Tracking Football Players: An Exploration of the Methods Used for Automatic Offside Detection

A report submitted to The University of Manchester
for the degree of Master of Engineering (with Honours)
in the Faculty of Science and Engineering

2023

Joseph Higgitt

Department of Computer Science

Contents

List of Tables

List of Figures

Abstract

Tracking Football Players: An Exploration of the Methods Used for Automatic Offside Detection
Joseph Higgitt
A report submitted to The University of Manchester
for the degree of Master of Engineering (with Honours), 2023

The aim of this report is to analyse the current methods used for tracking football players from a single-camera broadcast feed and propose a method that can accurately detect players in an offside position.

Offside calls in football are a contentious issue, with frequent controversies arising from incorrect decisions that can influence the outcome of matches. The limitations of the human eye is a main source for these error. Through a comprehensive review of existing literature and practical testing, this report will explore the potential for utilising technology to enhance the performance of an offside detector.

The model chosen uses image masking to identify players, before clustering teams together with DBSCAN. The Hough line transform is used to detect pitch lines, which are extrapolated to a vanishing point to give relative player positions. Angles drawn at this point give offside decisions.

This model was found to have the potential to become a robust detector, but due to a lack of training data and time the performance observed was disappointing. Therefore several improvements to the method proposed have been given.

Declaration

No portion of the work referred to in this report has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

I would like to thank my personal tutor, Dr Aphrodite Galata, for supporting me throughout this project - I'm sure it's been challenging!

I would like to sarcastically thank my dad for introducing me to football through our family club: the mighty West Bromwich Albion. He took me to my first-ever West Brom game in 2010.

1 Introduction

Dubbed 'the beautiful game' by many, association football is the most watched sport around the globe. A staggering total of 1.5 billion people tuned in to the 2022 FIFA World Cup final last December, a number over seven times that of Super Bowl LVI, held earlier that year [33]. Millions passionately support their local club week-in-week-out in the UK alone, leading to a huge contribution to the UK's GDP. During the 2016/17 season, £7.6 billion was generated for the UK economy [6].

Due to the money involved at the top of the football pyramid and the relentless desire of every club to win silverware during a campaign, the scrutiny placed upon officials has never been higher. It's become almost common practice to turn to the back pages of any newspaper following a weekend of action and be presented with a selection of controversial, high-profile decisions that have cost one team a result. The referee and their officiating team naturally face huge disadvantages when it comes to arriving at the correct decision: they can only see an incident from one angle in real-time, can have their view obscured by another player and face immense pressure from the tens of thousands of supporters present in the ground.

Footballing authorities are aware of this and have begun turning to increasingly advanced computer vision-centric techniques to aid the on-field officials. Goal-line technology was first used in the Premier League in the 2013/14 season, following a hugely controversial decision at the 2010 FIFA World Cup that saw Frank Lampard's would-be equaliser disallowed against Germany [26]. This system can automatically detect in seconds whether the ball has crossed the line, using fourteen high-speed cameras to achieve impressive precision [7].

Following on from the successful roll-out of goal-line technology, the next logical avenue for computer vision in football is offside decisions. Offsides were first defined in 1863 and have seen several amendments in the years following. The modern rule gives the opposing team a free kick if an attacking player interferes in play while standing in an offside position (a full explanation is provided in Appendix A). While deciding if a player is involved in the play at any moment is subjective (and is difficult for an algorithm to judge), whether a player is physically offside is very computable. As one of the most antagonising ways to disallow a goal, fast and reliable offside identification is essential to improving the fan experience.

1.1 Motivation

Offsides are currently detected by two officials, called the linesmen (or assistant referees), who stand in line with the second-last defender to get the best view. Professional linesmen are highly skilled officials who make decisions to the best of their ability, but coming to the correct decision is often not possible in certain circumstances. If a player is obscured by several others or the moment the ball is played is unclear, it is extremely difficult for the linesmen to reliably determine the outcome. Since offside decisions have the potential to change the outcome of a game, incorrect decisions can ruin the experience for fans, which in turn has seen a rising level of abuse aimed at the officials in recent years. Therefore, achieving higher accuracy of offside decisions is crucial to the continual improvement of the game in the future.

1.2 Aims

This project will utilise computer vision techniques to design an approach which can quickly and accurately identify attacking players standing in offside positions, helping an official arrive at the correct decision during a match.

This project will be successful if a player standing in an offside position can be successfully determined in an image of a football match taken from a broadcast camera. This will involve meeting the following objectives:

- Identifying the vertical white pitch lines around the 18-yard area.
- Detecting the precise position of players on the pitch.
- Classifying the players into two separate teams depending on their shirt colour.
- Selecting one team as the defending team and the other as the attacking team in the scenario of the image.
- Determining which player on the defending team is the second closest to the goal line.
- Returning an offside decision for each attacking player.

1.3 Challenges

This project will face several challenges during its completion which must be addressed to ensure success.

Due to the commercial nature of football, most footage from professional matches is the property of a broadcaster, which substantially prevents its use

by others. Even in this non-commercial research project, finding a large, varied dataset could be difficult. This commerciality also leads to many stadiums having large stands for supporters, which are also captured in the broadcast footage. Any person-detecting model must ignore detections in this non-pitch region.

Every football image is slightly different, with many factors affecting the ability to detect features within them. Parameters such as weather, time of day, grass colour and team colours will impact the performance of a generic model. In addition to this, all images contain noise which must be dealt with to reduce its impact on accuracy.

Occlusions, where one player blocks another, are inevitable in any football picture, as players must compete closely with each other for possession of the ball. This is especially important for offside detections where attacking players are often very close to the opposing team's defenders. Using a dataset containing footage from only a single camera means any algorithm produced should pay particular attention to candidate players with multiple distinct colours – this could indicate an occlusion.

1.4 Report Structure

Section 1 (this chapter) has introduced the motivations behind the project, the problem this report is aiming to address, the goals this project will try to achieve and the that will be challenges faced along the way.

Section 2 will introduce the background research conducted to guide the implementation of the aims, explaining key theoretical points found. The features required by a robust offside detector and the various approaches previously taken are explored.

Section 3 describes the development process, detailing how the model was designed and implemented stage-by-stage. The theory behind each stage is explained in-depth to aid understanding.

Section 4 explains the training process taken and evaluates the performance of the model, showing the results of a limited experiment.

Section 5 follows by reflecting upon the project. The finished model is compared to the initial aims set out, summarising how well they are met and the limitations encountered. From this, potential improvements and avenues of further work are described.

Section 6 draws conclusions from the entire project and this report.

2 Background

This section discusses the background research required by a project such as this. The various computer vision techniques that could be applied to this project are summarised and explained, giving an idea of how many other researchers have previously approached this problem and why this project makes the choices it does.

2.1 Current Applications of Computer Vision in Sport

Computer vision in sports has developed down two distinct but related avenues: technologies used to improve decision-making and technologies used to improve clarity with the fans by offering replays. The first notable use of computer vision was from the company Hawk-Eye, whose ball-tracking technology was originally used in cricket at the 2001 Ashes [11]. This system was used to detect Leg-Before-Wicket (LBW) decisions, analysing and visualising the ball's trajectory to determine if it was going to hit the stumps. Hawk-Eye expanded its coverage to a second sport in 2003 when it was utilised in tennis for the Australian Open to determine if a shot was in or out of the court. This initial implementation was purely for television audiences to rewatch decisions; the system wasn't used for decision-making until the 2006 US Open [11].

Football has been no stranger to advancements in technology. As mentioned in chapter 1, goal-line technology has been used for almost 10 years in the Premier League. In recent years, substantial attention has instead been placed on the emergence of the Video Assistant Referee (VAR), which was trialled in several FA Cup games from the start of 2018 prior to its full introduction to the Premier League from the start of the 2019/20 season. VAR is not an automatic solution, as decisions are made manually by an official. The job of the VAR official is to review key moments in a game and refer the on-field referee to a monitor at the pitch side if they think a decision needs reviewing. Every goal is checked, with potential offsides in the build-up to a goal being an important consideration. Although the system has seen some improvement in decisions from pre-VAR seasons, the roll-out has been far from perfect. The offside checks performed by VAR have provided some of the most controversial incidents, with very marginal calls and

incorrect line-drawing drawing ridicule from pundits [12].

2.1.1 Semi-Automatic Offside Technology

It is worth noting that two months into starting this project, FIFA announced that a semi-automatic offside detection system would be put in place for the 2022 World Cup in Qatar [8]. The new system is much more advanced than the one implemented in this paper, using twelve dedicated tracking cameras mounted underneath stadium canopies. Each player is tracked by up to 29 data points, 50 times per second. Additionally, a sensor embedded in the match ball gives the precise moment the ball is kicked. This system achieved exceptional accuracy during its use at the World Cup and offered a 3D rendering of the scenario for fans to view, improving the clarity of each decision. However, an approach like this is very costly, with only the richest clubs and leagues being able to afford its installation. In comparison, the method put forward by this paper requires no new sensors or cameras to be added to stadiums.

2.2 Literature Review

To support the implementation of this project, a substantial review of the existing literature was undertaken. Academic works considering not only offside determination but also player and line detection have been considered.

[23] describes a complete model that takes a football image and produces an offside decision for each attacker. The system first finds the vanishing point of the image from the pitch lines, which is the point where all vertical or horizontal lines in the scene meet. It then detects players and uses an advanced pose estimation solution to find the body part of the player which is closest to the goal line. Players are then classified into two teams using both K-means and DBSCAN clustering. With this information, a line is drawn for each player from their furthest-forward limb (projected onto the ground if necessary) to the vanishing point, representing that player's 'offside line'. Offsides can then be determined through simple comparisons with the last defending player. This paper also has an accompanying dataset of offside scenarios from real-world football matches.

[17] is referenced by the previous paper, describing a very similar procedure for offside detection. Some slight variations are seen in the method to determine the vanishing point, where discernible vertical patterns in the grass aid the detection. This is not always possible with real football images, as stadiums use a variety of grass patterns, but worked well here since the dataset consisted of very consistent screenshots from the football video game 'FIFA 18'. Additionally, this paper takes user input for team templates, meaning no team identification is needed.

[5] defines methods for determining key moments during football matches, such as goals, important decisions and slow-motion replays found in a football broadcast. To find these moments, camera shot classification is performed, which

includes robust player segmentation using the dominant pitch colour and a cylindrical distance metric for use in the HSI colour space.

[34] builds on the previous paper by purely focusing on identifying the key features within a football match, including reliable pitch line and player detection, as well as some basic shot classification.

[10] describes a method of detecting field lines in football pitches. A Convolutional Neural Network (CNN) architecture is proposed that effectively finds pitch lines. However, this paper focuses on robot football matches as opposed to real-life ones.

[18] introduces a method to find statistics, including team possession, action zones and attacking directions, from football matches. A player and ball-detecting CNN (which can predict its uncertainty) is used before line analysis is performed to determine pitch regions. This enables somewhat precise stat generation.

Upon reading these papers, it became clear that methods able to effectively detect both pitch lines and players on each team would be needed, in addition to strong player classification.

2.3 Colour Model Selection

An unexpected consideration that had to be made was the colour model used to represent the football image. This required more research than expected to understand the benefits and drawbacks of varying models.

2.3.1 Additive Models

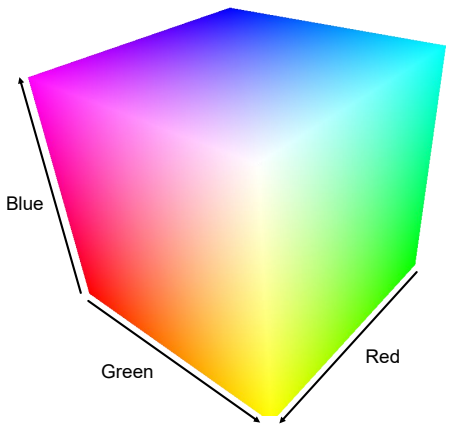


Figure 2.1: Diagram showing the lighter side of the cubic RGB model. Diagram taken from [25].

RGB, standing for Red Green Blue, is the most commonly used colour space by computers since pixels in a monitor contain a red, green and blue elements that can be lit in varying intensities. This is based on the theory that all visible

colours can be formed using the additive primary colours of red, green, and blue – this is why RGB is known as an additive model. Each colour has a red, green and blue component (or channel), which add together. The RGB model can be represented in 3D space by a cube (shown in Figure 2.1), where the distance from the origin along each axis corresponds to each channel value. RGB tends to be practically implemented with 8 bit channels, each containing 256 possible values, allowing for over 16.5 million unique colours [37]. In OpenCV, images are actually opened in BGR mode (Blue, Green, Red), which is essentially the same as RGB but with the channels reordered.

2.3.2 Hue & Saturation Models

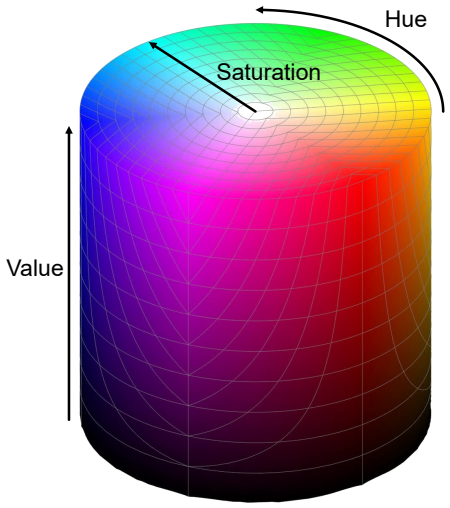


Figure 2.2: Diagram showing the cylindrical HSV model. Diagram taken from [36].

The issue with RGB is that it doesn't naturally relate colours in the same way that humans do. The human eye doesn't consider the red, green and blue levels of a colour, but rather its shade and saturation. This motivates the more natural HSI, HSV and HSL models (where H stands for hue, S stands for saturation and the final letter stands for intensity, value and lightness respectively). These models are cylindrical in shape (shown in Figure 2.2). Hue is an angle representing a complete spectrum of rainbow colours (red is 0 °, green is 120 ° and blue is 240 °). The saturation is how vivid the colour is, where 0 is greyscale and 100 is fully saturated. The final component varies between the models but generally describes how light or dark the colour is. 0 is always pure black and 100 is purely saturated in HSI and HSV (but pure white in HSL) [16]. OpenCV has representations of HSV and HSL (called HLS), but not HSI. The saturation and intensity channels have 256 values. Since the hue channel must also fit inside an 8 bit channel, it can take only 180 values (with a resolution of 2 °) [19].

2.4 Pitch Line Detection

Many methods exist for finding lines in an image. The papers read described simpler line detection approaches [5, 17, 18, 23, 34] and more state-of-the-art, deep learning approaches using Convolutional Neural Networks (CNNs) [10].

2.4.1 Traditional Approaches

Equations of lines in an image can be found well with longstanding methods. Many of the papers read first used an operation to find key edges in the image. The Hough line transform can then be applied to obtain the equations of these lines.

2.4.1.1 Hough Line Transform

Lines can be robustly found using the Hough line transform. Each line can be expressed by a polar coordinate containing the parameters ρ and θ, representing the respective magnitude and orientation of the normal vector to the line between the origin and the nearest point on the line to the origin. The parametric equation of these lines is given by Equation 2.1.

$$x \cos\theta + y \sin\theta = \rho \tag{2.1}$$

Since a single point (x, y) has an infinite number of potential lines that intersect it, there is a continuous spectrum of ρ and corresponding θ values representing the lines passing through this point. Hough parameter space is where ρ is plotted against θ, as opposed to Cartesian space where x is plotted against y. When a point from Cartesian space is plotted in parameter space, it becomes a sine wave showing all the possible parameter values of the lines intersecting it. If a second point's parameter lines are plotted, the point of intersection between the parameter lines gives the parameter values of the straight line intersecting both points. This enables the equation of the line intersecting multiple points along an edge to be determined, where the algorithm will identify a line once a threshold number of intersections in the parameter space is met [9, 22]. The process of the Hough line transform is shown in Figure 2.3.

[5, 17, 18, 23, 34] all made use of the Hough line transform in some form. However, for Hough's method to find line equations the pitch lines in the image must be found.

2.4.1.2 Canny Edge Detection

Lines tend to be found along edges in the image, which allows for the use of traditional edge detection methods. Canny edge detection could be used to find strong edges, with the longest edges being expected to be pitch lines – this is used

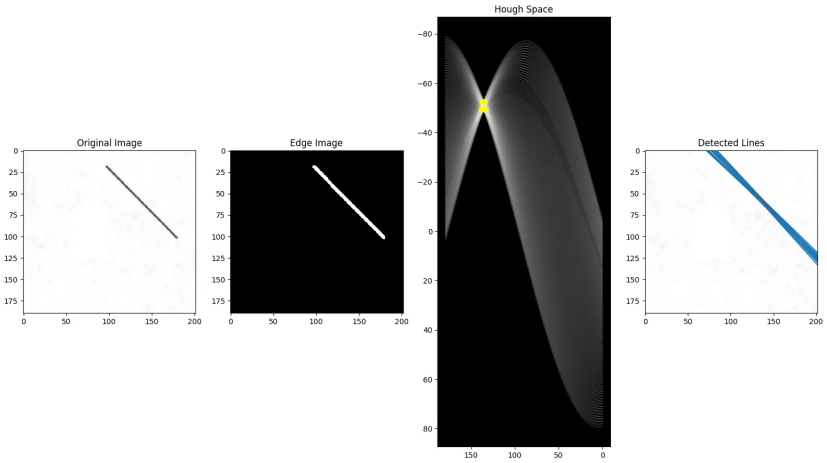


Figure 2.3: Diagram showing the stages of the Hough Transform, including the Hough parameter space. Diagram taken from [15].

by [18, 23]. However, edge detection will return the edges found around players, introducing noise into the Hough line transform, and will detect two edges at each line, since the pitch lines are several pixels wide.

2.4.1.3 Morphological Top-Hat Operator

Alternatively, [34] replaces the Canny function with the morphological top-hat operator. This function, along with the black hat, detect contrasted objects on non-uniform backgrounds. The top hat operator takes the difference between a greyscale image and its morphological opening with some structural element, which reveals brighter regions on darker backgrounds [3]. This means it successfully extracts white structures on the pitch, which are predominantly the pitch lines. Non-maxima suppression is then applied to reduce structures to single-pixel lines, providing a cleaner input into the Hough line transform, with less noise arising from players. The effects of this operation are shown in Figure 2.4.

Figure 2.4: Diagram showing the effect of the morphological top-hat operator. Diagram taken from [3].

2.4.2 Neural Network Approach

Another method explored was designing a neural network to detect lines, which has the potential to make stronger predictions. One such network is the CNN described in [10]. The main core of the network consists of four 'N-filter Inception-v2' stages leading into a final convolutional layer. Each inception stage is divided into three N-filter branches which are concatenated upon completion into a $3N$ channel. Each branch is composed of a varying number of small blocks, consisting of a convolutional filter, batch normalisation and rectified linear unit (except for the third branch, which contains a max pooling filter). Furthermore, uncertainty in the output can be estimated by making the network predict the parameters of a 1D Gaussian distribution for each column. This potentially enables the network to be more robust to occlusions.

2.5 Player Detection

As found with line detection, no two papers used the same approach to find players in the image. Various methods including mask manipulation, pose estimation and the use of neural networks have been explored.

2.5.1 Mask Manipulation

Players, officials and the football are assumed to be the only other objects present on the football field other than the pitch lines. Therefore, if the colour of the grass is found and masked, any gaps in the mask could contain players – this is the approach taken by [5, 17, 34]. A mask is a binary image describing areas of interest in an image. Pixels with a value of 1 are considered useful to the application whereas pixels with 0 are ignored. An example of masking an image is shown in Figure 2.5. This approach aims to create a mask containing only pixels belonging to players or field lines, using the dominant colour algorithm described in [5]. This algorithm works by finding an interval around the peak colour in an image histogram, before finding the mean within this bound. This prevents outlying colours from having a significant effect on the colour found.

Figure 2.5: Diagram showing the effect of applying a pitch mask to a football image.

In addition to describing the dominant colour algorithm, the paper also describes the 'cylindrical distance metric' (taken from [24]), which is used in the

HSI colour model. It is better suited for HSI than the standard Euclidean distance measure because it takes the angular nature of the hue channel into account (where, for example, hue values 10 ° and 350 ° are very similar colours). This metric accounts for achromatic colours too. An achromatic colour (a greyscale colour with a very low saturation value) should have an undefined hue, yet in memory must be assigned a value. Therefore the hue of an achromatic colour often seems arbitrary. The same shade of white, for instance, could have completely different hue values in different pixels, which can cause classification errors if not accounted for. Consequently, the cylindrical distance only uses intensity in the calculation for such colours. Using this metric, pixels lying within a threshold distance from the dominant colour are considered part of the mask. Once this mask has been calculated, it should only contain holes where objects lie.

Performing a morphological closing operation on this mask fills these holes, resulting in a mask of the playing area (shown in Figure 2.5) and all objects within. A simple subtraction returns a mask containing only objects on the pitch. Finally, lines can be removed using the results from line detection and player-shaped blobs can be filtered using height and width constraints. This approach is very logical but has the potential to struggle with any noise in the mask.

2.5.2 Pose Estimation

Alternatively, [23] detects players by finding the individual body parts of each player. This uses a pose estimation system from [1], which performs a greedy search to find correspondences between 2D poses identified from different camera angles. The algorithm estimates 2D poses before performing error correction, 2D pose association and triangulation to form a 3D skeleton. Since the application of this in [23] only uses one camera angle, it is assumed that the 3D skeleton estimation is not used. Once an approximate skeleton consisting of a player's head, shoulders, hips, knees, and ankles has been found, the region that lies between the player's knees and midriff is selected. RGB colour histograms from this region of interest are concatenated to form the feature vector used in the player classifier.

2.5.3 Neural Network Approaches

Neural networks lend themselves even more to player detection than pitch line detection, since many models already exist for detecting people. 'You Only Look Once (YOLO)', proposed by [27], was originally designed as a generic object detection that improved greatly on the performance of other methods. YOLO is very adaptable, having been applied to football in [2]. YOLO works by inputting an image into 24 convolutional layers designed to extract features, before running these results through two fully-connected layers which calculate bounding box predictions and probabilities.

YOLO is a generic object detector, therefore requiring adaptation for use in football. A different architecture could prove more effective for identifying players. [18] describes an approach using a CNN to detect both the players and the ball, in addition to a certainty of the classification. The CNN, called 'FootAndBall', is taken from [14] and consists of five convolutional stages performed one after another, decreasing the spatial resolution while increasing the number of feature channels. The feature maps are generated bottom-up before being processed top-down, with lower-level feature maps being combined with higher ones. The player confidence map and bounding box regressor are extracted from the results of the fourth and fifth convolution, with the rest being used for the ball classification.

2.6 Player Classification

Once players have been identified, they must be classified into two teams. This is made more difficult by the inclusion of a goalkeeper and up to three officials in any image.

2.6.1 Clustering

Clustering is a simple unsupervised approach to grouping similar features. Across the papers reviewed, two methods appeared: K-means and DBSCAN.

2.6.1.1 K-Means Clustering

K-means clustering [30] is perhaps the simplest clustering algorithm, grouping data into k groups. Cluster centres are randomly initialised in the dataset before multiple passes are performed. Each pass assigns a data point to the closest cluster centre, before updating the value of each cluster centre based on the points closest. K-means does a good job separating linear data but is strongly influenced by the number of cluster centres chosen and the starting cluster centres [30].

2.6.1.2 DBSCAN Clustering

DBSCAN (standing for 'Density-Based Spatial Clustering of Applications with Noise') [29] is another unsupervised clustering algorithm which groups data into an arbitrary number of groups. DBSCAN has two parameters: the radius around each data point ε and the minimum number of data points required for that point to be classified as a core point n. A circle (or hypersphere depending on the data's dimensionality) of radius ε is drawn around each data point. If this region contains at least n points, then this point is classified as a core point. If the region contains fewer points than n, it is classified as a border point, and if no points are found within it is treated as noise. Touching core and border points are grouped in a bottom-up fashion into the same cluster. This method

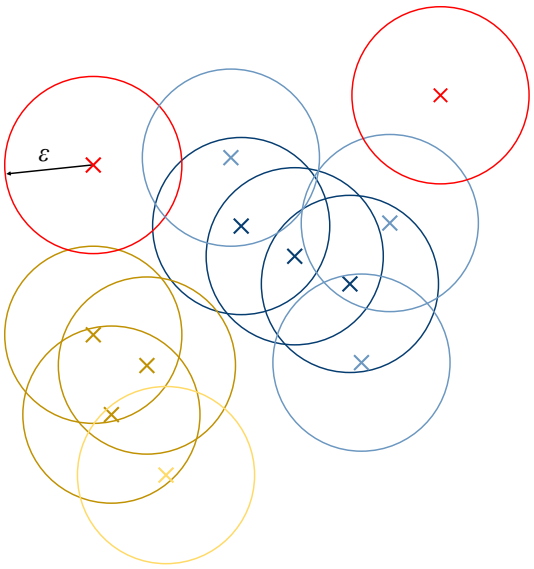


Figure 2.6: Diagram showing how points are clustered by DBSCAN. The darker-coloured points are core points and the lighter-coloured points are border points. Blue and yellow points are part of two separate clusters; red points are noise classifications.

is visualised in Figure 2.6. One huge advantage of DBSCAN is it is robust to outliers, as it classifies them as noise. However, it is significantly affected by the chosen ε value [29].

2.6.2 Template Matching

Instead of grouping similar blobs, they can be matched with templates of each team's kit – this is the approach taken by [34]. Here, a template for both teams and their goalkeeper is inputted. These templates are scaled using an estimation of the relative player size in the image. Each scaled template is moved across every pixel of the blob to find the position with the highest similarity. The template with the greatest similarity is selected and, if that similarity is greater than a threshold, the blob is identified as a player for the team represented by that template. This method requires representative templates to be provided, requiring substantial training and testing data.

2.7 Summary

After reviewing this literature, this project will follow a broadly similar approach to [23], using vertical pitch line detections to find the vanishing point, from which offsides can be determined. This was chosen over [17] due to the latter's reliance on the FIFA 18 dataset, which only approximates real-world scenarios. [23], on

the other hand, has a publicly available dataset of real football matches. Some necessary modifications will have to be made to the various stages of this method however to ensure the project meets deadlines. For detecting lines, a simpler approach using the top-hat operator and the Hough line transform will be implemented (similar to the method described in [34]). While robust pose estimation is mandatory for accurate real-world offside detection, the timescale of this project made this infeasible. Therefore, a mask-based approach was chosen instead, where a player's furthest back body part would be estimated by a bounding box. DBSCAN clustering will be used to classify the two teams, taking advantage of the noise classifications to identify the goalkeeper and officials.

Despite their substantial performance, neural networks were ultimately rejected after careful consideration. Acquiring the quantity of labelled data required for training the YOLO, FootAndBall or field boundary models was infeasible, forcing the discovery and manual labelling of new data. This would take significant effort away from developing a solution, with time instead spent preparing data. In addition, football is a highly commercialised sport with a complex set of licences that need to be navigated when obtaining new data. This makes it preferable to choose a pre-existing dataset.

3 Design & Implementation

This section describes the process taken to implement an offside-detecting algorithm. Inspiration was drawn from the background research outlined in the previous chapter, which guided the design process. The algorithm proposed is outlined briefly before being broken down into stages and explained in-depth.

3.1 Algorithm Overview

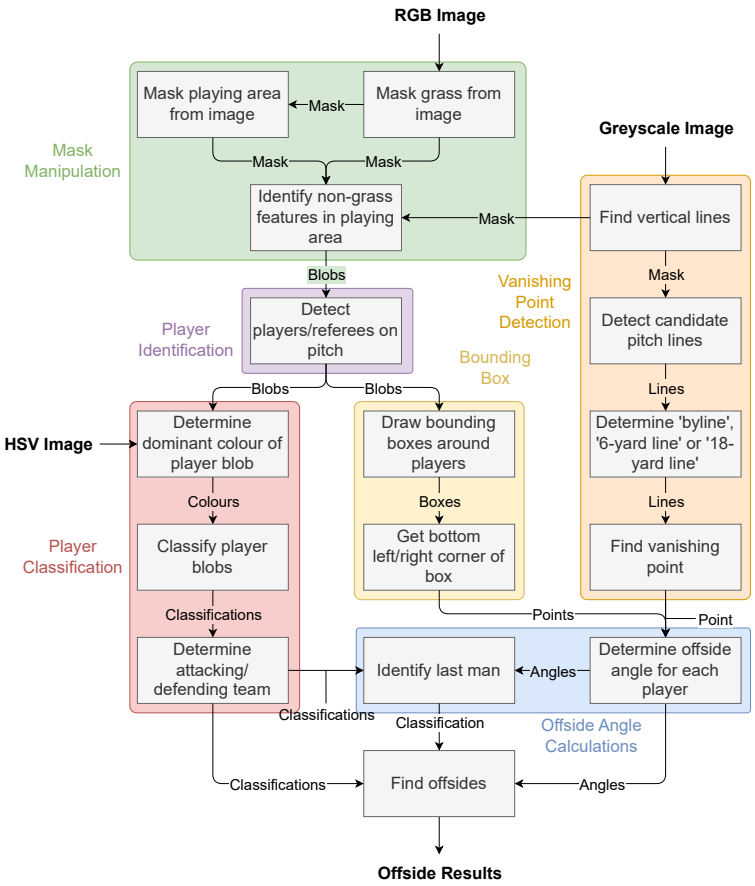


Figure 3.1: Diagram showing the stages performed by the offside detection algorithm.

Using information learnt in chapter 2, the structure of the offside-detecting algorithm was derived – this is shown in Figure 3.1. Designing the algorithm diagrammatically before development began enabled each stage to be identified and abstracted from others, simplifying the process. In order to successfully detect an offside, the algorithm complete the following tasks:

1. The algorithm first needs to mask the grass in the image, leaving gaps where objects are located within. This mask is manipulated to gain masks of both the playing area and objects within.

2. Players are initially detected from the object mask before refinements are made and their kit colour is found. With this information, any detections containing two players can be identified and separated, and players can be grouped into teams using DBSCAN.

3. Lines are detected using a separate method: top-hat morphology. This inputs into a Hough line transform, which is refined with DBSCAN. Geometric calculations allow for the vanishing point to be determined, from which offside lines for each player can be derived. This enables the defending team, second-last man and offside players to be found.

3.2 Pitch Masking

The first stage is to identify the playing area and the objects inside it. Football is predominantly played on grass or a grass-like surface with highly contrasting white lines marking areas of the pitch. This assumption can be used to select only the areas of the image with grass-coloured pixels, allowing for players to be found.

3.2.1 Dominant Colour Identification

Acquiring a mask of the grass in the playing area necessitates finding the average colour of the grass. The pitch typically takes up the majority of any football image, meaning it is reasonable to assume that the dominant colour in the image will be that of the grass. [5] provides an effective method for finding the dominant colour in an image histogram. First, the peak bin i_{peak} of the histogram H is found, which is the bin with the highest pixel count. An interval $[i_{\min}, i_{\max}]$ is then defined around i_{peak}. Both $i_{\min}$ and $i_{\max}$ must satisfy the conditions listed in Equations 3.1 - 3.5, which define the index as the smallest (or largest) index to the left (or right) of the peak that has a threshold number of pixels. The threshold is defined as a percentage k of the peak value, which is set by the user and $0 < k < 1$.

$$\mathrm{H}\left(i_{\min}\right) \geq k \times \mathrm{H}\left(i_{\text{peak}}\right) \tag{3.1}$$
$$\mathrm{H}\left(i_{\min} - 1\right) < k \times \mathrm{H}\left(i_{\text{peak}}\right) \tag{3.2}$$
$$\mathrm{H}\left(i_{\max}\right) \geq k \times \mathrm{H}\left(i_{\text{peak}}\right) \tag{3.3}$$
$$\mathrm{H}\left(i_{\max} - 1\right) < k \times \mathrm{H}\left(i_{\text{peak}}\right) \tag{3.4}$$
$$i_{\min} \leq i_{\text{peak}} \leq i_{\max} \tag{3.5}$$

Once this bound has been defined, the dominant colour C_{H} in each channel's histogram H is the mean value within the bound, given by Equation 3.6. This process is visualised in Figure 3.2.

$$C_{\mathrm{H}} = \frac{\sum_{i=i_{\min}}^{i_{\max}} \mathrm{H}\left(i\right) \times i}{\sum_{i=i_{\min}}^{i_{\max}} \mathrm{H}\left(i\right)} \tag{3.6}$$

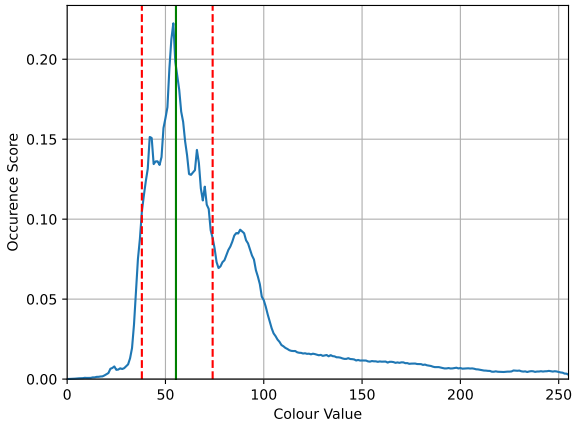


Figure 3.2: Diagram showing where the bounds (the black dashed lines) are placed around the peak of an image histogram, for $k = 0.4$. The dominant colour (the solid green line) is the mean within this region.

Sometimes the football pitch contains high amounts of contrast caused by extreme sunlight and hard shadows. This variation in the pitch's illumination is often incorrectly identified by the algorithm as two separate colours, implying that a singular dominant colour won't always reliably mask the whole pitch. Therefore, two additional parameters n and c are defined when calculating the dominant colour. n is the maximum number of dominant colours for the algorithm to find within the bounds and c is the percentage of the peak value that another index must be greater than to be considered a separate peak, where $0 < c < 1$. Therefore, for the peak index i_{p_1} and a secondary peak i_{p_j} (for distinct peaks $1 < j < n$), the condition in Equation 3.7 must be met.

$$\mathrm{H}\left(i_{p_1}\right) \leq c \times \mathrm{H}\left(i_{p_j}\right) \tag{3.7}$$

This is implemented by using a queue sorted by histogram value, where an index is popped if it is considered part of the bound of a previous dominant colour, ensuring values already assigned to one peak aren't considered a peak of their own.

The colour model used here was a key consideration. HSV was initially used but, through testing, it was found RGB gave better segmentation of broadcast watermarks which can disrupt detections.

3.2.2 Gaussian Modelling

Since the dominant colour is modelled as a Gaussian distribution, a standard deviation value is required as well as the mean. Here, the sigma value σ is found by flattening the histogram's corresponding image channel, removing any values not within $[i_{\min}, i_{\max}]$ and calculating the standard deviation.

Once both a dominant colour and sigma value have been calculated, the region to mask is the range between x standard deviations above the mean and x standard deviations below, where x is a user-defined parameter controlling the number of deviations. These deviation bounds are shown visually in Figure 3.3. Rather than computing if each pixel value lies within this range, the bounds $C - x\sigma$ and $C + x\sigma$ are found which enables the use of a more efficient OpenCV function `cv2.inRange()`. This utilises NumPy arrays to broadcast the comparison across the whole image.

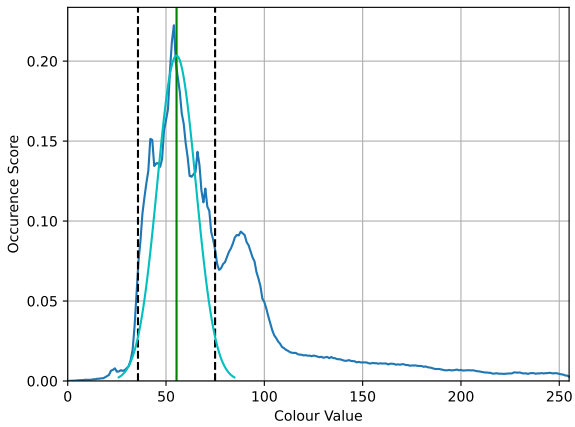
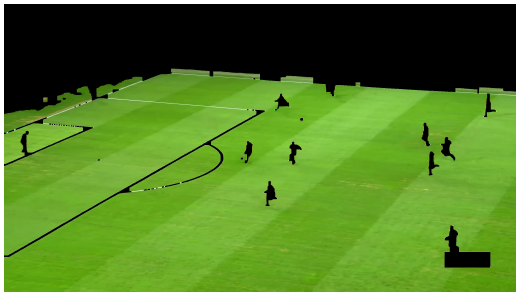


Figure 3.3: Diagram showing how a Gaussian distribution is modelled around the dominant colour (the solid green line) in an image histogram. The dashed black lines are placed 3 standard deviations from the dominant colour.

For masking multiple dominant colours, an empty mask $\mathbf{M}_{\text{grass}}$ is initialised. Each dominant colour C_i and its associated sigma value σ_i are separated into three channel groups which are then added and subtracted from one another to derive their bounds $C_i - x\sigma_i$ and $C_i + x\sigma_i$. These are passed as parameters into `inRange()` with the result being combined with the $\mathbf{M}_{\text{grass}}$ using a bitwise-OR operation.

3.2.3 Mask Manipulation

Now that the mask of the grass has been found, it must first have noise reduction performed. Some pixels in the crowd are close enough to the dominant colour to be included in the mask, resulting in small patches of noise not in the playing area. As well as this, some grass pixels inside the mask are just outside the range and are not included, leaving little black holes in the mask. To correct this, a morphological opening operation is performed first, which removes the majority of noise not in the playing area. After removing these, holes inside the mask can be safely removed with a gentle morphological close, which removes small holes in the mask but maintains large holes, which may contain features. These operations must be done in this order since performing the close first could result in non-grass noise becoming larger. The processed grass mask is shown in Figure 3.4(a).



(a) Grass mask.

(b) Pitch mask.

(c) Object mask.

Figure 3.4: Image masks found for a football image.

After removing this noise, a more aggressive close is performed which fills in all holes inside the grass mask. This results in a mask of the playing area

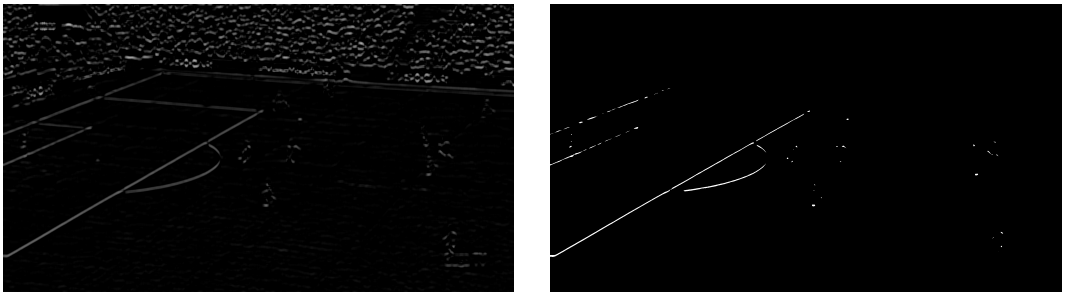
$\mathbf{M}_{\text{pitch}}$ (Figure 3.4(b)), with only the stands of the stadium being excluded. The grass mask is then subtracted from this mask to give the object mask $\mathbf{M}_{\text{obj}}$ (Figure 3.4(c)), which contains potential players. Again, some simple morphology is performed to remove some more noise.

3.3 Line Detection

The next step in the algorithm is to find the vertical pitch lines in the image. Using the pitch lines, a vanishing point can be found representing the point where all parallel lines in the real world meet in the image plane. Finding this point is crucial for comparing each player's position horizontally across the pitch.

3.3.1 Morphological Top-Hat Operator

As mentioned in section 2.4.1.3, the morphological top-hat operator performs the subtraction of a greyscale image and its opening, which effectively finds the brighter structures in the image. The top-hat does the best job of certain structures when the size of the kernel is similar, which is why a long, thin vertical kernel is used. As the brightest structures on the pitch, the white pitch lines are well found by this method, as shown in Figure 3.5(a). A threshold operation is performed to binarise the image and the playing area mask is used to ignore pixels in the crowd. The final step is to perform an erosion to thin the line mask down, the result is shown in Figure 3.5(b).



(a) Top-Hat result. (b) Line mask.

Figure 3.5: Vertical pitch line detection for a football image.

3.3.2 Hough Line

Now a mask of the vertical pitch lines has been found, a computable representation of these lines is needed for the offside calculations. The Hough line transform is one robust method to determine these equations (see section 2.4.1.1). The threshold number of points in the accumulator required to declare a line has been set deliberately low to return many candidate lines. As found through testing,

the performance of DBSCAN in the next step is greater when more candidate lines are provided.

3.3.3 Candidate Line Refinement

The output from the Hough line transform will contain many lines. Some of these will be clustered around the true line position, whilst others are simply noise. The first refinement is to filter out lines which aren't within bounds for ρ and θ that describe the range a true line will fall within. Next, the candidate lines require grouping to determine which pitch line they're representing. DBSCAN clustering was found to be the best method to find these clusters (see section 2.6.1.2), as data points belonging to the same line are densely clustered around the true point. scikit-learn's `DBSCAN` class was used, which returns a NumPy array of class labels corresponding to the inputted lines. Finally, the data points belonging to each cluster are averaged, deriving the best-fitting lines for each pitch line – this is shown in Figure 3.6.

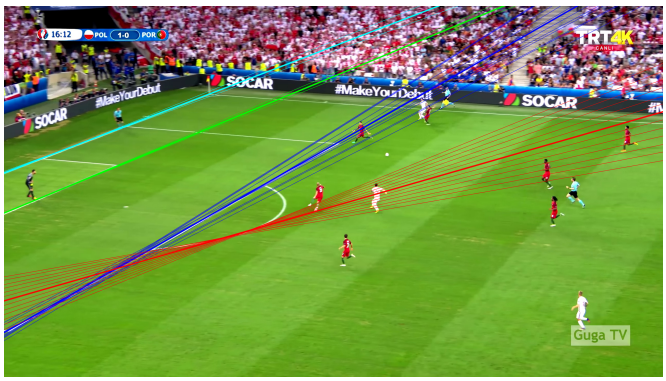


Figure 3.6: Image showing the candidate lines detected (the thinner lines) and how they've been clustered with DBSCAN to find the best-fitting lines (the thicker lines).

The first two lines detected, located on one of the by-line, 6-yard line, 18-yard line or halfway line, are chosen to find the vanishing point.

3.3.4 Pitch Direction Determination

The remainder of the algorithm has been built based on the assumption that the current defending team's goal is located on the left side of the image. Polar coordinates are defined from the top-left corner of the image, so the line parameters for images with the goal on the left and right sides differ completely. To avoid additional complexity when dealing with lines, images with the goal on the right side are flipped horizontally, thus moving the goal to the left side and ensuring the same algorithm works. However, the algorithm must correctly identify the pitch direction for this to work. This relies on the assumption that if there are no

valid pitch lines after line detection, the goal is on the right side and the image should be flipped. The algorithm must then remember whether the image was flipped so that it can be reversed later before a result is outputted.

3.4 Candidate Player Detection

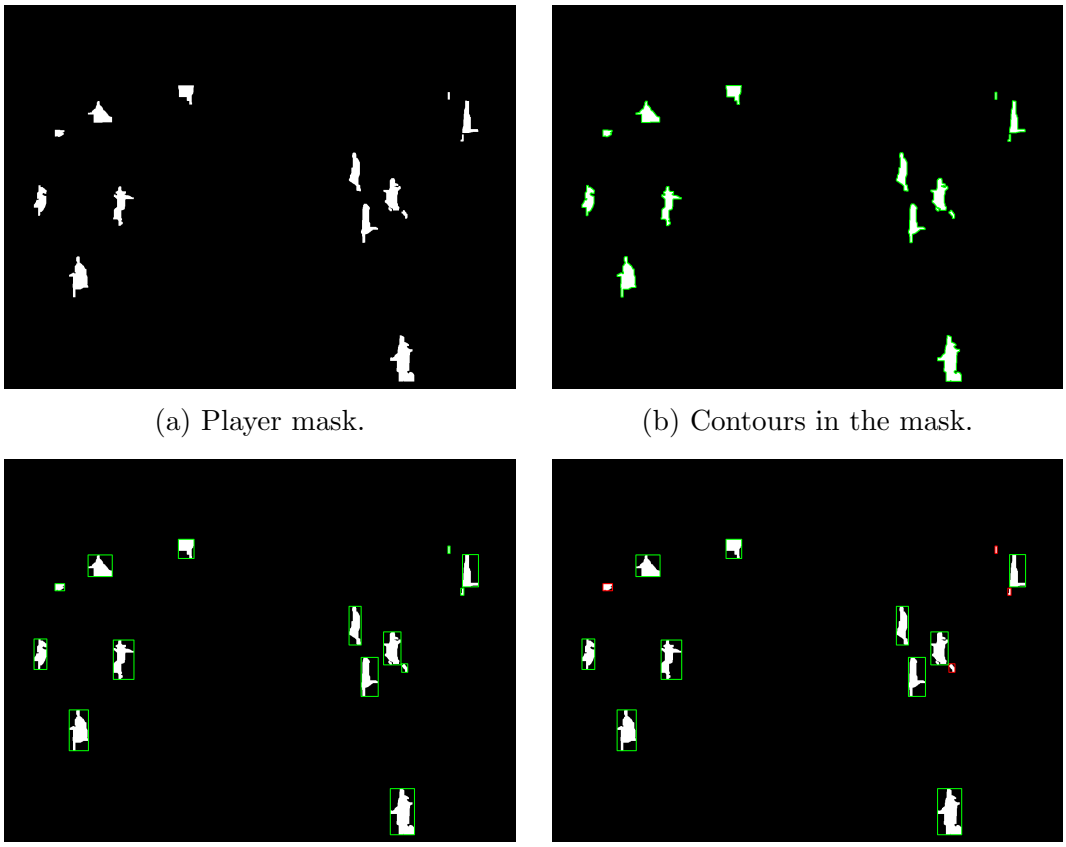
This stage uses the masks found in the preceding two stages to identify regions in the image which may contain players. Such regions are called 'blobs' (standing for binary large objects) and describe regions of interest in the image [13]. An area in a binary image with a sufficient number of connected 1-pixels forms a blob (the large white regions in Figure 3.7(a)), with smaller regions usually being noise. Any blobs found from the object mask won't be perfect, due to potential occlusions or noise in the mask. This is why the blobs are refined in a later step once they've been detected.

3.4.1 Blob Detection

As a preliminary step, the line mask found in the previous stage is subtracted from the object mask. This removes the lines in the object masking, leaving only blobs which may contain players and improving the blob detector's overall performance. Blobs can be found in an image by detecting their edges, known as contours. A contour is marked where there is a sufficient difference between neighbouring pixels. Since the object mask is binary, contours are easy to detect at boundaries between 0 and 1 regions. OpenCV's `cv2.findContours()` function is used to perform this contour detection [21], the results of which can be seen in Figure 3.7(b).

3.4.2 Bounding Box Determination

While the contours found in this step describe the approximate boundary of each blob, this information on its own is not especially useful to the algorithm. After contours in the mask have been identified, they're inputted into OpenCV's `cv2.boundingRect()` function [20] – the results of which are shown in Figure 3.7(c). This identifies the minimum bounding rectangle for each identified blob, parallel to the image axes. The vertical direction of the bounding boxes must be parallel to the vertical axis of the image to give a good estimation of the position of a player's furthest back body part, even if it is off the ground. `boundingRect()` finds the extreme values in both x and y directions, which it uses to calculate the box's size. It then returns four values per bounding box, describing the x and y position of its top left corner as well as its width w and height h. The bottom left corner, which is the furthest back position of the player on the ground, is therefore $(x, y+h)$. The final step is to remove any blobs which aren't within a given height $[h_{\text{lower}}, h_{\text{upper}}]$ and width $[w_{\text{lower}}, w_{\text{upper}}]$ constraint,



(a) Player mask.

(b) Contours in the mask.

(c) Bounding boxes in the mask.

(d) Filtered bounding boxes.

Figure 3.7: Process of detecting and refining candidate blobs in the player mask.

which ensures only player-sized blobs are found. This is shown in Figure 3.7(d), where green boxes are within the constraints and red boxes aren't.

3.5 Player Detection & Classification

The players identified in the previous stage are relatively accurate identifications, but some issues persist. Bounding boxes included in the identifications may not be players, but noise found in the grass. Two players may be contained within the same bounding box too if their blobs haven't been separated in an earlier stage. Therefore, some refinement of the boxes found is necessary. Since the dominant colour can be used to help determine if a box contains one, two or no players, this refinement is done at the same time as team identification.

3.5.1 Player Colour Identification

First, each bounding box in the image is treated as an image of its own, enabling the same dominant colour algorithm used earlier to be reapplied, this time to

the HSV image. An issue arising here is the algorithm will often still identify the grass colour as dominant if it's performed on the entire rectangular region. Hence the mask is also inputted to constrain the algorithm to just within the blob. The parameters n and c used by the dominant colour algorithm (explained in section 3.2.1) are set to allow two separate dominant colours to be found within the same bounding box.

3.5.2 Cylindrical Distance Metric

To estimate the number of players within a single blob, a colour similarity metric was required. The cylindrical distance metric from [5] was chosen over the Euclidean distance for the reasons summarised in section 2.5.1. While the metric was described for use with the HSI model, OpenCV doesn't offer a realisation of this model. After some research, it was determined that the HSI and HSV models were similar enough that this wouldn't cause an issue. The cylindrical distance $d(i,j)$ between two colours i and j (with respective hue H_x, saturation S_x and value V_x colour values) is given by Equations 3.8, 3.11.

$$d_V(i,j) = |V_i - V_j| \tag{3.8}$$

$$d_H(i,j) = \begin{cases} |H_j - H_i| & \text{if } |H_j - H_i| \leq 180^\circ \\ 360^\circ - |H_j - H_i| & \text{otherwise} \end{cases} \tag{3.9}$$

$$d_C(i,j) = \sqrt{{S_i}^2 + {S_j}^2 - 2S_iS_j\cos(d_H(i,j))} \tag{3.10}$$

$$d(i,j) = \begin{cases} \sqrt{(d_V(i,j))^2 + (d_C(i,j))^2} & \text{if } S_i > T_S \text{ or } S_j > T_S \\ d_V(i,j) & \text{otherwise} \end{cases} \tag{3.11}$$

The metric calculates two separate distances: the intensity distance $d_V(i,j)$ and the chromatic distance $d_C(i,j)$. $d_C(i,j)$ is only used to calculate $d(i,j)$ if either saturation value is greater than an achromatic threshold T_S. This threshold defines the achromatic region, where hue values can often be arbitrary. In these cases, only $d_V(i,j)$ is used.

3.5.3 Occlusion Handling

The first step in determining occlusions is to remove any grass identifications within blobs. If a dominant colour is within a threshold distance T_{grass} of the grass' dominant colour, it is classified as noise and removed.

Occlusions are assumed to occur when two non-grass colours exist inside a single blob and are sufficiently distinct from one another. If two dominant colours

are found for a blob and both colours are within a threshold distance $T_{\text{similarity}}$ of each other, it is assumed that the colours belong to the same player. Hence the mean is taken and no occlusion is found. However, if the two colours are not within $T_{\text{similarity}}$, the blob is assumed to have two players within. Players are separated within a blob by masking both dominant colours within the blob, indicating which region belongs to each. At this point, the newly created blobs are often small and not representative of the whole player – a player's kit, skin or hair may contain different colours to their shirt. Therefore, after a small morphological closure to remove noise, a large dilation is executed. The dilated region indicates which area of the original blob contains each player. Bitwise ANDing each dilated region with the original blob determines which part of the original blob should be classified as each individual player.

3.5.4 Team Clustering With DBSCAN

Once each player's dominant colour has been identified, players need to be clustered together. DBSCAN was chosen once again, this time for its ability to cluster outliers as noise (which will be expected with goalkeepers and referees), leaving just the two teams of players. ε has to be carefully set to ensure similar kits are clustered together; n is set to 2 to be able to cluster the referees if more than one is present in the image. Output from this classifier can be seen in Figure 3.8.

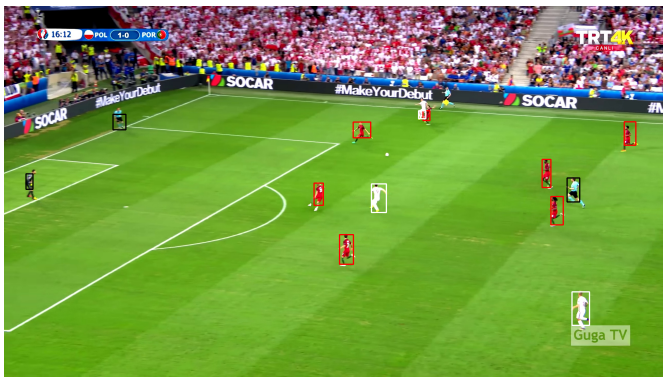


Figure 3.8: Image showing the team classifications performed by DBSCAN, with red detections on the defending team, white detections on the attacking team and black classifications being noise.

3.6 Line Calculations

The results from section 3.3 are finally used to calculate the vanishing point, which can then be used to find each player's relative position in the image.

3.6.1 Finding Vanishing Point

The vanishing point is crucial in determining the offside, as it gives a reference point for drawing any parallel line to the pitch's vertical lines – the effect of this can be seen in Figure 3.9. This point can be found by calculating the intersection of the vertical pitch lines. The point of intersection (x_v, y_v) of polar lines (ρ_1, θ_1) and (ρ_2, θ_2) is found using Equations 3.12, 3.13.

$$x_v = \frac{\rho_1 \sin\theta_2 - \rho_2 \sin\theta_1}{\sin\theta_2 \cos\theta_1 - \sin\theta_1 \cos\theta_2} \tag{3.12}$$

$$y_v = \frac{\rho}{\sin\theta} - \frac{x_v}{\tan\theta} \tag{3.13}$$

Equations 3.12, 3.13) are valid so long as $\theta_1, \theta_2 \neq 0, \frac{\pi}{2}$ radians. Otherwise, the sine, cosine or tangent functions are equal to 0, meaning the equations are undefined. In these cases, one or both of the lines is parallel to an axis, indicating x or y is equal to one of ρ_1 or ρ_2.

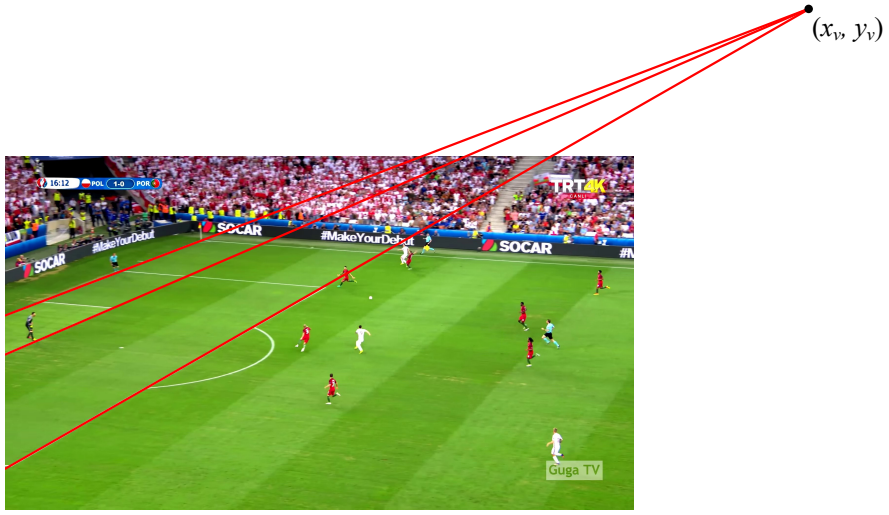


Figure 3.9: Diagram showing how the vanishing point is derived from the pitch lines in the image.

3.6.2 Finding Player Offside Lines

Once the vanishing point has been found, a polar line must be found between the bottom left corner of each player p's bounding box $(x_p, y_p + h_p)$ and the vanishing point (x_v, y_v). The resulting line will be in the form (ρ_p, θ_p) and is given by Equations 3.14, 3.15, where θ_p will be used to compare player positions.

$$\theta_p = \arctan \frac{x_p - x_v}{y_v - (y_p + h_p)} \tag{3.14}$$

$$\rho_p = x \cos \theta_p + y \sin \theta_p \tag{3.15}$$

Equation 3.14 is valid so long as $x_p \neq x_v$ and $y_v \neq (y_p + h_p)$.

3.7 Offside Detection

The final step is to determine any offsides, applying the rules listed by the offside rule. Two key identifications must be made which pave the way for a very simple offside comparison: the defending and attacking teams, and the second-last defender.

3.7.1 Identifying Defending Team

Robustly classifying one of the found teams as defending is not as simple as it may seem and many humans can't identify which team is which from a single frame. A more complex solution may look at the formation a team is playing and make an educated guess. This approach simplifies this, sampling the first q players, ordered by offside angles θ. The team with more players represented in this sample is chosen as the defending team. If a draw occurs (due to noisy classifications), the next player outside this sample is picked and so on until a team is picked. There exist scenarios where more of the players towards the goal line may be attacking players, but this assumption will be valid in the majority of cases.

3.7.2 Identifying Second-Last Defender

This step also may appear simple at first but has hidden complexity. The second last man was chosen to exclude the goalkeeper in the majority of cases since they're almost always positioned close to the goal line when their team is defending. Classifying the goalkeeper is decidedly difficult because rules require them to wear different kits to their outfield counterparts, with only a small badge indicating they belong to the same team. This method takes advantage of the noise classifications made by DBSCAN. If a non-grass noise classification exists towards the back of the defence, then it is selected as the defending team's goalkeeper and added to the team classifications. If a goalkeeper was found, the defending player with the second smallest angle is selected as the second-last man. However if no goalkeeper was found, then the player with the smallest angle is selected.

3.7.3 Determining Offside Decisions

The final step is very easy: determining the offside decisions to make. This simply takes the angle of the second-last player θ_{def} and compares the attacking players θ_{att_i} with it. If an attacking player's angle is less than the second-last defender's angle, then that attacker is offside. Conversely, if their angle is greater than or equal to the second-last defender's, then they're onside. This is shown in Equation 3.16. An example of an offside decision is shown in Figure 3.10.

$$\text{offsideDecision}\left(\theta_{\text{def}}, \theta_{\text{att}_i}\right) = \begin{cases} \text{offside} & \theta_{\text{att}_i} < \theta_{\text{def}} \\ \text{onside} & \theta_{\text{att}_i} \geq \theta_{\text{def}} \end{cases} \tag{3.16}$$

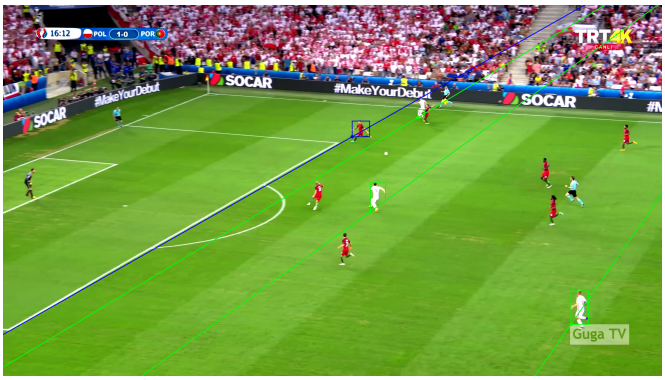


Figure 3.10: Image showing final offside classifications. The blue box is the second-last defender, the green boxes are onside attacking players and the red boxes are offside attackers.

3.8 Application Development

The following subsection will give a general overview of the codebase to support some of the development decisions.

3.8.1 Development Decisions

Python was selected as the language to develop this offside detector. This was due to the plethora of libraries available that abstract simple mathematical and image functions from their complex code implementations. In most cases, there was no point in creating these common operations since any new implementation will likely be less efficient and more restricted than the open-source alternative. This project has made extensive use of both OpenCV and NumPy, with NumPy's matrix operations especially helping the algorithm achieve better-than-expected

performance. In addition, scikit-learn's implementation of DBSCAN clustering was used.

All the code developed has been written to PEP8 standards to ensure consistency. This involves limiting the characters on each line to 79 (except in docstrings, where it is 72), using underscores in variable names and utilising four-space indentation, amongst other things [28]. Extensive use of docstrings can be seen, with every method and class having NumPy-style documentation. While Python is a dynamically typed language, type hints have been used throughout and, where appropriate, type checks have been performed. Comments have been included where it felt necessary to explain blocks of code. This has been done to support anyone else's use of the project and ensure high code quality.

3.8.2 Python Project Structure

The implementation itself is formed of five Python files, each storing some functionality:

- image.py – contains `Image` class and colour space information.
- detector.py – contains `OffsideDetector` and `OffsideResult` classes.
- run.py – contains code to run the offside detector by providing a command-line interface.
- parameters.py – contains the default parameter dictionary used by detector.py.
- utils.py – contains generic functions and type aliases used across the project.

3.8.2.1 Image Class

The `Image` class (contained within image.py) was written to encapsulate OpenCV images (which are simply 3D NumPy matrices) and store the image's colour space information. It was quickly realised that OpenCV's images were very basic and store no metadata about the image. In addition to this, `cvtColor()` requires a conversion code describing the current colour space as well as the colour space to be converted to. It was therefore decided that building a custom image class to store football images would reduce the complexity seen in the detector. `Image` contains over 30 methods which encapsulate a variety of OpenCV functions performed on the image stored.

3.8.2.2 Offside Detector Class

As the name suggests, the `OffsideDetector` class is what performs the offside detection. It contains a function `get_offsides()` which does this, containing the core structure of the algorithm described. Each small chunk has been moved into separate functions to enhance readability and aid development.

3.8.2.3 Offside Result Class

When reaching the end of the algorithm design, questions were raised as to the format of the output. The detector class needed not only to output decisions but the information to contextualise them, which includes player identifications, classifications and offside lines. Taking inspiration from many OpenCV and scikit-learn detector classes, a custom class to store results was written: `OffsideResult`. This stores a player matrix with information concerning each player as well as methods to visualise the offside result in the image.

3.8.2.4 Parameter Dictionary

Models such as this end up requiring many parameters for all the operations used at various stages. Instead of defining these in-place where used, they have been stored in a dictionary. This serves a double purpose of increasing understandability for other users, allowing for custom values to be used, and separating the parameters from the implementation to enable model training.

4 Evaluation

This section will describe how the model parameters were trained before explaining how its performance was tested through experimentation. The results of this experiment are analysed using various performance metrics, helping determine the strengths and weaknesses of the proposed model.

4.1 Model Training

The dataset used contained 480 images depicting various offside situations, but unfortunately, the accompanying labels were unusable for this project. Since insufficient time was available for data labelling, the 74 parameters used by the model had to be manually optimised. This proved a difficult task, as parameter values that work for one image won't necessarily work for others.

While each stage of the algorithm was built, a single reference image was chosen to test each stage and set parameter values. Upon completion of each stage, a small subset of images from the dataset was picked to confirm the model worked in more general scenarios. Each of these images was run through the model, with the partial solutions being observed. The parameters were then tweaked to find the values which worked for most images.

4.1.1 Model Performance Evaluation

The model produced a disappointing performance over the entire dataset, as it failed to return results for about 420 of the 480 images. The algorithm was programmed to return a `None` value when it couldn't identify adequate pitch lines, players, or teams to perform an offside detection. If an image returned a `None` value, it indicated that the model parameters were poorly tuned for that specific image. This was expected behaviour from the training process taken, since mannual training is not able to produce unbiased parameter values. With more time dedicated to labelling data and performing conventional model training with a loss function, the model would have a much greater chance of working on a broader set of images.

4.2 Evaluation Method

Although the model didn't work on every image, it did return a result for over 40 images, enabling some results analysis to be performed. To test how well the model worked, a method to measure model performance needed to be devised. Since testing the entire dataset was not feasible, a sample of 40 images which returned results was therefore taken from the dataset to manually test the model's performance. Each image would be observed and assessed on its ability to meet the following criteria:

- Whether the vanishing point was correctly drawn, indicating the pitch lines were correctly detected.
- The number of actual players found/not found, not including officials, and the number of non-player blobs being identified as candidate players.
- The number of players and officials correctly/incorrectly classified as 'Team A', 'Team B' and 'Noise', using the following rules:
  - Players are correctly classified if they're grouped with players on the same team and not as noise.
  - Officials are correctly classified if they're classified as noise.
  - Goalkeepers are correctly classified if they're classified as noise.
  - Non-player blobs incorrectly identified previously are correctly classified here if they're classified as noise.
- Whether the defending team was correctly chosen. Since the method to select the defending team is an approximation, this is not expected to be extremely high.
- Whether the offside line has been drawn from the correct point (the second-last man).
- When all the previous stages pass, the number of correct/incorrect/missed offside calls made.

4.2.1 Evaluation Metrics

After concluding the experiment, the results must be interpreted to determine if the model performs well or not. There are a variety of ways in which this can be done.

Since whether the vanishing point, defending team and second-last man were correctly identified is a binary result, these will be evaluated by a simple decimal score indicating the percentage of times each of these identifications was correct.

For player identifications and classifications, confusion matrices will be used. In the simplest case, a confusion matrix is a 2×2 matrix that neatly compares the predicted results to actual true values, containing cells for true negatives TN, false positives FP, false negatives FN and true positives TP [4]. The format of a confusion matrix is shown in Table 4.1.

		Predicted Classifications		
		Negative	Positive	Total
Actual Classifications	Negative	TN	FP	$TN + FP$
	Positive	FN	TP	$FN + TP$
	Total	$TN + FN$	$FP + TP$	$TN + FP + FN + TP$

Table 4.1: Diagram showing how classification information is represented in a confusion matrix.

By separating predictions into these categories in a confusion matrix, a model's classification performance can be analysed using several metrics. The metrics used here – accuracy (Equation 4.1), precision (Equation 4.2), recall (Equation 4.3) and the F1 score (Equation 4.4) – are widely used for this purpose [31].

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.3}$$

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{4.4}$$

4.3 Results

4.3.1 Vanishing Point Identification

Vanishing Point Accuracy	0.78

The vanishing point was successfully determined in 78% of the testing set, indicating the line detection method proposed is fairly robust. The cases where the vanishing point was incorrectly calculated were caused by incorrect line selection from the list of candidate lines. During testing, the Hough line transform often found lines in the semi-circle located at the edge of the 18-yard area (called 'the

D'). Since the semi-circle doesn't contain any straight parallel lines, line detections from this should not have been used to calculate the vanishing point. An alternative method for selecting the true pitch lines from the list of candidates could have improved this score.

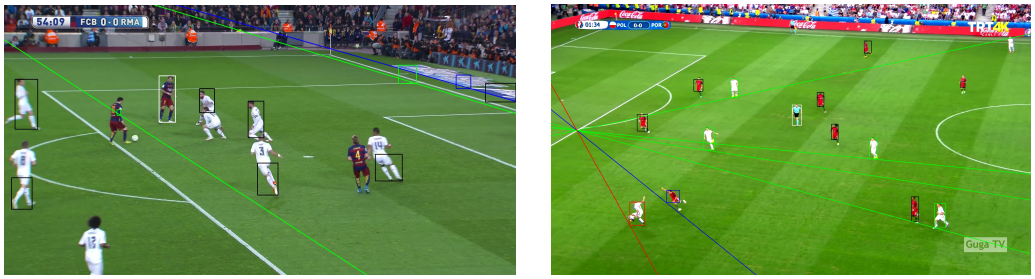


Figure 4.1: Examples of correct/incorrect vanishing point identification.

4.3.2 Player Identification

		Player Identifications		
		Not Found	Found	Total
Actual Players	Not Expected	0	44	44
	Expected	88	423	511
	Total	88	467	555

Table 4.2: Confusion matrix showing how often players were correctly identified.

The following metrics were calculated from player identifications shown in Table 4.2:

Accuracy	0.76
Precision	0.91
Recall	0.83
F1 Score	0.87

These results show that dominant-colour-masking can be used to successfully find players on a football pitch. However, the number of players not found is considerable, as is the number of non-player blobs. One potential issue in the algorithm's method is the subtraction of the line mask from the object mask to give better player detections. This seemed to be a logical way of preventing pitch lines from interfering with player detection but had the unintended consequence of destroying any player blobs which overlapped with the lines – this issue can be seen in Figure 4.2. Using dilation or closing morphological operations may have maintained more true players in the mask while removing pitch lines.

Figure 4.2: Example of player identification issues. The result includes a missed player, a non-player blob incorrectly identified as a player, and two improperly-handled occlusions.

The non-player blobs that were incorrectly identified typically were from fans, adverts or pitch lines. Football stadiums contain huge stands filled with supporters wearing the same shirts as their team, as well as many brightly-coloured advertisements surrounding the pitch. Both fans and adverts have the potential to cause detection issues if the playing area isn't correctly segmented. Fans or adverts being included in player detections implies the pitch segmentation was poor. On rare occasions, particularly thick pitch line regions were also classified as players; this occurred most frequently around 'the D' when one of the teams played in white. This could be fixed with better methods for removing noise from the player mask, as mentioned above.

4.3.3 Player Classification

		Predicted Classes		
		Noise	Player	Total
Actual Classes	Noise	55	42	97
	Player	18	387	405
	Total	73	429	502

Table 4.3: Confusion matrix showing how often candidate blobs were classified with a team, as players, or as noise.

The following metrics were calculated from the noise/player classifications shown in Table 4.3:

These results show that DBSCAN can differentiate players from noise, with an accuracy of 88%. The referee and goalkeeper were almost always correctly classified as noise when identified, and many non-player regions were correctly classified too. However many pitchside adverts were still incorrectly classified with a team. This was generally due to a high colour similarity between adverts

	Accuracy	0.88
Noise	Precision	0.75
	Recall	0.57
	F1 Score	0.65
Player	Precision	0.90
	Recall	0.96
	F1 Score	0.93

and team kits. The way to improve this would be to remove these non-player blobs at an early stage in the pipeline. Potentially using a different feature to describe players on each team might also reduce these classifications.

		Predicted Teams		
		Team A	Team B	Total
Actual Teams	Team A	135	8	143
	Team B	4	240	244
	Total	139	248	387

Table 4.4: Confusion matrix showing how often candidate blobs were classified with Team A or with Team B.

The following metrics were calculated from the team classifications shown in Table 4.4:

	Accuracy	0.97
Team A	Precision	0.97
	Recall	0.94
	F1 Score	0.96
Team B	Precision	0.97
	Recall	0.98
	F1 Score	0.98

The team separation results show the dominant colour and cylindrical distance metrics are an effective method of separating the two teams in a football match, with an accuracy of 97%. Incorrect classifications happen very rarely, typically when occlusions with players or pitchside adverts are not handled properly.

4.3.4 Defending Team & Second-Last Man Identification

The defending team was correctly selected 68% of the time. While this is not a particularly good score, choosing a defending team from a single frame with no prior knowledge of which team is which is a hard task. In fact, many humans will often struggle to determine this when glancing at a match. Incorrect identification and classification of non-player blobs at the edge of the pitch was the

Defending Team Accuracy	0.68
Second-Last Man Accuracy	0.57

most common cause of this, implying that if player identification improved, this assumption would improve likewise.

With a score of 57%, the second-last man selection did not perform well at all. The issue here was the incorrect separation of the crowd from the playing area, mentioned in 4.3.2. If a fan or advert is included in the player recognition, they will almost always be selected as the second-last man due to their offside angle being much less than any other player's angle. Better optimisation of the mask manipulation parameters would help solve this issue. This problem can be seen in Figure 4.2, where the advert behind the goal has been classified as the second-last defender.

4.3.5 Offside Classifications

		Predicted Decisions			
		Onside	Offside	Missed	Total
Actual Decisions	Onside	132	9	43	184
	Offside	3	3	3	9
	Invalid	6	6	0	12
	Total	141	18	46	205

Table 4.5: Confusion matrix showing results of offside classifications.

These results in Table 4.5 show the algorithm's performance is poor. The final offside detection method itself works well, however, the disappointing player and line identification done in earlier stages hinders this stage's success. The biggest issues highlighted here are attacking players the algorithm has failed to find and give a decision for (referred to as 'missing') and non-player blobs the algorithm has given a decision for (referred to as 'invalid'). Improvements in the player identification and classification pipeline mentioned previously will improve these results. Some examples of offside classifications are given in Figure 4.3. Additionally, offside decisions were incorrect more often than onside decisions. This was due to onsides being more common throughout the dataset and the second-last man detection being wrong.

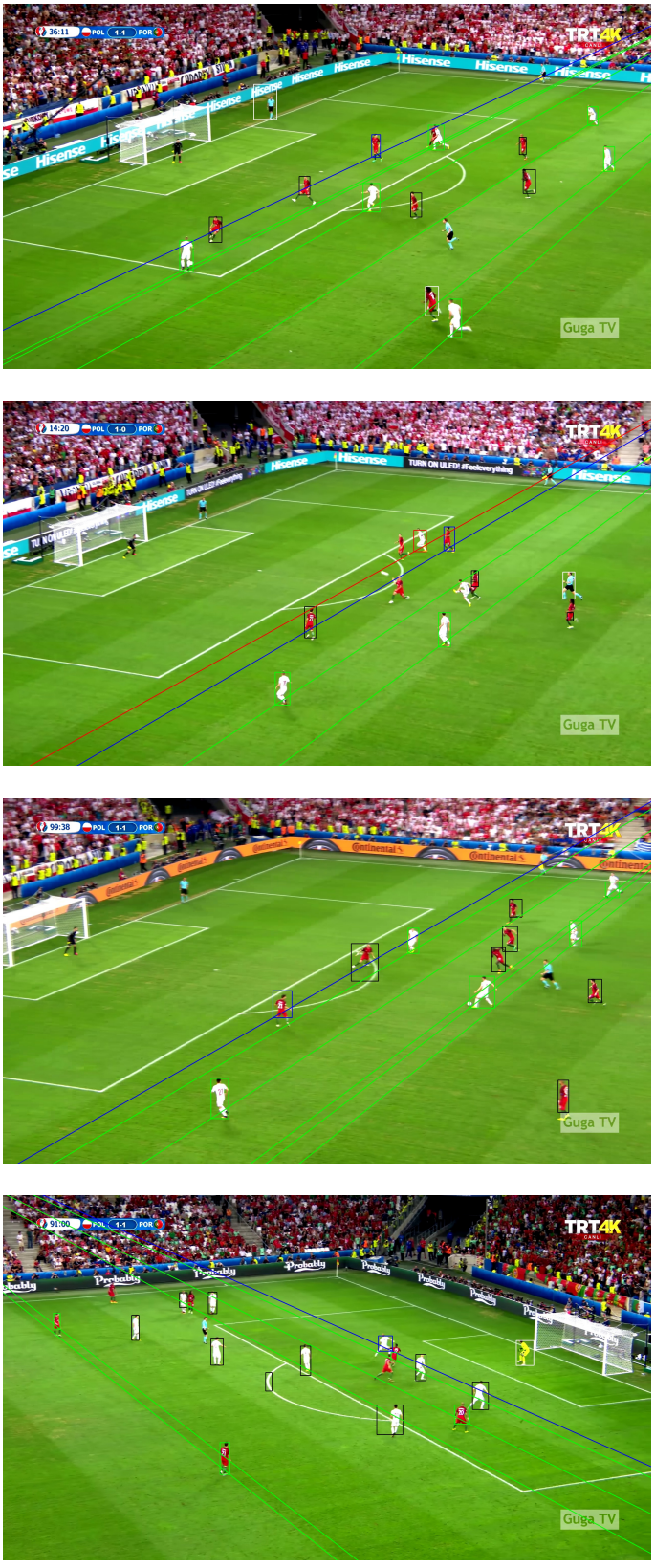


Figure 4.3: Examples of the results outputted by the detector.

5 Reflection

This section will explore what this project has achieved upon completion as well as improvements that could have been made. After summarising these points, potential avenues for future work will be suggested.

5.1 Achievements

The intended final product of this project was a working offside detector that would provide an accurate determination of each attacking player's offside position. Concerning this aim, the project is somewhat successful: the algorithm can detect players in offside positions in some broadcast football images. To meet this aim, several important achievements had to be accomplished:

- Vertical pitch lines are detected around the 18-yard area, which can be extrapolated to find the vanishing point of the image.
- Players are found in non-green regions of the playing area. Some occlusions are successfully detected and corrected.
- Each player's furthest-back body part, projected onto the ground, is compared with the vanishing point, giving an angle accurately representing their position on the pitch.
- Players are classified into separate teams based on the dominant colour of their kit.
- Upon selecting a defending team, the penultimate player to the goal on this team is identified.
- All detected attacking players are compared to the offside line, returning whether they are in an offside position.

5.2 Limitations

While the project was successful in detecting offsides, the project has significant limitations. Many images in the dataset failed to return a result, due to a lack of

pitch line and player detections. At least two valid pitch lines must be found to calculate the vanishing point, and at least two players must be found for each team for DBSCAN to work. While DBSCAN clustering can effectively group dense data points if ε is set correctly, the performance of DBSCAN is very sensitive to this parameter value. Since DBSCAN is used in both pitch line refinement and player classifications and both of these ε values were set using a limited proportion of the training data, it can't be expected to perform well across the whole data set.

The algorithm in its current state doesn't perform any ball detection. This isn't necessary for the majority of offside scenarios since the ball is most frequently in front of the defence. However, when the ball is behind the second-last defender, it becomes the object from which the offside line is drawn.

5.3 Improvements

From these limitations, it is clear there is still much that could be improved. Factors such as strong shadows in stadiums and occlusions between players and pitch lines caused considerably bigger problems than initially thought. The reference image was chosen to test the algorithm during development only featured one occlusion between players and none between players and pitch lines, meaning key algorithm decisions had substantial bias. A better approach would have seen more images tested upon the completion of smaller stages.

A simple ball detector could be implemented using the Hough circle transform. This is very similar to the line transform (described in section 2.4.1.1) but uses the circle equation instead. Since the ball is the only expected circle on the pitch, there should only be one clear detection (assuming no occlusions).

In addition, project planning could have been improved. Although a project plan had been created at the start, it was overly optimistic and made unrealistic deadlines. Some stages of development, such as writing the image class, took much longer than anticipated. Consequently, some project milestones had to be pushed back, resulting in less time for parameter optimisation. Training the model parameters would have increased the detector's generality, enabling its use across many more images featuring more challenging scenarios. Since the dataset used didn't include accurate ground truth labels, additional time towards the completion of the project would have allowed some of this data to be manually labelled for training purposes. Potential areas for improvement in the algorithm's structure may have been highlighted during this process too. While the algorithm works well on some images in the dataset, it uses non-optimal parameter values that have a considerable bias towards the match shown in the reference picture used during development.

5.4 Further Work

There are several potential avenues for future work stemming from the work conducted in this project. The algorithm was designed to process individual frames of a broadcast football match. While the algorithm would work for videos (as a video is essentially a sequence of consecutive images), it wouldn't take advantage of any inter-frame techniques, such as background subtraction and feature tracking. Such techniques could be applied to improve detections across several frames, allowing for an offside line to be dynamically tracked throughout a video.

Once a player-tracking approach is implemented, offside offences could be identified too. Offside offences (see section A.2) require knowledge across multiple frames, indicating whether an offside player touches the ball or is involved in active play. The moment the ball is kicked, players currently offside should be identified. Each offside player would then be tracked to determine whether an offside offence has occurred. If an offside player touches the ball or their trajectory moves towards the ball (indicating an attempt to play the ball), then a foul should be given.

A multiple-camera approach could also be adopted, enabling much more accurate localisation of player positions. Multi-camera systems almost completely remove the problem of occlusions, since a player is unlikely to be occluded from all angles. Offside decisions from such as system are likely to be much more accurate – this is more similar to the technology used at the Qatar World Cup, described to in section 2.1.1.

6 Conclusion

This report has explored the various computer vision methods that have previously been used to perform automatic offside recognition. Many approaches have been taken of varying complexity levels, ranging from simple image operations to large neural networks. The selected approach had to be achievable within the confines of a university year and with publicly available image data. More complex deep learning approaches would undoubtedly exhibit higher performance than the model outlined in this paper. However, they would not be viable in the timeframe given and require more labelled training data than was available at the start of the project. On the other hand, a detailed understanding of both computer vision and how offside recognition works has enabled simpler image processing techniques to be pieced together to form the proposed model.

While it cannot be said that the implemented program in its current form is a robust offside detector, since many test images couldn't be processed. Despite this, the algorithm presented has the potential to reliably identify players in offside positions. If the improvements described in section 5.3 can be made, this report is confident the model's performance would increase. Once consistently accurate detections are guaranteed, a low-cost offside detector such as this could feasibly be installed in stadiums across the country, as most football grounds have at least one broadcast camera. This could go a long way to combating the motivation of this project: improving the public's confidence in decisions made by officials to make the beautiful game more enjoyable for all.

References

[1] Lewis Bridgeman et al. "Multi-Person 3D Pose Estimation and Tracking in Sports". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. June 2019, pp. 2487–2496. DOI: 10.1109/CVPRW.2019.00304.

[2] Matija Buric, Miran Pobar, and Marina Ivašić-Kos. "Adapting YOLO Network for Ball and Player Detection". In: *Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods - ICPRAM*. Jan. 2019, pp. 845–851. DOI: 10.5220/0007582008450851.

[3] Computer Vision Notes – Wordpress. *Top hat/White hat*. 2020. URL: https://cvexplained.wordpress.com/2020/05/20/top-hat-white-hat/ (visited on 04/06/2023).

[4] Dennis T. "Confusion Matrix Visualization". In: *Medium* (July 2019). URL: https://medium.com/@dtuk81/confusion-matrix-visualization-fc31e3f30fea (visited on 04/19/2023).

[5] A Ekin, A M Tekalp, and R Mehrotra. "Automatic soccer video analysis and summarization". In: *IEEE Transactions on Image Processing* 12.7 (July 2003), pp. 796–807. DOI: 10.1109/TIP.2003.812758.

[6] Ernst & Young. *Premier League - Economic and social impact*. Tech. rep. Premier League, Jan. 2019.

[7] FIFA. *Goal-line technology*. Nov. 2022. URL: https://www.fifa.com/technical/football-technology/football-technologies-and-innovations-at-the-fifa-world-cup-2022/goal-line-technology (visited on 03/31/2023).

[8] FIFA. *Semi-automated offside technology to be used at FIFA World Cup 2022™*. July 2022. URL: https://www.fifa.com/technical/media-releases/semi-automated-offside-technology-to-be-used-at-fifa-world-cup-2022-tm (visited on 03/31/2023).

[9] Robert Fisher et al. "Hough Transform". In: *Hypermedia Image Processing Reference – The University of Edinburgh* (2003). URL: https://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm (visited on 04/13/2023).

[10] Arne Hasselbring and Andreas Baude. "Soccer Field Boundary Detection Using Convolutional Neural Networks". In: *RoboCup 2021: Robot World Cup XXIV*. Springer-Verlag, June 2021, pp. 202–213. ISBN: 978-3-030-98681-0. DOI: 10.1007/978-3-030-98682-7_17.

[11] Hawk-Eye Innovations. *About*. 2023. URL: https://www.hawkeyeinnovations.com/about (visited on 04/06/2023).

[12] Nasir Jabbar. "Alan Shearer absolutely rips into VAR after Chelsea vs. West Ham controversy, he's spot on". In: *Sport Bible* (Sept. 2022). URL: https://www.sportbible.com/football/alan-shearer-chelsea-west-ham-var-20220904 (visited on 04/21/2023).

[13] javatpoint. *OpenCV Blob Detection*. 2023. URL: https://www.javatpoint.com/opencv-blob-detection#:~:text=Blob%5C%20stands%5C%20for%5C%20Binary%5C%20Large,binary%5C%20objects%5C%20are%5C%20usually%5C%20noise. (visited on 04/16/2023).

[14] Jacek Komorowski, Grzegorz Kurzejamski, and Grzegorz Sarwas. "FootAndBall: Integrated Player and Ball Detector". In: *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP*. Jan. 2020, pp. 47–56. DOI: 10.5220/0008916000470056.

[15] Socret Lee. "Lines Detection with Hough Transform". In: *Towards Data Science – Medium* (May 2020). URL: https://towardsdatascience.com/lines-detection-with-hough-transform-84020b3b1549 (visited on 04/13/2023).

[16] MasterClass. "Hue, Saturation, Value: How to Use HSV Color Model in Photography". In: *Articles – MasterClass* (Sept. 2021). URL: https://www.masterclass.com/articles/how-to-use-hsv-color-model-in-photography (visited on 04/13/2023).

[17] Karthik Muthuraman, Pranav Joshi, and Suraj Kiran Raman. *Vision Based Dynamic Offside Line Marker for Soccer Games*. Apr. 2018.

[18] Filip Öberg. "Football analysis using machine learning and computer vision". MA thesis. Luleå University of Technology, June 2021.

[19] OpenCV. *Changing Colorspaces*. 2023. URL: https://docs.opencv.org/5.x/d9/db0/tutorial_py_colorspaces.html (visited on 04/14/2023).

[20] OpenCV. *Contour Features*. 2023. URL: https://docs.opencv.org/5.x/d9/db0/tutorial_py_contour_features.html (visited on 04/16/2023).

[21] OpenCV. *Contours: Getting Started*. 2023. URL: https://docs.opencv.org/5.x/d9/db0/tutorial_py_contours_begin.html (visited on 04/16/2023).

[22] OpenCV. *Hough Line Transform*. 2023. URL: https://docs.opencv.org/5.x/d9/db0/tutorial_hough_lines.html (visited on 04/06/2023).

[23] Neeraj Panse and Ameya Mahabaleshwarkar. "A Dataset & Methodology for Computer Vision Based Offside Detection in Soccer". In: *Proceedings of the 3rd International Workshop on Multimedia Content Analysis in Sports*. Association for Computing Machinery, Oct. 2020, pp. 19–26. DOI: 10.1145/3422844.3423055.

[24] Konstantinos Plataniotis and Anastasios Venetsanopoulos. *Color Image Processing and Applications*. Digital Signal Processing. Springer Berlin, Heidelberg, Jan. 2000. ISBN: 978-3-642-08626-7. DOI: 10.1007/978-3-662-04186-4.

[25] PNG Egg. *RGB color model*. URL: https://www.pngegg.com/en/png-tbpnn (visited on 04/13/2023).

[26] Premier League. *The history of goalline technology*. Apr. 2013. URL: https://www.premierleague.com/news/60519 (visited on 04/01/2023).

[27] J Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, June 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91.

[28] Guido van Rossum, Barry Warsaw, and Nick Coghlan. "PEP 8 – Style Guide for Python Code". In: *Python Enhancement Proposals* (Aug. 2013). URL: https://peps.python.org/pep-0008/ (visited on 04/16/2023).

[29] Abhishek Sharma. "How to Master the Popular DBSCAN Clustering Algorithm for Machine Learning". In: *Analytics Vidhya* (Sept. 2020). URL: https://www.analyticsvidhya.com/blog/2020/09/how-dbscan-clustering-works/ (visited on 03/19/2023).

[30] Natasha Sharma. "K-Means Clustering Explained". In: *MLOps Blog - neptune.ai* (Feb. 2023). URL: https://neptune.ai/blog/k-means-clustering (visited on 04/05/2023).

[31] Koo Ping Shung. "Accuracy, Precision, Recall or F1?" In: *Towards Data Science – Medium* (Mar. 2018). URL: https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9 (visited on 04/19/2023).

[32] Sideline Soccer. *The History of the Offside Rule*. 2022. URL: http://www.sidelinesoccer.com/history-of-the-offside-rule (visited on 04/17/2023).

[33] Robert Summerscales. "FIFA World Cup Final Beat Super Bowl LVI By More Than One BILLION Viewers In TV Ratings". In: *FanNation* (Jan. 2023). URL: https://www.si.com/fannation/soccer/futbol/news/how-fifa-world-cup-final-beat-super-bowl-lvi-in-tv-ratings#:~:text=FIFA%5C%20has%5C%20claimed%5C%20that%5C%20around,3%5C%2D3%5C%20draw%5C%20in%5C%20Qatar. (visited on 03/27/2023).

[34] Li Sun and Guizhong Liu. "Field lines and players detection and recognition in soccer video". In: *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. Apr. 2009, pp. 1237–1240. DOI: 10.1109/ICASSP.2009.4959814.

[35] The Football Association. *Law 11: Offside*. 2022. URL: https://www.thefa.com/football-rules-governance/lawsandrules/laws/football-11-11/law-11---offside (visited on 04/16/2023).

[36] VerbaGleb – Wikipedia. *RGB 2 HSV conversion with grid*. URL: https://en.wikipedia.org/wiki/File:RGB_2_HSV_conversion_with_grid.ogg (visited on 04/13/2023).

[37] Md. Zubair. "A Comprehensive Guide on Color Representation in Computer Vision". In: *Towards Data Science – Medium* (Apr. 2023). URL: https://towardsdatascience.com/how-color-is-represented-and-viewed-in-computer-vision-b1cc97681b68#d7ce (visited on 04/13/2023).

A The Offside Rule

Offsides are an quintessential part of modern footabll, having been first introduced in 1863 [32]. The idea behind the rule is to discourage attacking players from 'goal-hanging', where an attacker waits behind the defence for their team to play a long ball to them, providing them an easy 1-v-1 opportunity to score against the goalkeeper. For an attacker to legally touch the ball behind the defence (or be involved in play in general), they cannot have been in an offside position before the ball is passed.

A.1 Offside Position

The offside position itself is not an offence, but rather describes whether the position of an attacker could constitute an offside offence. A player is in an offside position if both of the following conditions are met:

- They're is in the opposing team's half.
- They're closer to the opponent's goal line than both the ball and the second-last opponent.

The part of the player's body chosen for offside detection is the body part closest to the goal, excluding their hands and arms (since outfield players cannot touch the ball with these).

A.2 Offside Offence

An offside offence is only be committed when a player standing in an offside position the moment the ball is played becomes actively involved in the game. This includes:

- Touching the ball.
- Attempting to touch the ball.
- Preventing an opponent from being able to play the ball (by obstructing their vision or challenging for the ball).

If an offside player receives the ball from an opponent who has deliberately played the ball, then they aren't considered offside [35].