These new future lines can be investigated and provide great added value to the project.

BIBLIOGRAPHY

[1] *Nacsport: Software de videoanálisis para todos los deportes*, Nacsport.com, 2022. [Online]. Available: https://www.nacsport.com/index.php?lc=es-es (visited on 06/09/2022).

[2] *Wyscout*, Wyscout, 2017. [Online]. Available: https://wyscout.com/ (visited on 06/09/2022).

[3] Newsline, *Ross lanzó una nueva versión de piero*, Newsline Report, Nov. 2020. [Online]. Available: https://www.newslinereport.com/tecnologia/nota/ross-lanzo-una-nueva-version-de-piero (visited on 06/09/2022).

[4] *Mediacoach*, Mediacoach.es, 2022. [Online]. Available: https://www.mediacoach.es/ (visited on 06/09/2022).

[5] D. A. .-. ENAIRE, *Enaire drones*, Enaire.es, 2022. [Online]. Available: https://drones.enaire.es/ (visited on 06/08/2022).

[6] J. Couto, *A guide to video analytics: Applications and opportunities*, Tryolabs, Oct. 2019. [Online]. Available: https://tryolabs.com/guides/video-analytics-guide (visited on 06/09/2022).

[7] *SVM kernels accuracy and generalization capability on apnea detection from ECG*, Jul. 2010, pp. 187–191. DOI: 10.13140/RG.2.1.4573.8325.

[8] K. Nguyen, N. Huynh, P. Nguyen, K. Duy, D. Vo, and T. Nguyen, "Detecting objects from space: An evaluation of deep-learning modern approaches," *Electronics*, vol. 9, p. 583, Mar. 2020. DOI: 10.3390/electronics9040583.

[9] W. Contributors, *Kalman filter*, Wikipedia, Jun. 2022. [Online]. Available: https://en.wikipedia.org/wiki/Kalman_filter (visited on 06/09/2022).

[10] H. Norris, *Why should you choose python for real-time applications?* Medium, Aug. 2019. [Online]. Available: https://medium.datadriveninvestor.com/why-should-you-choose-python-for-real-time-applications-8422a30f167d (visited on 06/05/2022).

[11] *About - opencv*, OpenCV, Nov. 2020. [Online]. Available: https://opencv.org/about/ (visited on 06/05/2022).

[12] *Comparison of support vector machines and deep learning for vehicle detection*, Nov. 2018.

[13] W. Contributors, *Histogram of oriented gradients*, Wikipedia, Apr. 2022. [Online]. Available: https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients (visited on 06/06/2022).

[14] *Hog descriptor*, Intel, 2022. [Online]. Available: https://www.intel.com/content/www/us/en/develop/documentation/ipp-dev-reference/top/volume-2-image-processing/computer-vision/feature-detection-functions/histogram-of-oriented-gradients-hog-descriptor.html (visited on 06/06/2022).

[15] V. Meel, *Yolov3: Real-time object detection algorithm (what's new?) - viso.ai*, viso.ai, Jan. 2022. [Online]. Available: https://viso.ai/deep-learning/yolov3-overview/ (visited on 06/06/2022).

[16] *Deep neural network-based robust ship detection under different weather conditions*, Oct. 2019. DOI: 10.1109/ITSC.2019.8917475.

[17] J. Redmon, *Yolo: Real-time object detection*, Pjreddie.com, 2012. [Online]. Available: https://pjreddie.com/darknet/yolo/ (visited on 06/06/2022).

[18] pjreddie, *Github - pjreddie/darknet: Convolutional neural networks*, GitHub, Mar. 2022. [Online]. Available: https://github.com/pjreddie/darknet (visited on 06/06/2022).

[19] J. Prakash, *Non maximum suppression: Theory and implementation in pytorch*, LearnOpenCV – OpenCV, PyTorch, Keras, Tensorflow examples and tutorials, Jun. 2021. [Online]. Available: https://learnopencv.com/non-maximum-suppression-theory-and-implementation-in-pytorch/ (visited on 06/06/2022).

[20] J. Cruz Martinez, *Facebook*, Livecodestream.dev, 2022. [Online]. Available: https://livecodestream.dev/post/object-tracking-with-opencv/ (visited on 06/06/2022).

[21] S. Mallick, *Object tracking using opencv (c++/python)*, LearnOpenCV – OpenCV, PyTorch, Keras, Tensorflow examples and tutorials, Feb. 2017. [Online]. Available: https://learnopencv.com/object-tracking-using-opencv-cpp-python/ (visited on 06/07/2022).

[22] A. Kim, *Camera calibration - towards data science*, Medium, Feb. 2021. [Online]. Available: https://towardsdatascience.com/camera-calibration-fda5beb373c3 (visited on 06/05/2022).

[23] D. Bhatt, *A comprehensive guide for camera calibration in computer vision*, Analytics Vidhya, Oct. 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-for-camera-calibration-in-computer-vision/ (visited on 06/05/2022).

[24] K. Sadekar and S. Mallick, *Camera calibration using opencv | learnopencv*, LearnOpenCV – OpenCV, PyTorch, Keras, Tensorflow examples and tutorials, Feb. 2020. [Online]. Available: https://learnopencv.com/camera-calibration-using-opencv/ (visited on 06/05/2022).

[25] A. Cioppa, A. Deliège, S. Giancola, O. Barnich, B. Ghanem, and M. Droogenbroeck, *Camera calibration and player localization in soccernet-v2 and investigation of their representations for action spotting floriane magera* evs broadcast equipment*. (visited on 06/05/2022).

[26] X. Nie, S. Chen, and R. Hamid, *A robust and efficient framework for sports-field registration*. (visited on 06/05/2022).

[27] F. Coldefy, R. Guerchouche, and F. Coldefy, *Camera calibration methods evaluation procedure for images rectification and 3d reconstruction camera calibration methods evaluation procedure for images rectification and 3d reconstruction*. (visited on 06/05/2022).

[28] unknown, *Figure 1. pinhole camera model: Ideal projection of a 3d object on a 2d...* ResearchGate, May 2017. [Online]. Available: https://www.researchgate.net/figure/Pinhole-Camera-Model-ideal-projection-of-a-3D-object-on-a-2D-image_fig1_326518096 (visited on 06/05/2022).

[29] Gluón, *Estimación de posición con opencv y...* Lab. Gluón, Feb. 2020. [Online]. Available: https://www.laboratoriogluon.com/estimacion-de-posicion-con-opencv-y-python/ (visited on 06/06/2022).

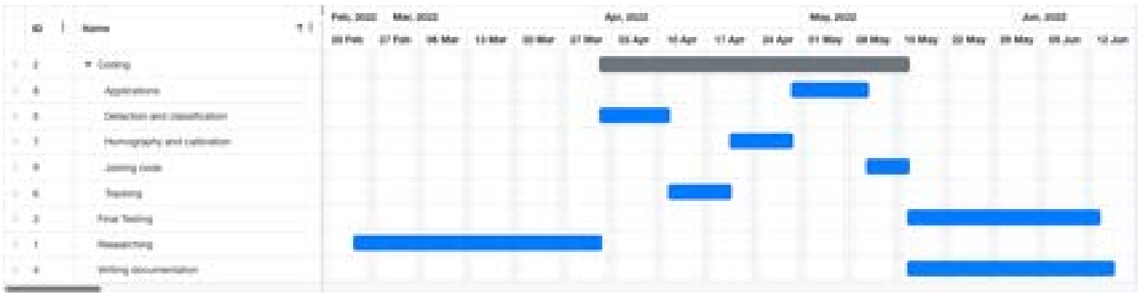
[30] K. Pratap, *What is a heat map? and why it has become essential in the current sports industry:* Indifoot, May 2021. [Online]. Available: https://www.indifoot.com/blog/what-is-a-heat-map-and-why-it-has-become-essential-in-the-current-sports-industry (visited on 06/07/2022).

APPENDIX 1 - PLANNING

This project could be divided into four phases:

1. **Researching:** it is the first step and the longer one.
2. **Coding:** it could be divided into the four modules of the system.
    - Detection and classification
    - Tracking
    - Calibration and homography
    - Applications
3. **Testing:** it is done in parallel with documentation.
4. **Writing documentation:** it is done in parallel with testing. It is the last part, once the code is finished.

The distribution in time is shown in the next Gantt diagram.



APPENDIX 2 - OTHER EXAMPLE OF VIDEO ANALYSIS

Another example of video analysis for a different sport, padel.

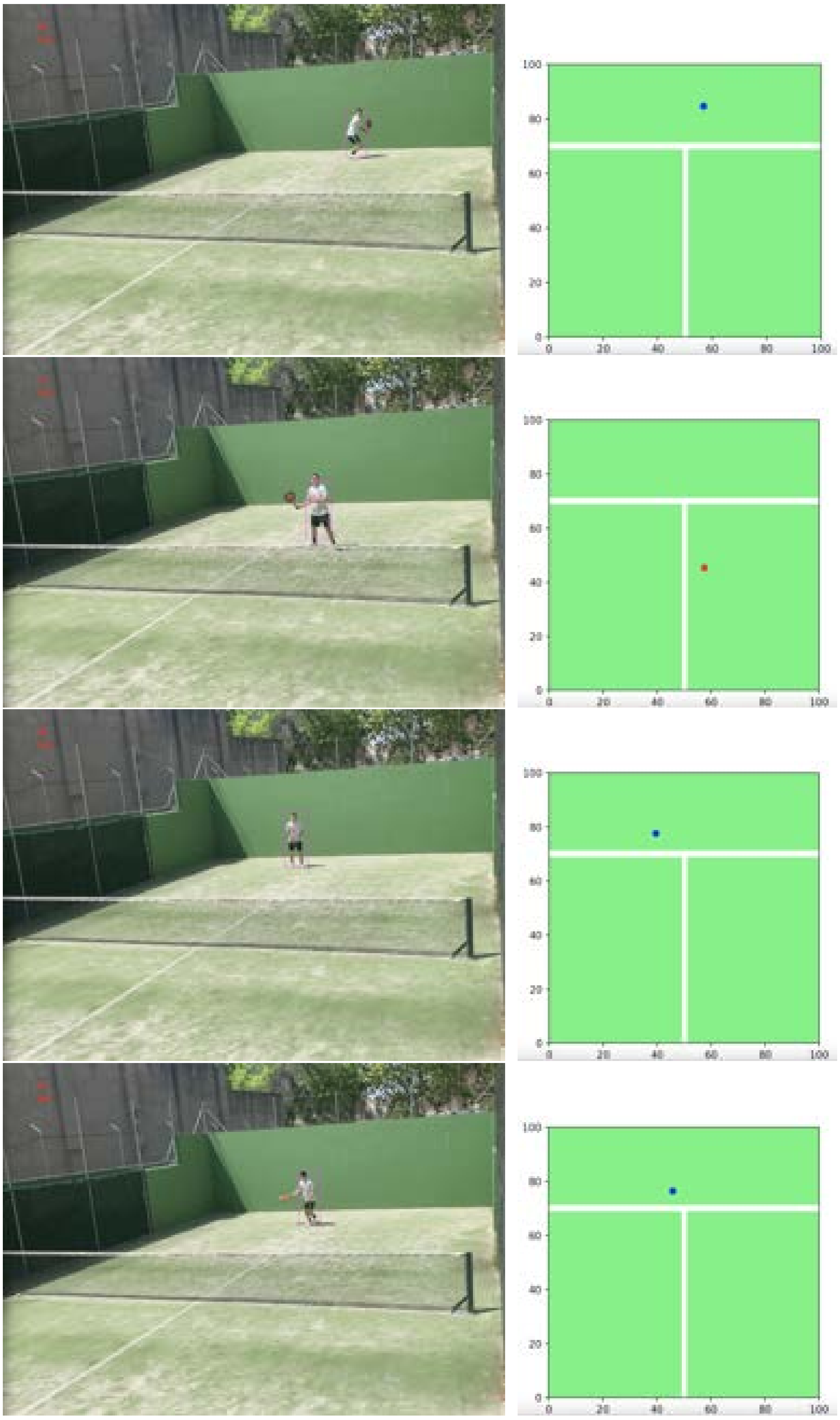


Fig. 5.1. Case 3 screenshots

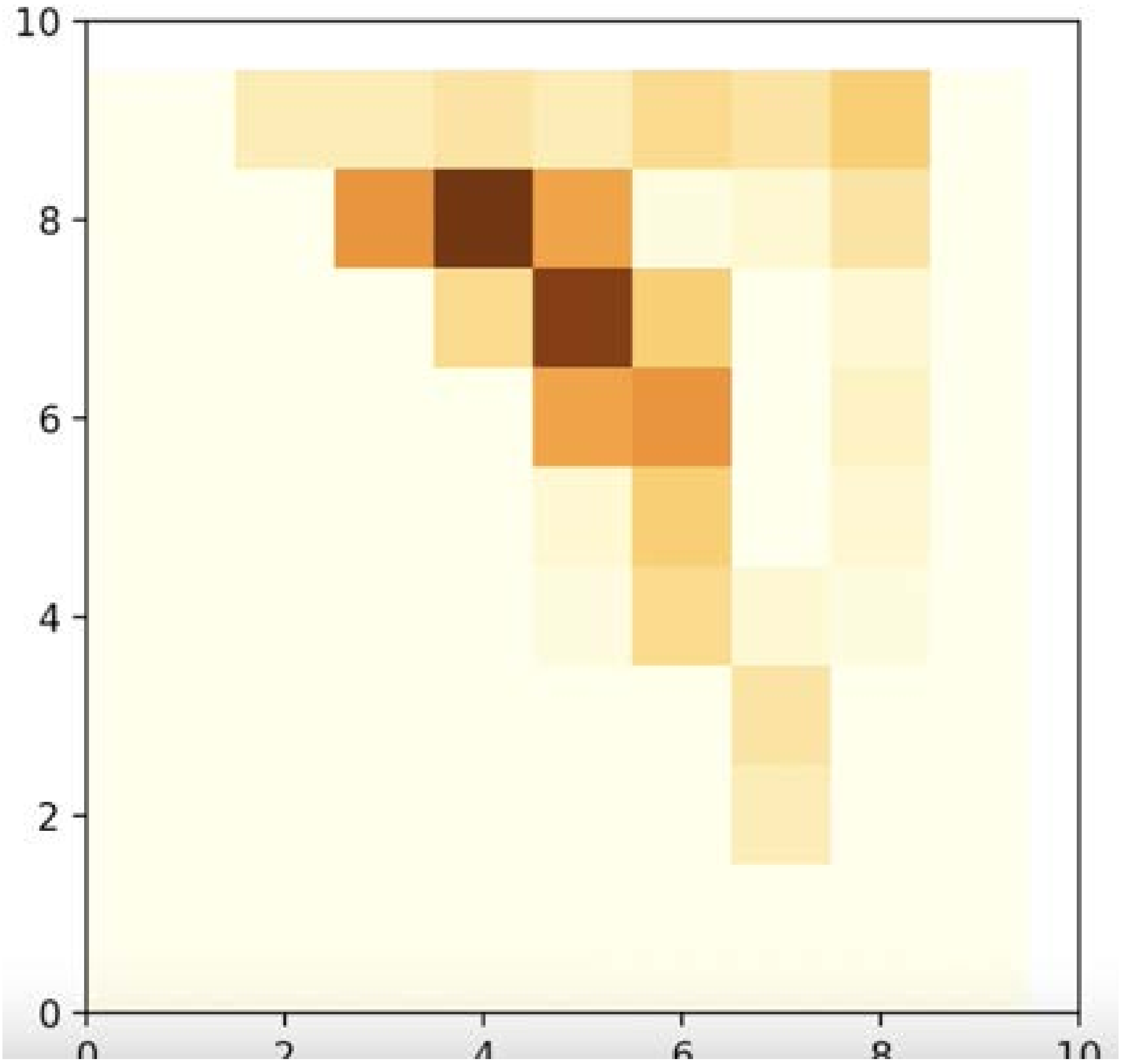


Fig. 5.2. Case 3 heatmap