UNIVERSIDAD POLITÉCNICA DE MADRID

Aplicaciones Telemáticas Avanzadas

Referee Robot: Changing the paradigm of Football

Rubén Mascuñana, Roberto Padovano, Jose Casasola, Sergio Sacristán, Adrián Azplicueta y Juan Luis Miguel

Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación – Universidad Politécnica de Madrid

Abstract

We have a clear motivation for this project: to alter the referee in football. They implemented VAR in 2018, which was a complete disaster, therefore our project will represent a paradigm shift in sport. As a result, our research focuses on knowledge representation and reasoning.

How are we going to go about doing it? Our team plans to build a system that consists of a large number of high-precision cameras and a specific product called LiDAR, which will be in charge of collecting the exact images for, the central neural system, which will collect all of these images to make the correct decision at any given time, such as calling offside or foul using a specific algorithm and AI based on TensorFlow.

Keywords: Football, Referee, Neural Networks, AI, Knowledge Representation, Reasoning, LiDAR, Tensorflow, YOLO, Python

Document Version: 2.0 (Hito_5) Date: 10/05/2022

1. Introduction

Who has never really watched a football game? Football (soccer in the United States) is the most popular sport on the planet; we all know how good Brazilians are with the ball, or how passionate Argentines are about their clubs, which is passed down from generation to generation.

Soccer is the most popular sport in Spain, according to the CSD (High Sports Council) [1], with 32 percent of Spaniards kicking a ball. Furthermore, 185,000 individuals in Spain make their living from football, which means that their employment is inextricably linked to the sport, resulting in football accounting for 1.37 percent of the country's GDP [2].

We all understood the significance of football, but we never considered everything that may influence the game, such as money, people, emotions, and so on. We also need to convey the major concepts and rules of the sport to anyone who is unfamiliar with it.

Soccer is a team sport that began in 1863 in England with the formation of the Football Association (FA). The basic principle is that two teams of 11 players with a clear objective compete to see who can score the most goals. In the attempt to see who gets most of the goals there are some controversial plays where both teams are not able to agree on the decision to be made. To solve this problem, Soccer invented a new role called referee, a neutral person who will oversee making the correct decision based on a clearly defined regulation.

Nonetheless, because of the effect of his choices on the field, the referee has always been in the spotlight. We all make mistakes and make bad decisions, but in soccer, you may determine a game by whistling a foul that isn't a foul or sending off a player who didn't deserve to be sent off. To summarize this concept, there are two key points to consider:

First is the human error, we must assume that the referee is a human that is not ubiquitous in the field so he cannot witness everything that occurs and make the correct decision in a matter of seconds without affecting the game.

Second, there is the issue of interpretation, which is one of the most serious issues. Although international soccer organizations such as FIFA and UEFA have regulations, these rules might be interpreted differently depending on who you ask (one example could be the force applied when stealing a ball from the opponent or the intention intent to do harm in a struggle at a corner).

To summarize, our suggestion is to correctly employ technical breakthroughs, as opposed to other methods that combine AI and humans and do not address any of the issues raised above. We're going to revolutionize soccer's paradigm by removing the human referee and replacing it with an AI referee.

2. State of Art

2.1. Ongoing Research

Back then in 2010's, the decision that the referee and linesman took were far less precise and slow than they are now and much obliged to the use of software, cameras and even other referees sitting outside of the stadium and evaluating the frame that this camera gives them the sport improve exponentially.

Experience enables us to be more practical, since technology had been implemented in football, the way of seeing it had changed. Due to the number of data and information which is given, for example the number of throws, passes, offsides, etc.… Technology brings more attention to the sport, all this data is public, so the viewer evaluates by itself what think it is more important.

It is obvious that Covid made the sport a little less attractive, even though it was thought that fans were going to lose the feeling of watching football, technology put the facts on the table, and this kept the collective 'sense of belonging' alive, irrespective of where the football fans are. This was not only a problem for the fan but also for the footballers that did not receive the warmth of the fans, so companies like Yamaha found the solution developing a Remote cheerer.

Offside calls have been under focus for the past years, loads of software tried to reduce the number of mistakes taken by referees but it seemed to conclude with the opposite result leaving fans disappointed.

Slow motion cameras are not only seen as an improvement for the sport when making decisions by the referee, it is also providing useful information for the fans at home.

Technology had been increasing its use in the past decade as a necessary matter, innovation way of reaching the maximum expected and as revolution for what it is known as "old school". It all started with cameras, when back then thinking of watching football at home was thought to be impossible and nowadays it is common.

2.2. Existing Technologies

Many technologies are now being used to improve the rules of sport, such as high framerate cameras, referee assistance like VAR, Hawkeye, and so on. All of these technologies have one thing in common: none of them can completely replace the human aspect, which is the faultiest. Some technologies not related to sports might be used to suit the needs, such as LiDAR, which would allow real-time rendering of the field so that a software application built

with TensorFlow could make match judgments in real time and without human intervention. We'll go through the various utilities that can be obtained from all these technologies, as well as how to use and apply them in our system.

2.2.1. Hawkeye

Hawk-eye is a sophisticated computer system that is used in a variety of sports to visually follow the trajectory of the ball and present a moving image of its statistically most likely path.

What is Hawk-eye's mechanism of operation? First, we must calibrate the cameras to address the issue of the cameras' non-uniform distance from the playing field, and then we must include a statistical generator to provide statistics based on the data acquired to aid the ball recognition algorithm in predicting the ball's approximate location.

An algorithm is used to find the pixels corresponding to the ball in the image obtained using the size and shape of the ball to achieve it, geometric algorithm is used to look at multiple images and then combine them cleverly to get the ball coordinates in 3d.

Triangulation is the most common used method for determining the location of a point by measuring angles rather than measuring distances to the point directly to find the 2D coordinates of the ball.

In football, software aggregates data from all cameras to monitor the ball, and as soon as the ball crosses the goal line, the system transmits an instantaneous signal to the referee.

We've seen hawk-eye technology in use for many years, but technology is always evolving, and nowadays Foxteen is the most commonly utilized technology in tennis for calling outs. This technique employs 40 cameras that can create 100000 photos per second, allowing the position of the ball to be determined with pinpoint accuracy.

Moreover, Foxteen uses a combination of ultra-high-speed cameras and a high-speed laser scanner technology to track the player's and the ball's movements.

To ensure the best possible measurements, the cameras and lasers are placed near together. Because the ball is registered by up to 5 separate sensors, ordinary impediments have no effect on the calculation. Judges are still required because this technology just allows for perception; they can only view images.

Instead of existing methods that estimate the trajectory projection of the ball using triangulated samples, this next-generation Hawk-eye provides 100 percent accuracy.

Figure 1: Example of the Structure of a Hawk-eye system

2.2.2. LiDAR

LiDAR stands for "light detection and ranging." Also known as "laser scanning" or "3D scanning," this technology creates a 3D representation of the studied area using eye-safe laser beams.

A sensor releases pulsed light waves from a laser into the environment, which bounce off surrounding objects and return to the sensor. The sensor then estimates the distance traveled based on the time it takes for each pulse to return.

The laser, scanner, and GPS receiver are the key components of the LiDAR [3]. It also includes photodetectors and optics, which are in charge of data gathering and analysis. Drones, airplanes, and helicopters can be used to assist in the collection of more accurate data in special instances.

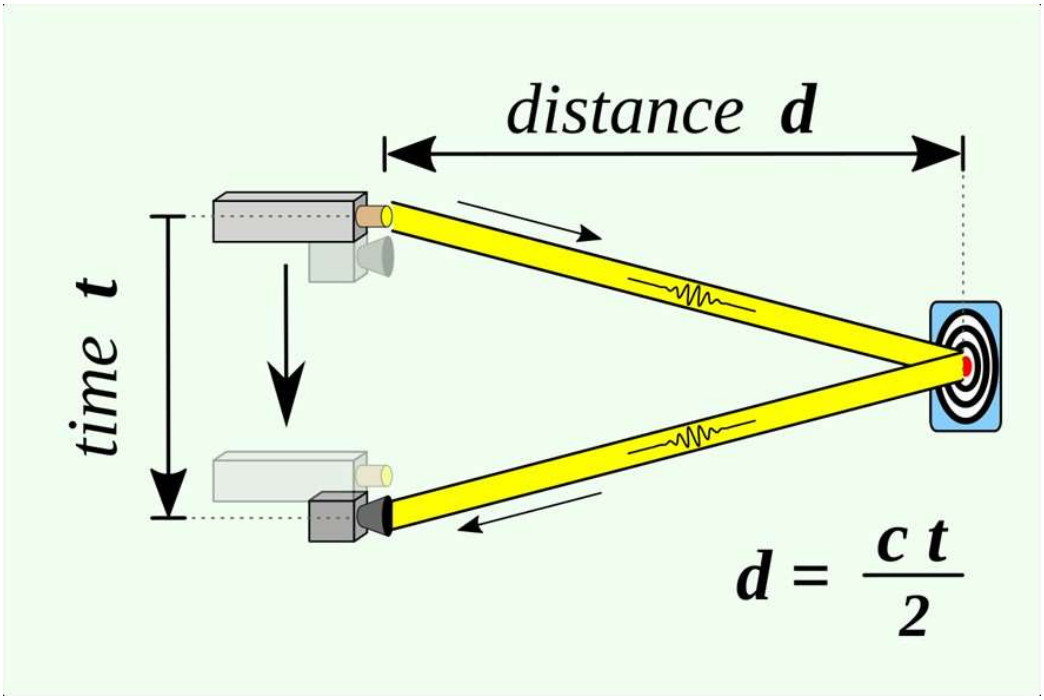


Figure 2: Physical Structure of a LiDAR

It is feasible to generate a 3D map recreation of an exact environment in real time using this information repeated million times per second. One use for this project is to depict the soccer pitch so that an AI can decide the offside.

Figure 3: Example of 3D Image generated by LiDAR

The Geographic Resource Analysis Support System (*GRASS*) [4] is one of numerous commercial and open-source tools and technologies available. Most tools treat and process data in the *LAS3* format, which is a standard format that stores the values x, y, and z, as well as the intensity value and the return number, for every point in the cloud, even though this format is quite flexible and can include data fields specified by the client. *GRASS*, its algorithms, and its ability to work with other tools will be the focus of our discussion.

- As previously stated, *GRASS* is an open-source tool used by *LiDAR*; however, *GRASS* is useless without *QGIS*, which provides access to the data base and capabilities. *QGIS* is a free and open-source Geographic Information System that works with a wide range of geospatial vector and raster files as well as database formats. *GRASS* and *QGIS* can be used for a variety of things.

- Imagery, which allows for atmospheric correction and border detection, is implemented in *GRASS* using Fast Fourier Transformation (FFT).

Topology, which describes the relationship between points, lines, and polygons that represent special objects in a geographic region, is implemented in *GRASS* using algorithms that allow for data topology cleanness (v. clean).

These usages mentioned before are just two of hundreds, but for our project we think that 3D data visualization by plugins like *Gfis2theeejs* and the *GRASS* contribution with the generation of 3D modeling with algorithms of *NVIZ* would help us achieve our purpose.

2.2.3. TensorFlow / Keras

TensorFlow is an end-to-end open-source platform for machine learning, It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy powered ML applications.

Keras is an API of high level (interface of application programming) that can use the functions of TensorFlow based on Phyton. This API simplifies the use of TensorFlow and gives you a great modularity in your projects. In practical terms, Keras was designed for simplify the implementation of hard, but powered tools of TensorFlow.

These two tools are used to create a convolutional neural network, **CNN**, but what is exactly a CNN? [5]. A convolutional neural network is a type of neural network that specializes in processing data that has a grid-like topology such as an image. A digital image is a binary representation of visual data. It contains a series of pixels arranged in a grid-like fashion that contains pixel values to denote how bright and what color each pixel should be.

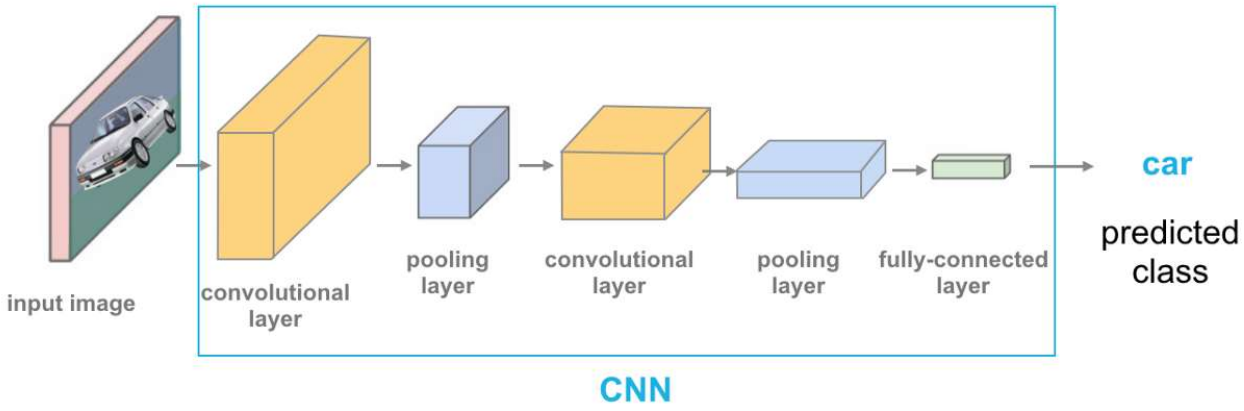


Figure 4: Structure of CNN (Convolutional Neural Network)

First of all, we need the input, obviously, a group of images to be classified. As you can see in this figure, we have three layers inside the CNN:

1. Convolution Layer: Is the core building block of the CNN. It carries the main portion of the network´s computational load. This layer performs a dot product between two matrices, where one matrix is the set of learnable parameters, and the other matrix is the restricted portion of the receptive field. To resume the process, we can see it in this GIF how exactly it works.

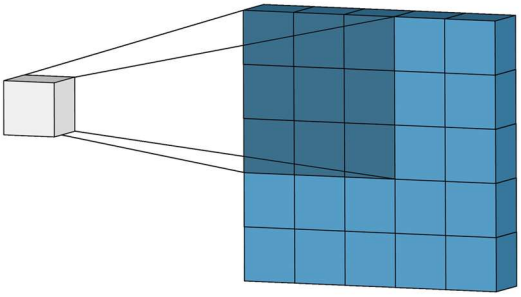


Figure 5: Example of a Convolution Operation

2. Pooling Layer: This layer replaces the output of the network at certain locations by deriving a summary statistic of the nearby outputs. This helps in reducing the spatial size of the representation, which decreases the required amount of computation and weights. The pooling operation is processed on every slice of the representation individually. The pooling operation is well defined by this image.

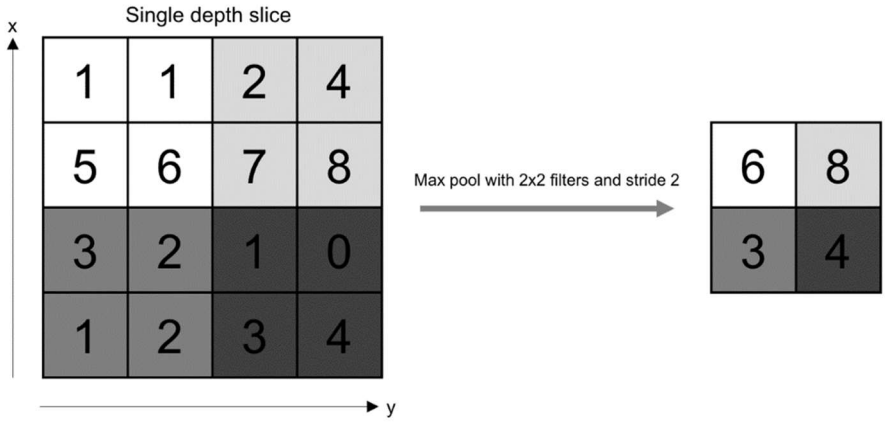


Figure 6: Example of a Pooling Operation

3. Fully-Connected Layer: This layer has full connectivity with all neurons in the preceding and succeeding layer as seen in a regular neural network. His goal is to help to map the representation between the input and output.

All this work will be implanted in our project for the image recognition and processing to analyse if the play is offside or the player used his hand, etc

2.2.4. VAR (Video Assistant Referee)

VAR is an assistant referee system whose major purpose is to prevent blatant human error from determining the outcome of a match in a negative way.

A collection of cameras captures a vast number of images from various views during a game, which are then shown to trained referees in a control booth. The primary referee in the field talks with the production room using these photos, and the VAR referees make the decision and return it to the field.

VAR can only be used in four situations: a potential goal, a penalty, a direct send-off, or identification confusion. However, our initiative will expand on this, allowing us to make more decisions in a faster and more secure manner.

Figure 7: Structure of "VAR Room"

2.2.5. Players Pose Tracking

When developing our project, we have encountered many implementations and other problems. One of the problems that we have found is the monitoring of the players and all their limbs, the soccer ball and other elements involved in a soccer match.

The objective of tracking players is to always know their position and therefore it is necessary to determine exactly the location of all parts of the body in order to have the most precise and exact information possible when making any decision that requires it.

We have decided to implement Skeletal-Tracking technology with the objective of creating a tracking system for the human skeleton to be able to track the user in 360º. This system must two differentiated parts in its architecture, which are the client and the server. This separation is done to facilitate the differentiation of tasks, the integration of functions and the coordination of several clients by the server.

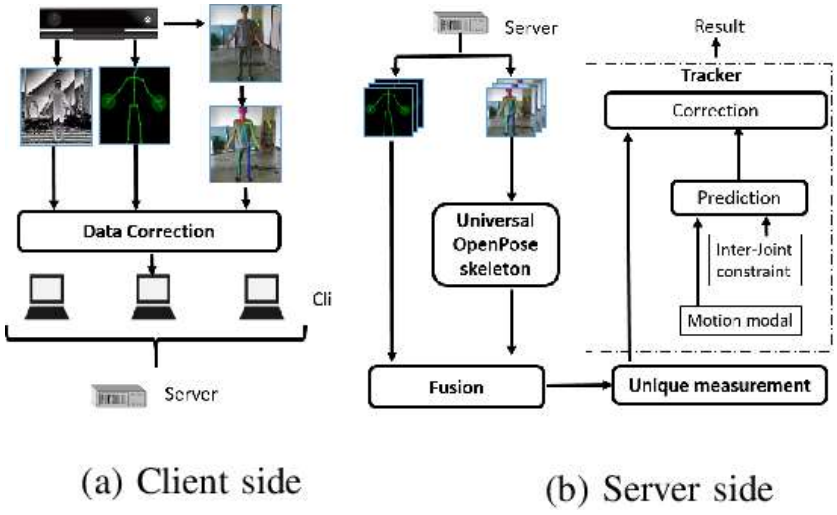


Figure 7: Structure of Skeletal Tracking Framework

Each player (client) which you want to track to process information from its corresponding Kinect camera will send data to the server and the server will integrate the information of all players to track the skeleton of each user. The server is also in charge of sending out synchronization signals to process a frame simultaneously. In this way, when a client receives synchronization signals, the detected Kinect 3D joints and OpenPose 2D joints are sent to the server to be merged.

On the server, the OpenPose 3D skeletons are reconstructed and the OpenPose 2D joints are matched between the various cameras. After reconstructing the skeleton with the OpenPose 2D, ach 3D joint detected by a single camera is then compared to OpenPose 3D and any confusion is treated with a left-to-right sweep.

Before explaining a practice use, it is important to know what it is OpenPose, well OpenPose is a real-time multiple-person detection library, for us it will help us as back-up information to check if our reconstruction of the human body match with the library.

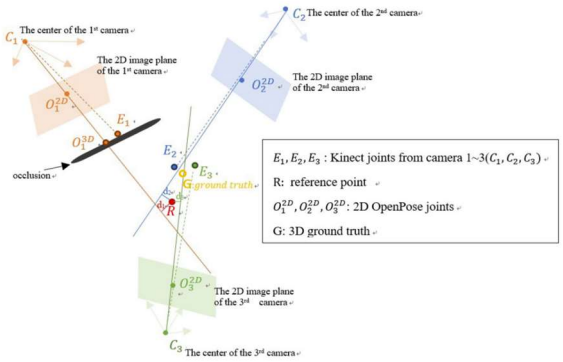


Figure 8: Representation of a Real Problem into the Field

For our project we are going to suppose that the left-right confusion problem on the server side it is solved and let's assume that the angle of occlusion created by the positioning of the humans with respect to the focus of the camera is also fixed.[9]

So now for this use case we will explain what the problem is and how we will fix it. In the image we can see that there is a point called G ((player1) let's suppose that this is the human that we want to track in our server), we will have an angle of occlusion (player 2) already determined that it will be another human in the same direction seen from the camera 1 and with the same position of the arms, head and legs as the G point and lastly we concluded that with 3 cameras positioned orthogonally the problem will be solved.

Now that the situation has been explained, I will make my point on how I came to this conclusion.

Knowing that our angle of occlusion (player 2) is not letting the camera 1 determinate a real 3D tracking of the player 1, 2 more cameras were positioned on a orthogonally position to assure that player 2 was not the player 1 but it was only an angle of occlusion. With the data process by the server and with the determination that player 2 is of no use to us, player 1 is visible for our software.

It is also important to know that every camera has installed the library of OpenPose 2D to determinate if what the focus of the camera is looking is a human or not, and after that reconstruction whether the 3D reconstruction matches a human body or not.

3. Development of research work

First of all, our project focus into the creation of an AI system capable of detect precisely offside following the rules of FIFA. Having the objective clear allows us to split our problem in four parts. These four parts will be the guide to our work and will clearly determine the steps to make it. So, we are going to explain every little problem and to sum up, we are going to gather all together to get a solution to the problem.

3.1. Players´ Recognition

We evaluated two widely utilized detectors for player recognition; the need to test both stems from the fact that each has its own set of advantages and limitations.

YOLOv5 (You Only Look Once) and SSD (Single Shot Multibox Detector) are the main detectors, both of which use CNN and deep learning, the optimization of this set of neurons is done through the gradient descent.

Gradient descent reduces the objective function $f(\theta)$ to the smallest value possible by updating θ in the opposite direction of the gradient $f(\theta)$[6]. The algorithm takes steps towards a local minimum of the objective function by iteratively updating in the opposite gradient direction. The gradient descent updating rule is $\theta_{t+1} = \theta_t - \eta\nabla f(\theta t)$, [7] where η (the learning rate) determines the step size with which the algorithm moves towards a local minimum.

As a result, a filter that recognizes patterns to detect the object is created.

YOLO differs from SSD in that it separates the image into an array and only analyses each section of the array once, reducing duplicate data. SSD, on the other hand, separates the image into pixels, which then produce numerous anchor boxes [8] that it investigates separately and redundantly. The accuracy is higher, but the pace is slower.

Intersection over Union is the metric by which we evaluate the measurement accuracy of an image detector on a data set; as shown in the figure, a 0 indicates that there is no overlap between the boxes, while a 1 indicates that the union of the boxes is the same box as there is complete overlap, indicating that they are completely overlapping.

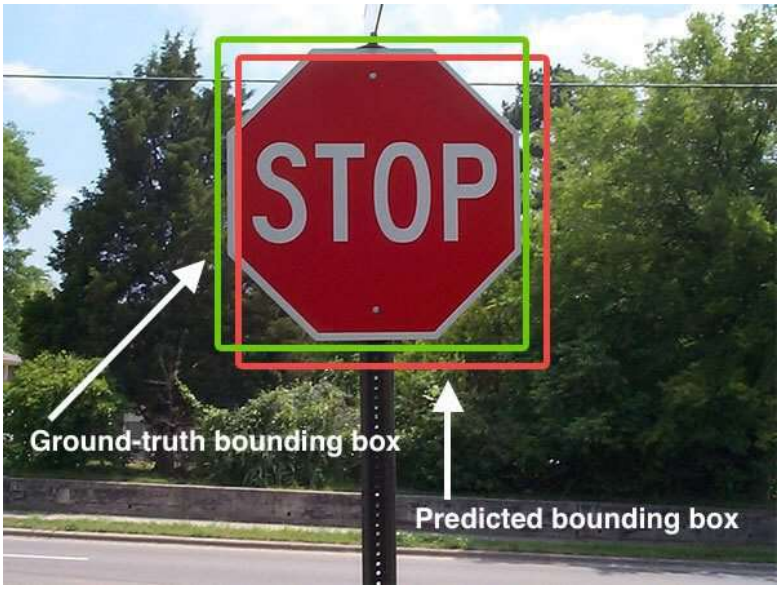


Figure 9: An example of detecting a stop sign in an image. The predicted bounding box is drawn in red while the ground-truth bounding box is drawn in green. Our goal is to compute the Intersection over Union between these bounding boxes.

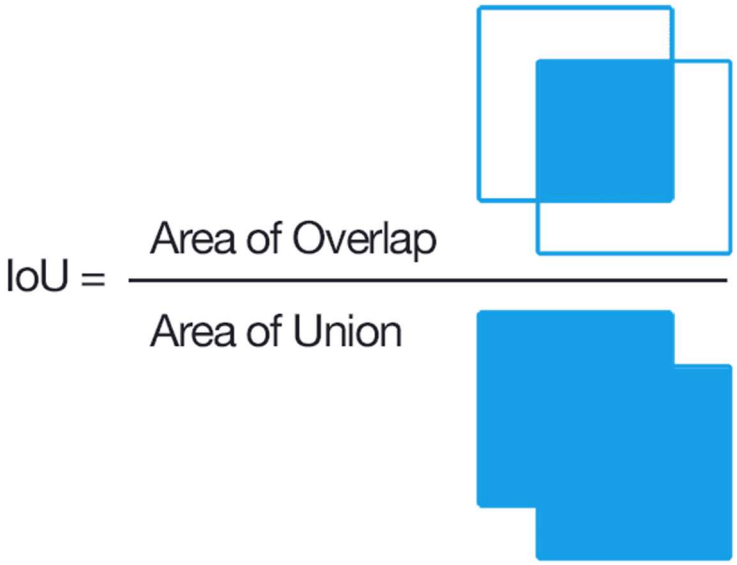


Figure 10: Computing the Intersection over Union is as simple as dividing the area of overlap between the bounding boxes by the area of union

We want to get an IoU of 1 as quickly as possible with a low error rate, and we have the advantage that, unlike a normal image, we only need to indicate a ball, the players, and their delimitation, so that the network's training is efficient in the search for patterns to recognize is faster, bounded, and more accurate. With SSD, we cannot attain this speed because we cannot analyze in real time as fast as we want.

The dataset utilized for network learning will be the one used in the F.I.G.C.- C.N.R. Project's soccer data collecting campaign, which yielded a detection result as shown in the figure.

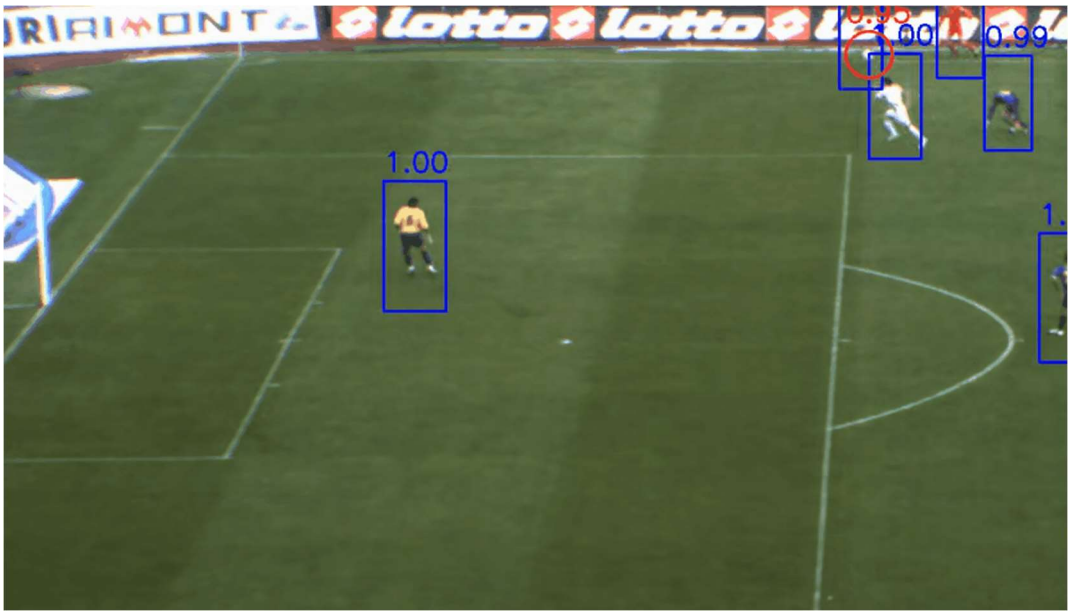


Figure 11: GIF of Detection with the Percentage of Success

The network's biggest challenge is differentiating between two or more players when they're at an occlusion angle and the system does not know whether there's more than one. This is solved by using skeletal tracking, which will be detailed in more depth later.

Another challenge is to distinguish the ball well enough because it is such a little object that it might be confused with the players' boots as well as the field's grass; nevertheless, there is a solution for this. We can use feature pyramid network design pattern which efficiently combines higher resolution, low level feature maps with lower resolution maps encoding higher level features and having larger receptive field, this gives us more accuracy and speed just replacing the feature extractor of common detectors and generates multiple feature map layers with better quality information.

3.1.1. Development

After looking for studies or existing algorithms and technologies with goals similar to ours, we've come to the conclusion that building a neural network is the most effective way to achieve success. The goal of neural networks is to mimic the functioning of human learning by using layers made up of nodes called artificial neurons that carry signals from one layer to the next.

Other detectors, such as SSD, are more accurate, but they are slower, and thus do not meet our use case.

3.1.2. Data

First, we must find appropriate data to train the neural network; in our case, we require greater accuracy and speed, and since we will only be using two types of objects (players and balls), we will create our own dataset to teach the network, avoiding the network's use of Coco, which expands and inefficiently expands the database (reducing false positives and errors by detecting things that were not easily detected), and adapting it to our needs.

To construct our own dataset, we will use OID [7] and photos obtained from real matches and databases to obtain football images, which is what we require. We will then label it using the LabelImg program to label the players and ball so that our network learns what those two labels mean.

To complete the process of obtaining the database of soccer photographs, we develop a roboflow project that allows us to apply noise, blur, rotation, and other effects. In addition to providing us with folders containing training data in a percentage determined by us (70,20,10), with photographs chosen at random.

3.1.3. Training

The training of a neural network can be divided into three distinct steps. To begin, the input data must be passed through the layers that make up the neural network. Second, the network's output must be assessed. Because our model belongs to the supervised learning algorithms category, we must calculate the deviation of the created outcome from the label or predicted output linked with the data we fed into the network. Finally, the weights of the network are changed according to their contribution to the total error of the network in the backpropagation process.

Several parameters will be used to assess error and efficiency:

- Precision: Is computed as and represents the fraction of correct detections.

$$True\ Positive/(True\ Positive + False\ Positive).$$

- Recall: What percentage of the objects I´ve catched are really a labeled object.

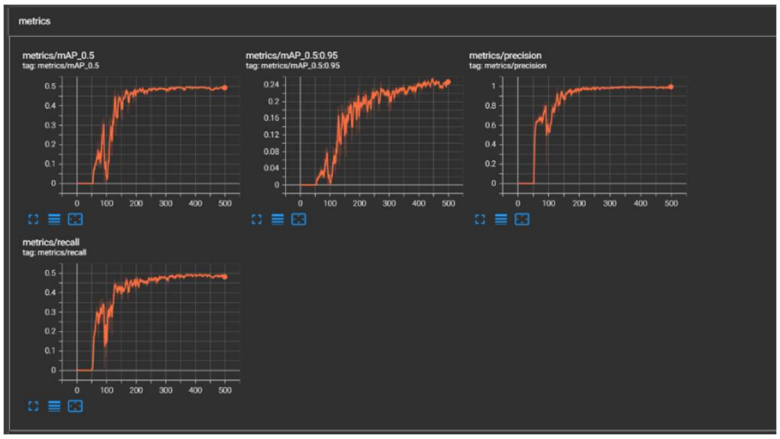
$$True\ Positive\ /\ (True\ Positive\ +\ False\ Negative).$$

- Loss: A metric for assessing how well a model matches the data.

We'll start with a pre-trained Yolo model, especifically Yolov5s, which generates a smaller but quicker network that meets our real-time detection use case.

During the tests, we noticed discrepancies in performance. First, we utilized YoloV5 trained with COCO [8](Common Objects in Context) and got lower results than predicted because it had issues with object confusion due to the vast dataset (baseball glove instead of Assistant Umpire's flag, for example).

Then we train it with our own dataset, and it improves learning, but it is still inefficient for the application we require. We first tried with Yolov5s, but the results were not better, so we used the Yolov5L network, which builds a larger and more complicated network with more accuracy than Yolov5s. It was better but we weren't satisfied with our results.



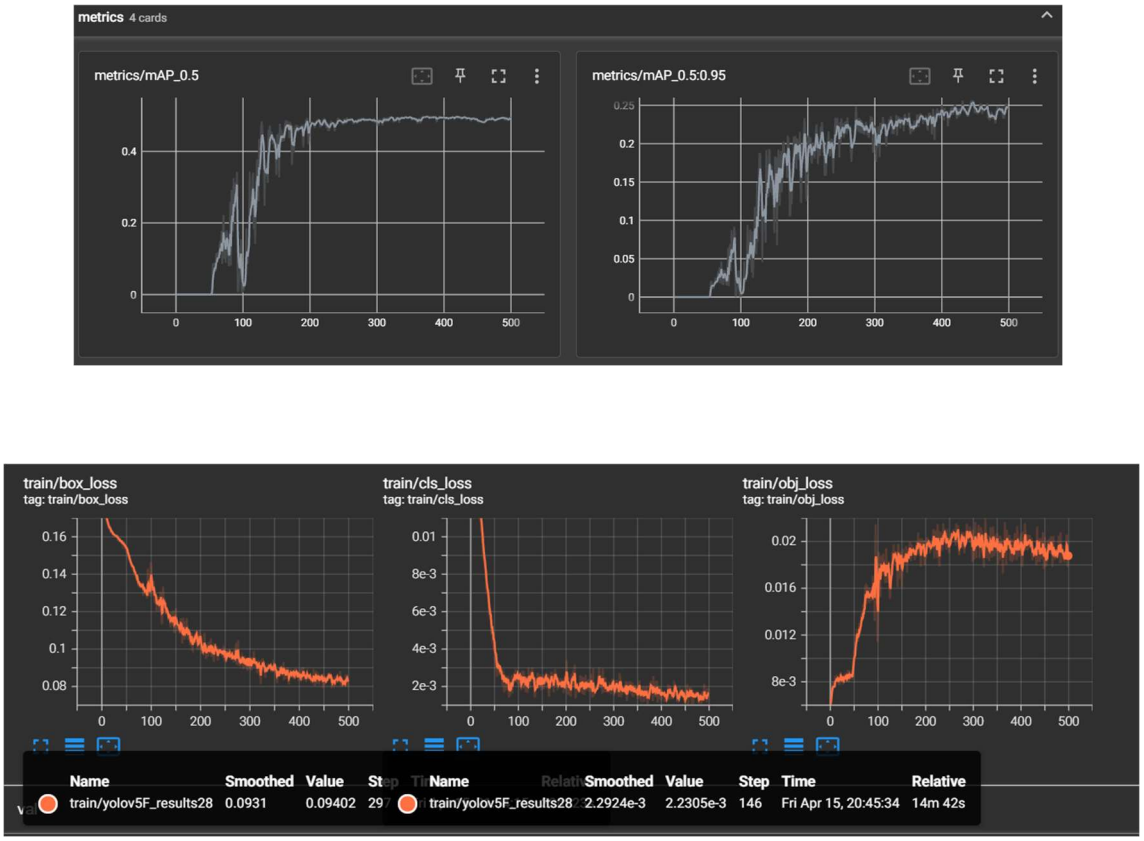


Figure 12: Results of our Trained Network

We discovered a project called FootAndBall [9] that updated this network and produced significantly more accurate and faster results once we realized we weren't obtaining the required performance and that the best thing to do was to modify our network.

We utilized the same videos for all of the tests so that we could examine the efficiency of each one quantitatively and tweak it to our needs.

We still have issues with occlusion angles that could be solved by using multiple cameras to obtain different angles and that the network can detect everything effectively and does not lose the ball or a player due to being covered, which would be a problem in order to detect out of games in a completely reliable way. This will be discussed in future work.

We are unable to conduct this test because many angles of the same match are hard to obtain. Furthermore, we can utilize Yolo to obtain each player's coordinates in order to move the landscape to a 2D plane, which will be easier to analyze later.

3.2. Mapping Players´ Position

Once we have all these problems solved, we are capable of recognize every object in the field such as the ball, the referee and, obviously, the players. Moreover, we apply the suitable filter to recognize the team of every player and finally, to reach the maximum precision we track the movements and the human poses.

Now, the next step is getting all this information in a map to show the results and make an easy way to calculate offside. For this purpose, we search for a method to convert the players´ position in real time, from our video source to the map.

This method is called **homography** and his function is to make a geometrical transformation between two planes. In other words, it is a mapping between two planar projections of an image. Mathematically, the planar homography is represented by a 3x3 transformation matrix represented as:

$$s\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Figure 13: Matrix of Homography

In our case, we want to make a planar transformation between a 3D frame like a football field to a 2D plane, in particular a birds-eye plane. We are going to demonstrate what is the result of this method:

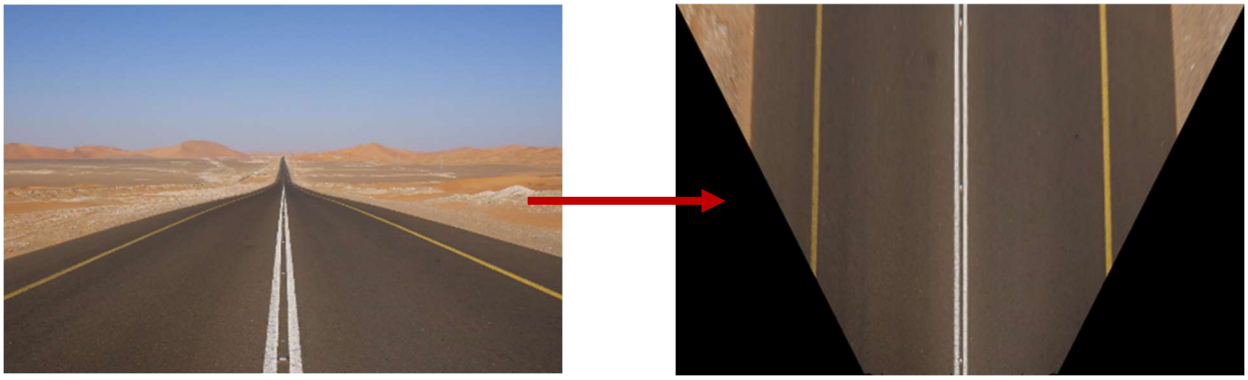


Figure 14: Explanation of Homography Process

As you can see in these images, we have a long road with perspective of the mountains, but, applying the homography function gives the result of the road as you can see if you are a bird or maybe like the first car videogames that the perspective was on 2D.

But now, the question is, how we can use this method for our purpose, we need to convert our video streaming data of the field into a 2D map. For this problem, we use a OpenCV's getPerspectiveTransform() function [10]. This

function calculates a perspective transformation from four points that must be matched with four points on the 2D map.

But to deal with this type of data (coordinates) we need NumPy [11], that is a Python library specialized in numerical computation and allows representing collections of data of the same type in various dimensions, and very efficient functions for their manipulation.

Specifically, we need an array with four points for each map. An example of the special array structure could be:

Array = np.array([101, 185], [393, 151], [479, 323], [187,441])

We will have to calibrate the fixed camera with the four points of the 2D map. A graphic example of what we must do:

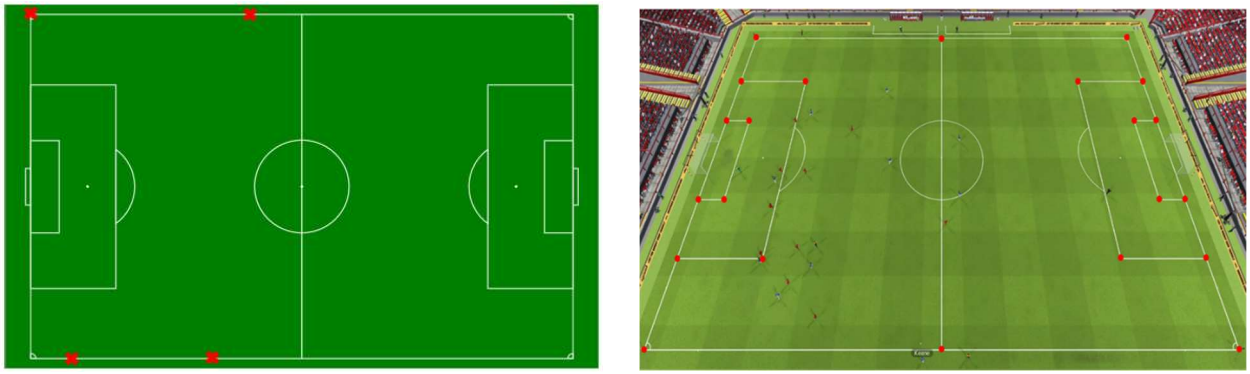


Figure 15: Transformation process of a Field

The implementation of this process gives us the following result:

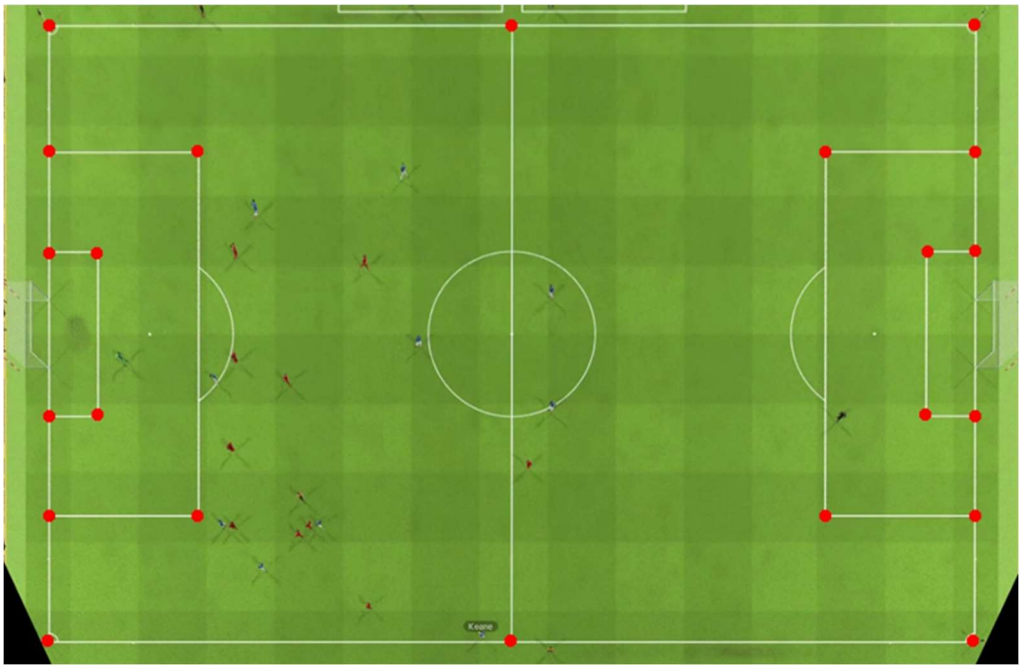


Figure 16: Homography Process

3.3. Team Recognition and Classification

Now, we can consider the capability to detect with a great percentage of success what is a person and what is a ball. Also, we have prepared the map doing a homography process to have all the coordinates of the field in a global template to put the coordinates of the player into it.

Even though, we could have the positions of the players into the map, we cannot differentiate, for example, the team to which the player belongs, and this is very important to fulfill the goal of this project, that is creating an algorithm to detect offside in a play.

Two approaches that our team proposed was:

1- Train another object detection model (or improve a lot the detection model that we are already using) to differentiate who is from these team and from the opposite team, and who is the referee. But we ruled out this idea because every game is different in terms of the jerseys of any team and the referee jersey, so, this object detection model would not be effective for any game no matter who is playing.

2- Use our current object detection model and extract the color applying some transformations from the detections. This method will be easy to integrate with all the technologies that we are using.

So, in the moment we compare these two possible solutions we concluded that the best option was the second one. We investigated how we could make a color filter in Python, as it is our main programming code.

This investigation leads us another time to the OpenCV library, where we can find a way to make color masks in order to get the t-shirt color from every single player and classify all of them into his home team. To introduce the color masks we need, first, to understand what a color space is.

A color space is a specific organization of colors. Combined with the color profiles supported by different physical devices, it supports reproducible color representations - whether such representations include analog or digital representations, the color space must be arbitrary [14].

By default, the color space used in OpenCV is BGR (Blue, Green and Red), which is odd for us, but, in the past, most of the digital devices worked with cameras using BGR, but, in the last 2 decades the RGB was implemented as the new king of color spaces.

We are going to explain how we apply the color masks (with the default color space) into our frames, and, later, what is the result and the conclusions about this phase.

1. First, we need to define the lower and upper threshold of the color that we are going to filter. In our case we are going to filter blue color and red color because of the t-shit color teams.
2. Later, we are going to implement *inRange()* function with the photo to filter, and the mentioned thresholds.

So, these are the results of color filtering:

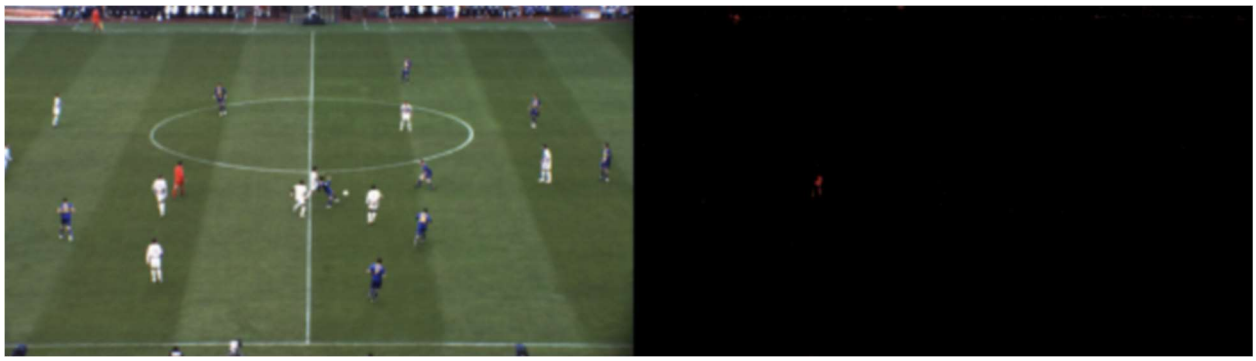


Figure 17: Color Mask to Identify the Referee

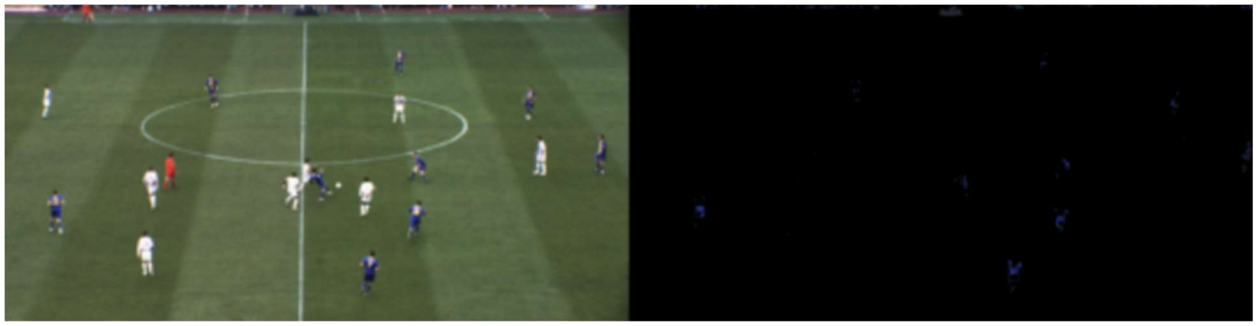


Figure 18: Color Mask to Identify the Blue Team

As we can see, the results are satisfactory but, when we were stepping into future phases, we discovered that the filter is not as powerful as we need to finish our goal. So, we went back to the first stage again.

After analyzing the results obtained, we started a new investigation, trying to change BGR color space into another color space with which we obtain a bigger color mask. We find out that HSV is the easiest solution to get the perfect color mask for our objective.

HSV (for Hue, Saturation, Value) is an alternative representation of the RGB color model, developed by computer graphics researchers in the 1970s to obtain a better match of how the human eye perceives color properties. In these models, the colors of each hue are arranged in radial disks around a central axis of neutral colors, from black at the bottom to white at the top.

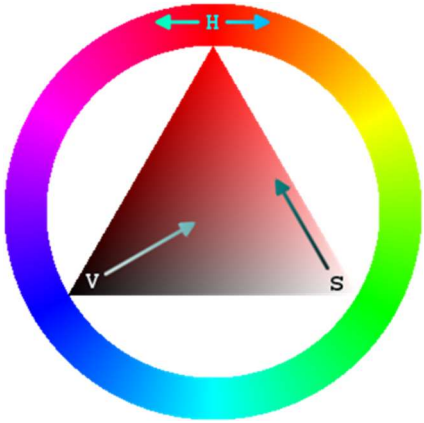


Figure 19: HSV Color Space

HSV represents a way of simulating the mix of different colors to create colors in the real world, with a lightness dimension like that of varying amounts of black or white in the mix (for example, to make "light red" also produce a red pigment white; this white corresponds to high "brightness" values in the HSL representation). Fully saturated colors are placed around a circle with a lightness value of ½, where a lightness value of 0 or 1 corresponds to full black and full white, respectively.

In addition to the new color space, we also implemented two color masks, instead of one, trying to get as much color as we can from the input. The new method has been carried out as follows:

1. We also used the *inRange()* function, but in this case, we used twice the same so we need to define 4 thresholds (two for the lower layer and two for the upper layer).
2. When we get the two *inRange()* function, we need to fuse both of them, reaching the goal that we get, that is increase the color filter. In order to reach this, we use *bitwise_or()* function.
3. Finally, we apply the new masks and obtained the following results.

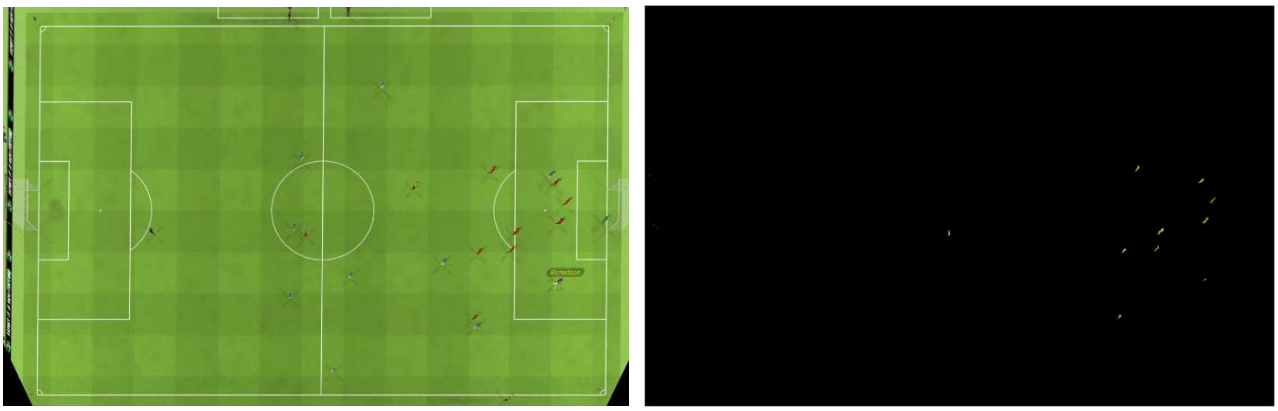


Figure 20: HSV Color Filter (Red Team)

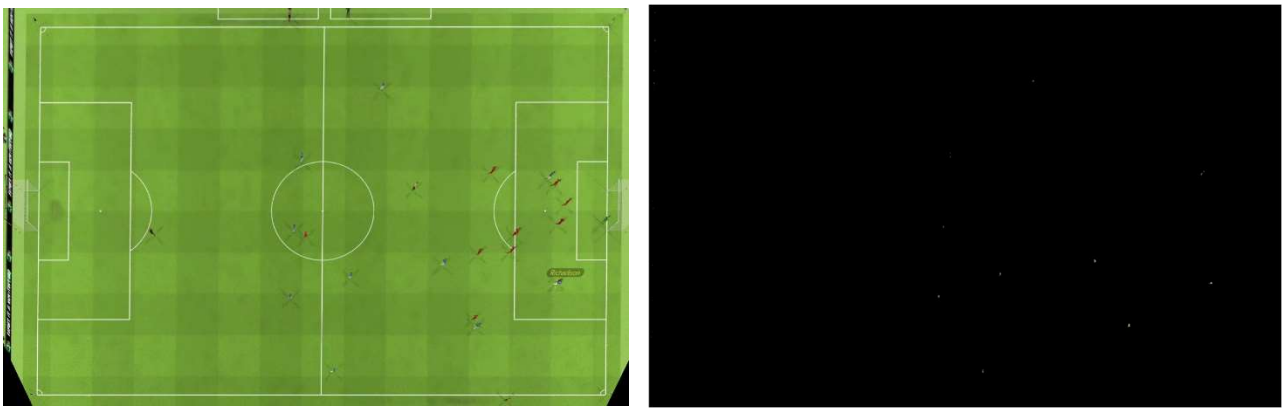


Figure 21: HSV Color Filter (Blue Team)

Now we can confirm that these are the results that we wanted, and we can continue to the next phase, that is the players mapping.

3.4. Players´ Mapping

Once we have located the players, the referee, the soccer ball, and managed to differentiate the players of the different teams by color using the previously explained color mask. In addition, as a previous step we have represented the videos of the game in a zenithally plane through the homography process which was also explained previously.

Once all the previous work explained has been carried out, we are going to carry out the graphic representation of the players, referee, and ball in a 2D plane to facilitate the subsequent processing of the information and the decision-making on the validity of the plays.

In a first step, we generate the homography matrix for the videos denoted and their top-view, we also use the inverse homography matrix to edge the map, this method is applied equally to every video or frame we use.

In the second step, we apply color masks in order to classify every player into his home team, and this process will be incredibly useful for this phase.

Like in the last phase, we had a brainstorming and get two approaches that our team thought could fit into our possible solution:

1- The bounding box in YOLO has the possibility to get the center of the rectangle, so, we can draw a point in this center, and get the coordinates (in a specific format that is measured from zero to one).

2- Search a method to reuse the color masks in order to get the contour of the player, detect his torso and draw a circle in the center of it, and get the coordinates (in pixels).

In conclusion, we choose the second option because the YOLO bounding box being processed by the homography process gets blurred and distorted, so the center of the rectangle leads to failure. So, we start the investigation with a way to get the contour of an object (or various objects).

Like the latest phases of our project, our investigation concludes in the implementation of OpenCV. This library holds a powerful method to achieve the new goal, OpenCV contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition [15].

This example shows the power of the OpenCV contour and is a great start to understand how we pretend to do our objective.

Figure 23: Contour of an iPhone and other gadgets

As we can see in the last image, OpenCV draws a line in whatever color we want around the object defining a contour of the object. So, we can confirm that is the greatest solution for us.

Now, before we start applying the OpenCV functions, the library itself in his guidebook recommends to the user to apply first some image processing to achieve the greatest result at the time to get the contour. OpenCV recommends applying a threshold or a canny edge detection.

In our implementation we decided to use both of the options, because we must get the contour no matter what the conditions of the image are, imagine if our program losses the players positions in the exact frame when an offside is being committed. So, here are the steps of our OpenCV contour implementation:

1. First, we need to get the color masks, and, why not the source image, this is because we want only to get the torso of all the players in his home team and not some other things like the public watching the game or other objects that we don't need to care of. Moreover, this helps us in the moment of classify the team, we can draw the circle with the same color, and put the color of the team in that exact time.
2. As we commented in the previous point, we use the masked image, so the color space that it has is HSV, and as we mentioned in the last phase, OpenCV works with BGR color space so we need to apply the transformation with *cvtColor(_, cv.COLOR_HSV2BGR)*.
3. Now we have the image in a perfect way to work with it. Now, we start with the transformations that the guidebook recommends us. First of all we need to put the image in a grayscale with the function *cvtColor(_, cv.COLOR_BGR2GRAY).*
4. Furthermore, we apply both of the methods mentioned later and one more to smooth the result. The first one to apply is the threshold, with a personalized thresh that we test to get the best result. Secondly, we apply a *GaussianBlur* to smooth the image to avoid a very sharp image, and to finish we make the Canny Edge transformation getting this image:

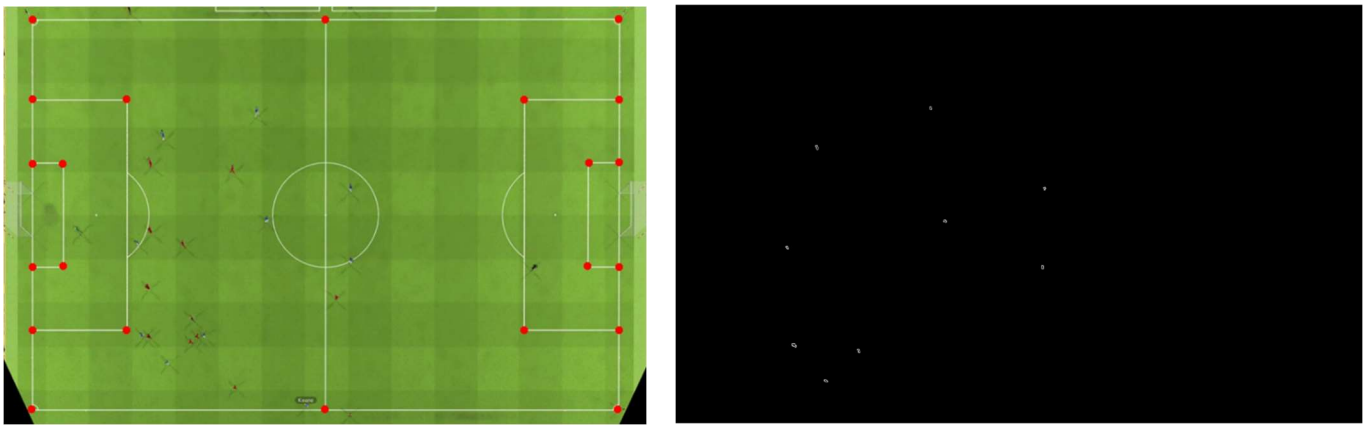


Figure 24: Contour of Blue Team Players

5. In this final step, once we have the contour of every single player we use the function *minEnclosingCircle()* to get the center of the contour (this function returns coordinates x and y of the center) and now we just need to store all this data in a array for future cases and draw the circle.

When all these processes are done, the result that we got is the following (from the start of the input to the last result, the minimap):

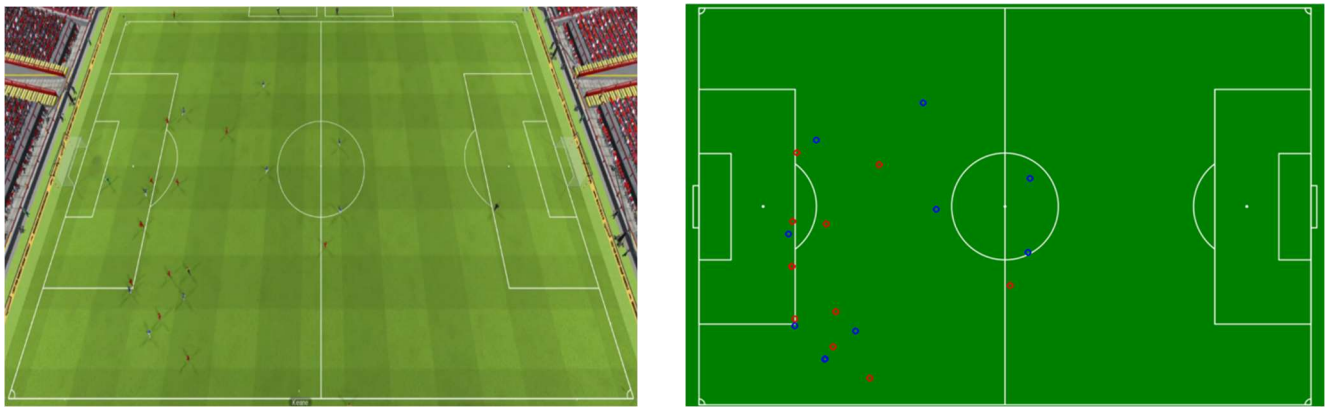


Figure 25: 2D Minimap of our Proyect

3.5. Offside Detection

For this part of the project our goal is to set an algorithm which can identify the position of the players among the pitch and determinate with a set of parameters whether the player is in a correct or incorrect position.

As it was seen in the previous steps, homography and color mask is a key to the matter as well as YOLOv5. With the help of all these 3 procedures,the offside algorithm is just basic coding a multi IF sentence where we compare the positions of the two players (one of defender team and another one of the attacker team) and see if the attacker is ahead of the defender.

During all the execution of the algorithm we must check frame by frame the position of the defender closest to the defending goal and all the players of the attacking team, and, with this real-time comparison we can check every single frame and reach the best precision.

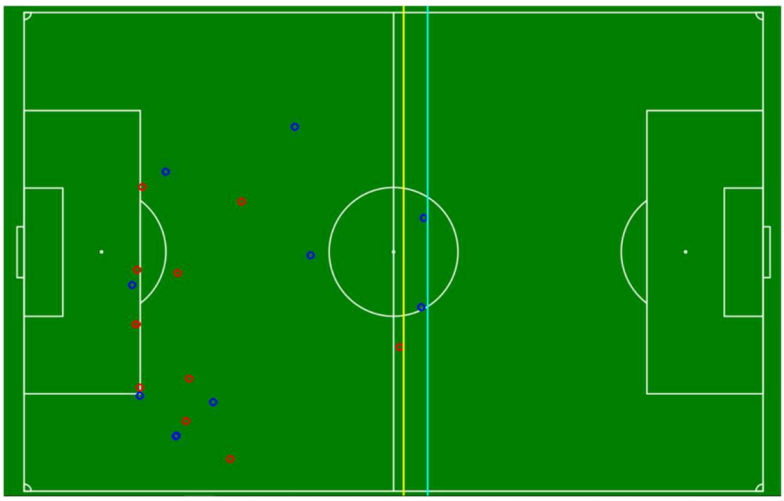


Figure 26: 2D map with the two offside lines.

The previous figure shows a graphic example of how our system takes the position of the last defending player in the defending field and draws a blue line. The system does exactly the same but drawing a yellow line for the last attacker. This helps the customer to see in a graphical way how the play is being executed, and, furthermore, we can generate a graphic for the possible video streaming of the soccer match.

Checking the position, at this point there can be two possible cases (the base of our algorithm):

- Attacker Coordinates > Defender Coordinates: Attacker is in offside position.

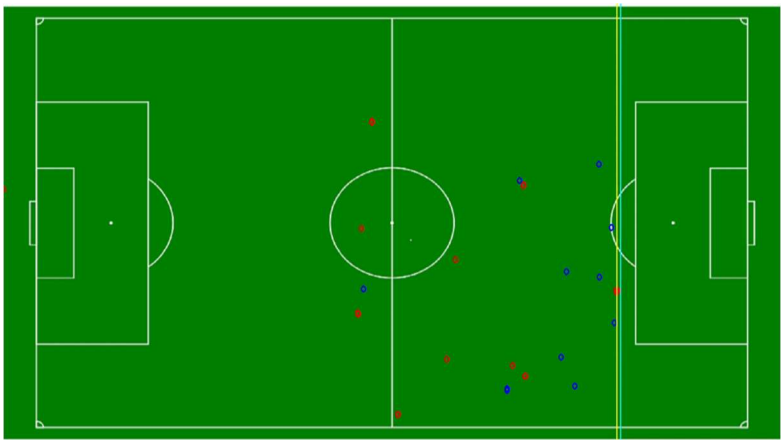


Figure 27: Example of Offside Play.

- Defender Coordinates => Attacker Coordinates = Attacker is in a correct position.

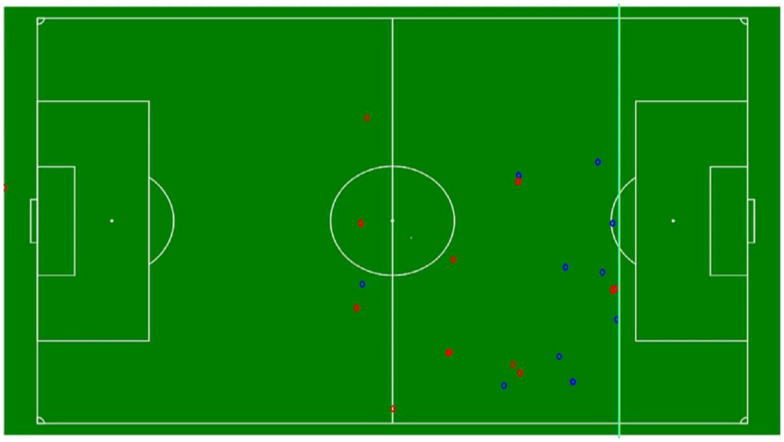


Figure 28: Example of Correct Position.

As we mentioned in previous phases, the information of each frame is saved in a JSON file where we represent some of the most important parameters such as if the frame is in offside or not, the team color t-shirt and many others that we are going to see in the next figure:

```
"Frame": 28,
"Equipo A": "Camiseta Roja",
"Equipo B": "Camiseta Azul",
"Coordenada X - Camiseta Roja (en pixeles)": 1476,
"Coordenada X - Camiseta Azul (en pixeles)": 1475,
"Fuera de Juego": "Si"
```

```
"Frame": 27,
"Equipo A": "Camiseta Roja",
"Equipo B": "Camiseta Azul",
"Coordenada X - Camiseta Roja (en pixeles)": "1474",
"Coordenada X - Camiseta Azul (en pixeles)": "1476",
"Fuera de Juego": "No"
```


Figure 29: JSON File of the Execution of our Project

When we have implemented and optimized the offside algorithm, it is time to test how operative it is with several videos. For them we have taken several videos from the Football Manager 2021 simulator, which allowed us to take realistic game images perspectives and change the color of shirts for both teams very easily. The decision of using the images of Football Manager 2021 was because of the lack of video of soccer games due to copyright laws around the world.

We have tried with 25 different matches, and we have managed to get the following data:

	Accuracy	**Effectiveness**
Offside	0.81	0.86
No Offside	0.92	0.89

Figure 30: Table of the results of our algorithm with examples

As we mentioned in the team recognition and classification part, we started using BGR color space and we calculated the effectiveness of detecting the contour and the painted circle, and the data revealed a big number of failures detecting the points, so, based on that data we changed the color space to HSV, and we conclude with this new measures:

	Accuracy
RGB	0.69
HSV	0.97

Figure 31: Table comparing the Accuracy of BGR / HSV color space

To sum up all the workflow from our offside algorithm we created a state diagram to show how we managed to create the steps to determine what is an offside or not.

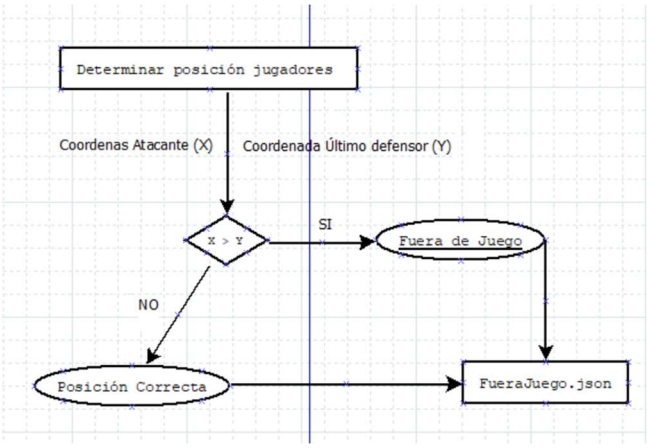


Figure 32: State Diagram of the Offside Algorithm

4. Conclusions and Future Work

Our team is amused and satisfied with the final results of our project, our first approach was to deliver a program with the capability of measuring offside plays, detecting them and informing the supervisors of the match. At the next milestone when we were making the state of the art, we were very ambitious, and we thought about a full program that could measure other fouls like touching the ball with the hand or physical fouls. However, we had limited time and resources, so we did a smart movement, which was to focus on the offside detection and try to make it simple but with a big accuracy.

To conclude and sum up all the ideas that we exposed during all this document, we created a complete workflow of our project or program.

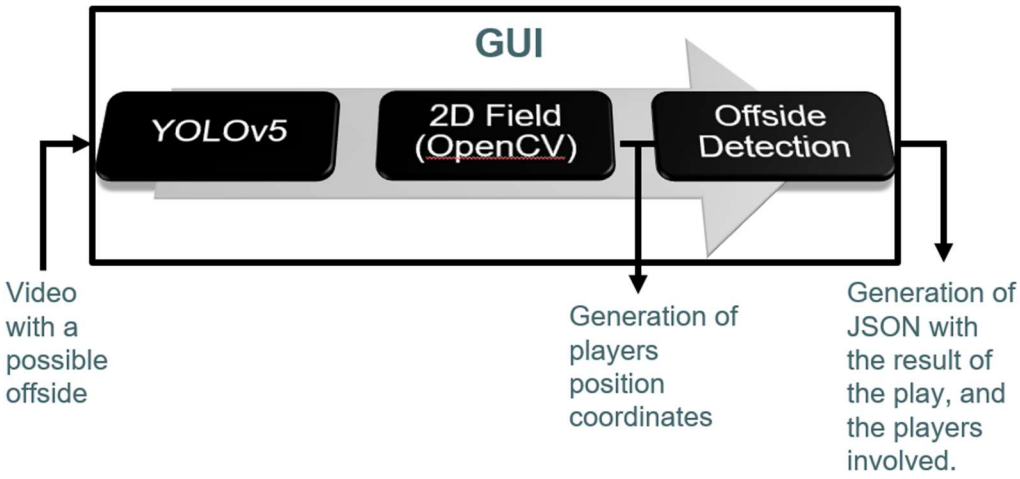


Figure 33: Workflow of our Robot Referee

Now, talking about present, with the project done, we can conclude that we managed to find the solution to map players' position and create an algorithm to observe if the player is in offside or not.

On the other hand, our way to measure the players´ position does not follow exactly the rules of FIFA in modern soccer, the rule tells that the most protruding part of the body that needs to be measured is the shoulder, and our program uses the torso of the player to measure that. So, one of the biggest lines of future work is the implementation of projects like "Skeletal Tracking" or "AlphaPose" that can follow the movements of body parts.

Nevertheless, our work is very scalable, thanks to the creation of a JSON model with all the coordinates of the players, and other types of data that we mentioned before, so, another line of future work is creating more situations to measure, or even create a big data solution for soccer, to measure lots of statistics which in our case we are losing the opportunity to manage them and it's a shame to lose them.

To sum up, we would like to create a proper GUI to use our algorithm, because we consider that the use of our project is very old-fashioned, or even incomplete, so this is the last line of future work to mention.

5. References

[1] Los deportes más practicados: el desfase entre el deporte federado y el deporte popular y recreativo. (s. f.). CSD - Consejo Superior de Deportes. Recuperado 17 de marzo de 2022, de https://www.csd.gob.es/es/los-deportes-mas-practicados-el-desfase-entre-el-deporte-federado-y-el-deporte-popular-y-recreativo

[2] La industria del fútbol profesional genera 185.000 empleos, 4.100 M€ en impuestos y una facturación equivalente al 1,37% del PIB en España. (s. f.). Fútbol Global. Recuperado 17 de marzo de 2022, de https://newsletter.laliga.es/futbol-global/la-industria-del-futbol-profesional-genera-185-000-empleos-4-100-me-en-impuestos-y-una-facturacion-equivalente-al-137-del-pib-en-espana-1

[3] Procesamiento y accesibilidad de datos LiDAR a través de aplicaciones distribuidas. (s. f.). Procesamiento y accesibilidad de datos LiDAR a través de aplicaciones distribuidas. Recuperado 17 de marzo de 2022, de https://core.ac.uk/download/pdf/148657708.pdf

[4] Qué es y qué podemos hacer con GRASS GIS. (s. f.). MappingGIS. Recuperado 17 de marzo de 2022, de https://mappinggis.com/2016/05/puedo-grass-gis-7/

[5] Convolutional Neural Networks, Explained - Towards Data Science. (s. f.). Medium. Recuperado 17 de marzo de 2022, de https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

[6] Zhao, K., Di, S., Li, S., Liang, X., Zhai, Y., Chen, J., Ouyang, K., Cappello, F. and Chen, Z. (n.d.). FT-CNN: Algorithm-Based Fault Tolerance for Convolutional Neural Networks. [online] Available at: https://arxiv.org/pdf/2003.12203.pdf [Accessed 26 Mar. 2022].

[7] M. (s. f.-a). GitHub - Medium/oid: Utilities for object identity and hashing. GitHub. Recuperado 26 de marzo de 2022, de https://github.com/Medium/oid

[8] Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. and Dolí, P. (n.d.). Microsoft COCO: Common Objects in Context. [online] Available at: https://arxiv.org/pdf/1405.0312.pdf.

[9] jac99 (2022). jac99/FootAndBall. [online] GitHub. Available at: https://github.com/jac99/FootAndBall.

[9] Sebastian Ruder. (2016). An overview of gradient descent optimization algorithms. [online] Available at: https://ruder.io/optimizing-gradient-descent/#:~:text=Gradient%20descent%20is%20a%20way [Accessed 26 Mar. 2022].

[10] Varadarajan, V., Garg, D. and Kotecha, K. (2021). An Efficient Deep Convolutional Neural Network Approach for Object Detection and Recognition Using a Multi-Scale Anchor Box in Real-Time. Future Internet, 13(12), p.307.

[11] Manh-Hung Nguyen, Ching-Chun Hsiao, Wen-Huang Cheng, · Ching-Chun Huang (2021). Practical 3D human skeleton tracking based on multi-view and multi-Kinect fusion.

[12] P. Raguraman, A. Meghana, Y. Navya, Sk. Karishma, S. Iswarya (2021). Color Detection of RGB Images Using Python and OpenCv

[13] Hawk Eye · Automatic Birds Eye View Registration of Sports Videos. (s. f.). Git Hub. Recuperado 1 de mayo de 2022, de https://nihal111.github.io/hawk_eye/

[14] Wikipedia contributors. (2022, 19 febrero). Color space. Wikipedia. Recuperado 4 de mayo de 2022, de https://en.wikipedia.org/wiki/Color_space

[15] OpenCV: Contours : Getting Started. (s. f.). OpenCV. Recuperado 5 de mayo de 2022, de https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html

[16] Football players detection project using python and opencv (part6). (2020, 4 diciembre). YouTube. Recuperado 5 de mayo de 2022, de https://www.youtube.com/watch?v=fhJVTNbKfik&t=28s&ab_channel=TheWalkingProgrammers

[17] GitHub - saleh1312/football-players-matching. (s. f.). GitHub. Recuperado 5 de mayo de 2022, de https://github.com/saleh1312/football-players-matching

[18] K, C. (2021, 15 diciembre). Football players tracking — identifying players' team based on their jersey colors using OpenCV. Medium. Recuperado 5 de mayo de 2022, de https://towardsdatascience.com/football-players-tracking-identifying-players-team-based-on-their-jersey-colors-using-opencv-7eed1b8a1095

[19] Z. (2018, 29 junio). Football/Soccer on Your Tabletop – Zbigatron. Zbigatron. Recuperado 5 de mayo de 2022, de https://zbigatron.com/football-soccer-on-your-tabletop/

[20] S. (s. f.). Home · stephanj/basketballVideoAnalysis Wiki. GitHub. Recuperado 5 de mayo de 2022, de https://github.com/stephanj/basketballVideoAnalysis/wiki

[21] DELBOM, A. X. E. L. (2020). DETECTING OFFSIDE POSITION USING 3D RECONSTRUCTION. DETECTING OFFSIDE POSITION USING 3D RECONSTRUCTION. Recuperado 5 de mayo de 2022, de https://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=9030362&fileOId=9030364

[22] OpenCV: Creating Bounding boxes and circles for contours. (s. f.). OpenCV. Recuperado 5 de mayo de 2022, de https://docs.opencv.org/4.x/da/d0c/tutorial_bounding_rects_circles.html

[23] OpenCV: Drawing Functions in OpenCV. (s. f.). OpenCV. Recuperado 5 de mayo de 2022, de https://docs.opencv.org/4.x/dc/da5/tutorial_py_drawing_functions.html

[24] Using OpenCV with Tkinter. (s. f.). Tutorials Point. Recuperado 5 de mayo de 2022, de https://www.tutorialspoint.com/using-opencv-with-tkinter

[25] Clases - Tutorial de Python. (s. f.). Tutorial Recursos Python. Recuperado 5 de mayo de 2022, de https://tutorial.recursospython.com/clases/