Computer Vision

Project 2 Laboratory 5

Project Title:

Soccer Off-side Detection

Victor Miguel Velazquez Espitia - 2043179

June 2023

1 Introduction

Offside rule is fundamental in soccer. It governs the positioning of players during gameplay. It is enforced to ensure fair play and maintain the balance between attackers and defenders. According to the offside rule, an attacking player is considered offside if they are nearer to the opponent's goal line than both the ball and the last defender(not considering the goalkeeper) when the ball is played to them.

Offline detection refers to the determination of whether a player is positioned correctly relative to the mentioned rule. In this report, we present a comprehensive study on offline detection in soccer, utilizing computer vision techniques to automatically analyze given images and identify players in offline positions. Comparing results with different models and algorithms, of segmentation, and object detection. Finally presenting a robust pipeline that achieves the desired objective.

Figure 1. Example of offside

2 Objective:

The primary objective of this project is to develop a computer vision-based algorithm that can reliably detect when a player is in an offline position given an image of a soccer game. By leveraging advanced image analysis techniques, we aim to provide an accurate analysis into player positioning, helping to make informed decisions, on whether is an off-side or not.

3 Implementation and Methodology

Considering the best of the well known computer vision techniques, the objective was broken down in the following steps, that defines the **pipeline**:

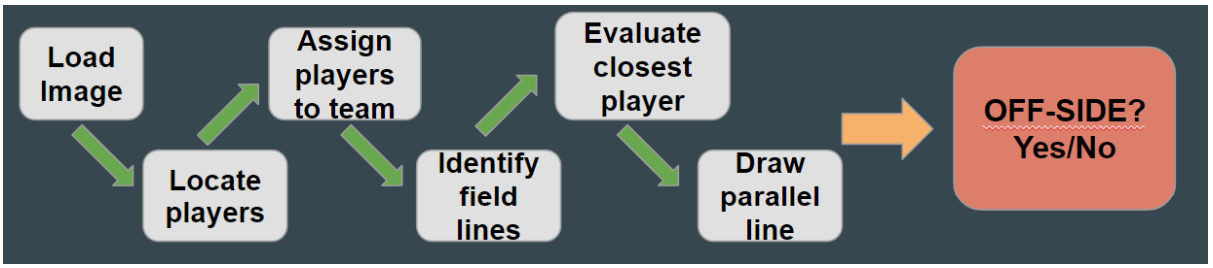

Figure 2. Implemented pipeline overview

In this section of the document each of the pipeline steps will be studied, presenting the different proposals and chosen solutions:

3.1 Load image

Trivial step is done using a function of the **OpenCV** library. But worth mentioning is that the project and analysis done was developed based on the OpenCV library. OpenCV is a powerful open-source computer vision and machine learning software library. Which provides a wide range of functions and algorithms for image and video analysis.

3.2 Locate players

As one of the crucial points, it is important to identify all the players in the scene. This was done through **object detection**, which involves analyzing images to identify regions of interest that potentially contain objects. This is achieved by combining techniques like feature extraction, classification, and spatial localization. The process typically involves using pre-trained models or deep learning algorithms to accurately detect and classify objects.

The chosen one approach was a **Convolutional Neural Network**(CNNs) approach. CNNs are usually composed of multiple layers, including convolutional layers, pooling layers, and fully connected layers.

In particular, implementing the socalled, **Faster R-CNN** (Region-based Convolutional Neural Network) a popular state of the art object detection algorithm. On the **Pytorch** framework, which provides a pretrained Faster R-CNN model.

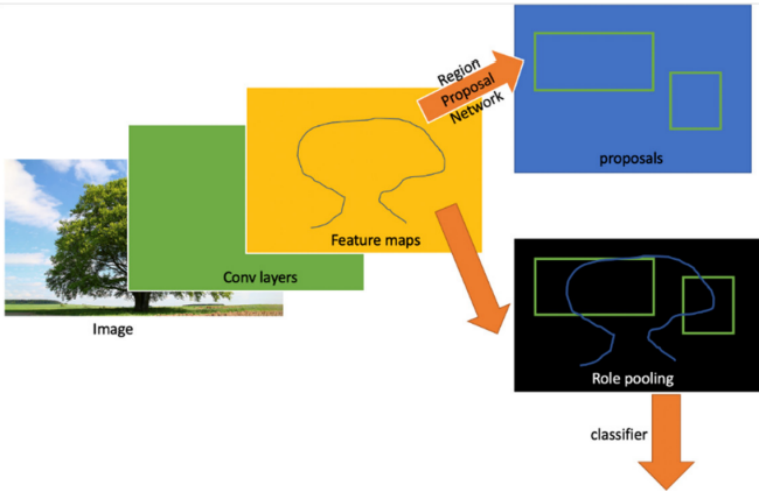

Figure 3. Faster R-CNN

To run Faster R-CNN was necessary to create an individual python script, where we will give the input image of the soccer game to pass through the model. To finally get the following output:

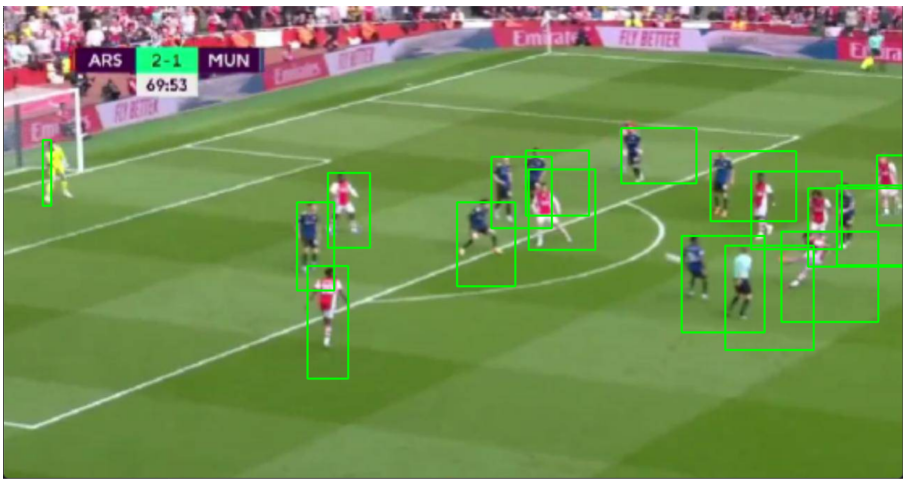

Figure 4. Bounding boxes raw output

To pass the values of the bounding boxes to the C++ script, a text file was generated where the values of top corner coordinates, height and width, were saved for each of the bounding boxes. File that must be read by the C++ main file, such that it can continue with the image processing.

3.3 Assign players to teams

Once the players have been identified, and with the help of the bounding boxes, the player's team was identified as follows:

1.- Reduce Boxes size to only capture players' jerseys. Because features such as head, skin color, and amount of grass in the bounding box, affect the average of pixels value per bounding box.

2.-Calculate the average of the RGB values in the bounding boxes, this based on the cv::mean(roiImage) function.

3.-Define an evaluation value (EV), calculated based on the three variables of the mean color. Which function is defined as follows:

$$EV = \sqrt{B^2 + G^2 + R^2}$$

This value gives an idea of how similar are the colors of the bounding boxes. Thus, allowing to generate ranges to differentiate between the teams t-shirt color.

4.-By iterating, manually generating a set of values, to differentiate between players t-shirt color. For the scope of this project, we limited the case to red, blue, and extra colors. Where extra represents any other color, expecting to be goal-keeper and referee. The ranges were defined as follow:

Table 1. Color Ranges for identification

Color	Blue	Red	Extra
Range	EV < 196	196 < EV < 256	EV > 256

Finally to have a clear result of the assignment of the players, an ID and the color where assigned, and displayed in the image. As we can see in figure

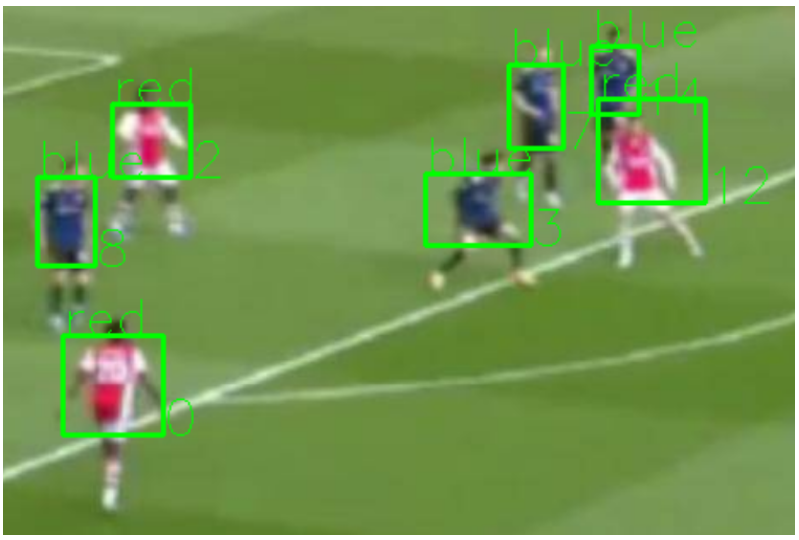

Figure 5. Reduced Boxes with player's id and team

3.4 Identify Field Lines

Another crucial step is that of identifying the field lines, since as the rule mentions. The offside is if the opponent player is nearer to the goal line than any other defender. Hence needed to identify the goal-line.
To this aim a Line detection function was design which performs the following steps:

- Convert Image to grayscale
- Apply Canny Edge Detection
- Detect lines using Hough Transform
- Filter Horizontal and Similar Lines
- Generate a track-Bar on houghlines() Threshold

Performing the mentioned steps and even with strict filtering and requirements on the lines detection. Identifying uniquely the Goal-Line is a complex task. Therefore, once a small amount of lines was output, the values that describe the Goal-Line were manually selected for future processing.

Figure 5. Line Detection results

3.5 Evaluate Closest player

For this step was implemented a function that iteratively obtained the perpendicular distance from the same point on the bounding box of each player to the Goal-Line. And then selecting the smallest one, without considering the goalkeeper.

3.6 Draw Parallel Line

In this step we simply draw a parallel line to the Goal-Line in the evaluated point of the bounding box of the identified Closest player.

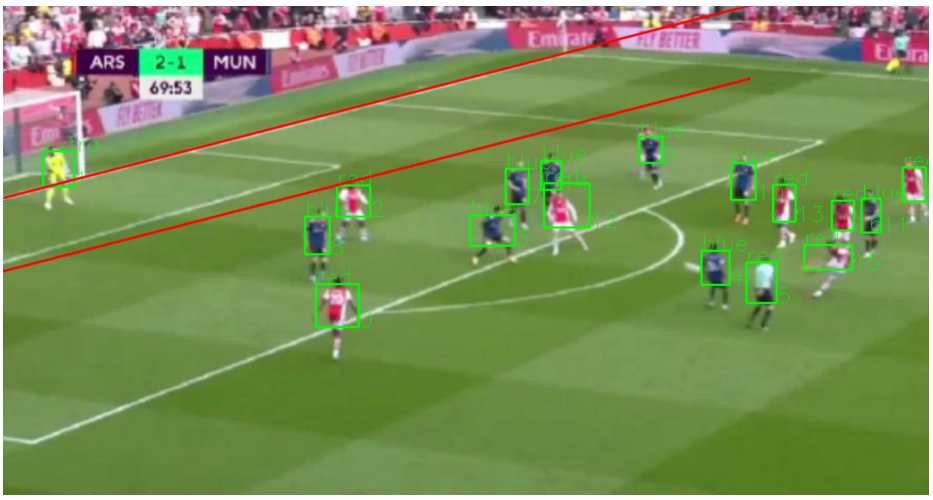

Figure 6. Off-Line Detection results

3.7 Determine Off-Side

Finally if the closest player belongs to the attacking team, there is an Off-Side. Otherwise, it is a legal maneuver.
An example of the final output can be seen in figure 7, for the given example there is an Off-Side!

```
-------------Start Soccer OffSide Detection Algorithm-------------
Processing Image...
.
.
Identifying players...
.
.
Making calculations...
.
.
Drawing OffSide Line...
.
.
...
Distance to closest player player: 98.2279
ID of closest player player: 2
Color of closest player: Red
--------------------------------
Is the attacking team red? (yes/no):
yes
OffSide detected! Ilegal Maneuver
-------------------End of program-------------------
------------------------------------------------------
```

Figure 7. Final output

4 Alternative attempts explored

On the endeavor of determining the best route for identification of the players and teams. Alternatives through segmentation algorithms were explored, such as:

1. Segmentation with **Ranges**: Which results were deficient. Since its not able to give a more specific segmentation of a higher variety of colors and shapes.

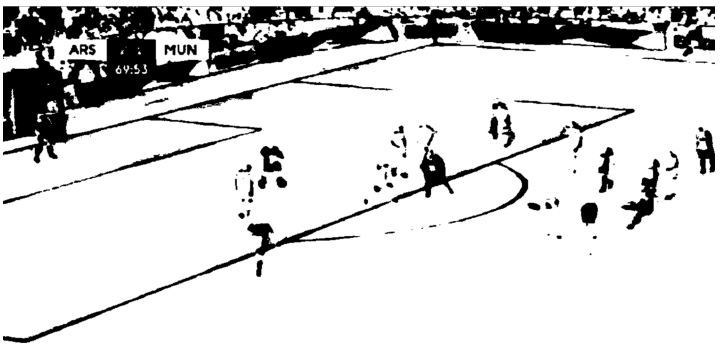

Figure 8. Ranges segmentation

2. Segmentation with **Otsu optimal threshold:** Which results were equivalent or worse to Ranges Segmentation, thus not worth presenting.
3. **K-Means** Segmentation: This approach presented the best results from the implemented segmentation algorithms. As we can see on the segmented image, the players are well identifiable, and we can differentiate between the red and blue players, bcs those gave different segments. A drawback is that the segmentation from goalkeeper and referee are not differentiable from the players. Limitation that was not overcomable even trying with different amounts of clusters, or high number of iterations. Hence making segmentation approach infeasible for the project's objective.

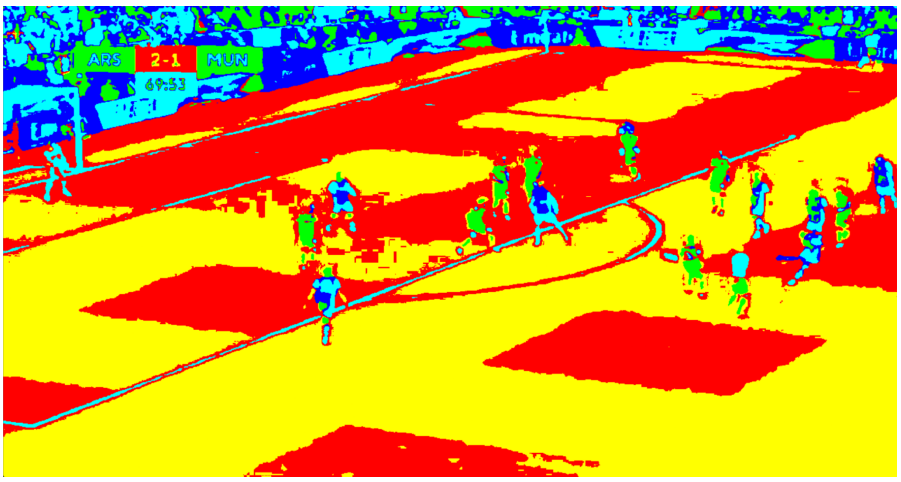

Figure 9. K-means segmentation

Following the same idea, and attempting to find the best object detection algorithm, **Haar Cascade classifier** was implemented**,** which is a machine learning-based object detection algorithm used to identify objects.

Figure 10. Haar Cascade results

The Haar Cascade classifier does not perform well in detecting all players in the images. Haar cascades have limitations in certain scenarios, especially when there are variations in pose, scale, or occlusions. And in this case players are presented in diverse poses and angles. Making the classifier very limited and non-addecuate. As is possible to see in the picture it just identifies correctly two players, using the person detector. Was explored the possibility of using Haar Cascade for classification of upper and lower body, but results were equivalent to identifying the whole body of a person. Due to these limited results, a more robust model such as R-CNN was chosen.

3 Conclusion

This report describes the project of soccer offline detection, which leverages computer vision techniques to automatically identify when players are in an offline position. By combining image processing, deep learning, geometric analysis, and contextual cues, we aim to provide an effective and reliable system for offline detection in soccer images. The project's outcome for the scope of this project was limited to the case of red and blue teams. Which represents a solid first approach to the solution of the general problem. The potential of future results is to be able to detect off-line in any possible scenario, with a more robust algorithm that prevents unexpected situations, different jersey conditions, among others to enhance the understanding and analysis of soccer plays. With the mentioned statements, it is concluded that the analysis of sport images via computer vision techniques giving specific rules has a great potential for the analysis of the game, dropping decisive conclusions, and avoiding the human error possibilities. Being this a wide field, it can also be used to study strategies as well as limitation of sports maneuvers.

3 Read Me file

The project is composed of 1 python folder and 3 cpp files, plus the image to process in this case "offside3"
-inside the python folder we find the torchdetection.py file which is the model used to find the bounding boxes, the image, and the bounding_boxes.txt, which is used to pass the bounding boxes to the offside_detection.cpp file to continue processing.

File functions.h contains the declaration of developed functions.
File functions.cpp contains the definition of the functions.

In these previous mentioned files we can find all the functions used to experiment with the images. They are leaved there for possible evaluation and evidence of the work.
In file offside_detection.cpp we find the main() function as well as the important functions used to arrived to the result.

The **main pipeline of functions** is the following:

torchdetection.py --> offside_detection.cpp --> Main() --> imread() --> read_boxes() --> draw_offside_lide() --> draw_goal_line() --> calculateDistance_toline() --> functions.cpp --> checkOffside()

All other functions were used for experimentation. In case user wants to try any other function just has to uncomment the function from the main()