A

IV Semester

MINOR PROJECT REPORT
ON

"FOOTBALL - OFFSIDE LINE DETECTION"

BY

ANKIT PAUL	18100007
ANNAPRAGADA NISHANT	18100008
RAHUL NAGWANSHI	18100044

UNDER THE GUIDANCE OF
PROF. SANTOSH KUMAR

DEPARTMENT of Computer science engineering
Dr. SPM INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY NAYA RAIPUR
ATAL NAGAR - 493661, INDIA

Jul 2020

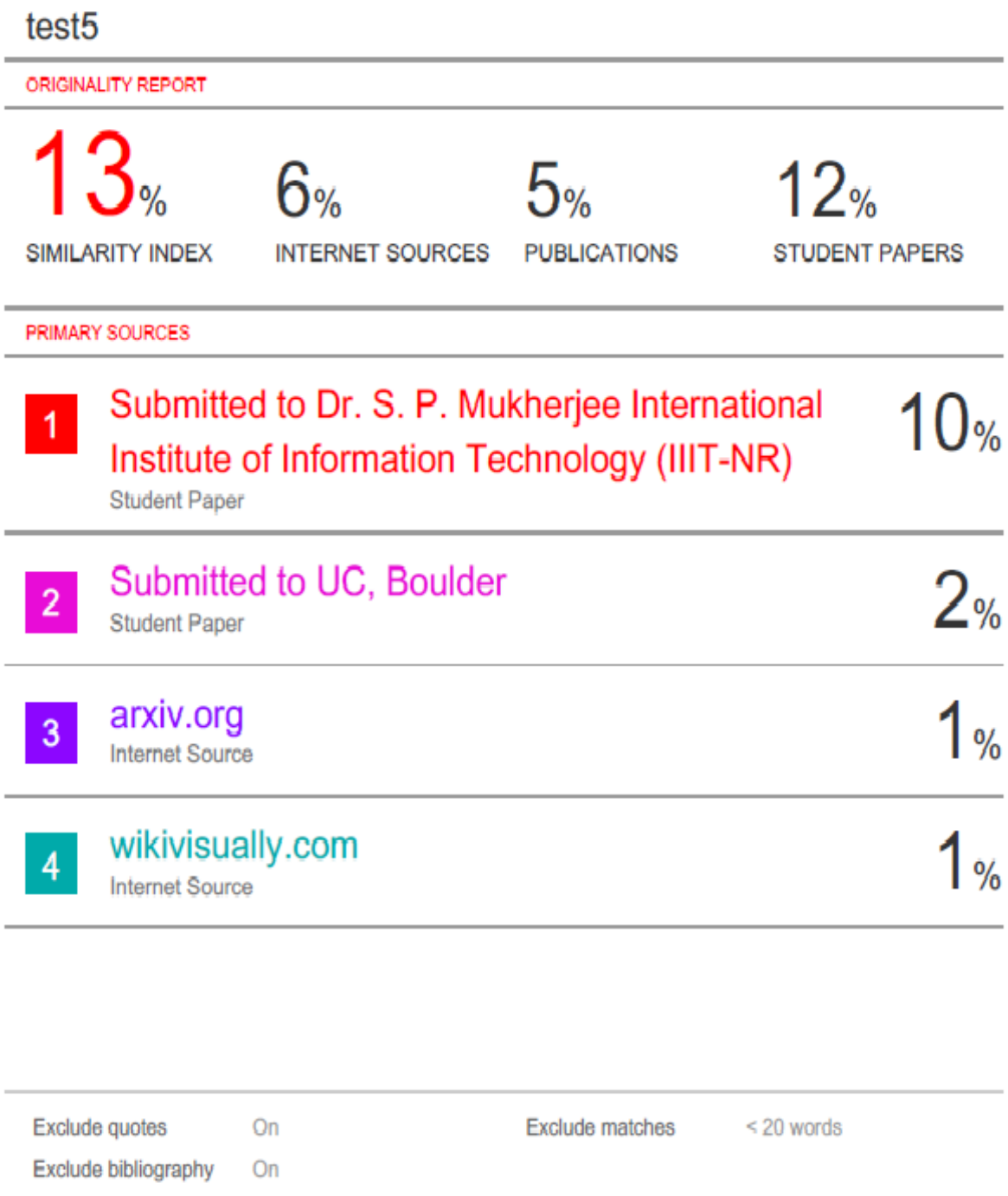
Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Dated: 02/07/20

Ankit Paul
Rahul Nagwanshi
Annapragada Nishant

PLAGIARISM REPORT



test5

ORIGINALITY REPORT

13%	6%	5%	12%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Dr. S. P. Mukherjee International Institute of Information Technology (IIIT-NR) Student Paper	10%
2	Submitted to UC, Boulder Student Paper	2%
3	arxiv.org Internet Source	1%
4	wikivisually.com Internet Source	1%

Exclude quotes	On	Exclude matches	< 20 words
Exclude bibliography	On		

Certificate

This is to certify that the project titled “Offside Line Detection” by “Ankit Paul, Annapragada Nishant, Rahul Nagwanshi” has been carried out under my/our supervision and that this work has not been submitted elsewhere for a degree/diploma.

Dated: 02/07/20

Dr. Santosh Kumar
Department of CSE
DSPM IIIT Naya Raipur-493661

Acknowledgments

At the outset, we would like to express my whole hearted and deep sense of gratitude to my guide Prof. Santosh Kumar for his guidance, help and encouragement throughout our project. I greatly admire his attitude towards research, creative thinking, hard work and dedication in work. I am highly grateful to him for patiently checking all my manuscripts and thesis. This thesis would not have been possible without his bounteous efforts. More than guide, he is my mentor for shaping my personal and professional life, without whom I would not have been where I am today. I owe my profound gratitude to Prof. Santosh Kumar for his supports in all respects.

Ankit Paul

Annapragada Nishant

Rahul Nagawanshi

ABSTRACT

Offside is the most complicated rule in football. On an average, a game of football has around 50 offside detections. Therefore, false offside detections cannot be neglected. Current offside decisions are determined manually by the linesmen running on the side of the field. These judgements may not be accurate and are prone to errors because occlusion between the players and afterimage may confuse the linesman and the referee. These false detections and rash calls tend to remain controversial and drastically affect the outcome of the game. Wrong decisions can lead to quarrels among players and supporters. This work proposes an automatic linesman assistance system to assist accurate refereeing for football-offside detection using computer vision and image processing techniques.

Contents

List of Figures

Chapter 1

Introduction

Offside rule, as defined by English Football Association "A player is in an offside position if: any part of the head, body or feet is in the opponents' half (excluding the halfway line) and any part of the head, body or feet is nearer to the opponents' goal line than both the ball and the second-last opponent". Conventionally, offside decisions are taken by linesmen, who cover the whole game by moving along the sidelines. Offside calls remain highly controversial and one of the most complicated rules in any game. These kinds of decisions have to be made at a split of a second which requires linesmen's utmost attention at the very moment which they miss most of the time affecting in game conditions resulting in quarrels among the players, supporters and spectators which drastically change the outcome of the game. Computer vision can be used very effectively for accurately detecting offside.

We attempt to do this sequentially in this paper. We read a video frame by frame.For every frame, we detect player and segment them into their respective teams. We track all the players on the pitch and keep track of the last player on the defending side. The offside line is a dynamic line which is made through the last player of the defending side.

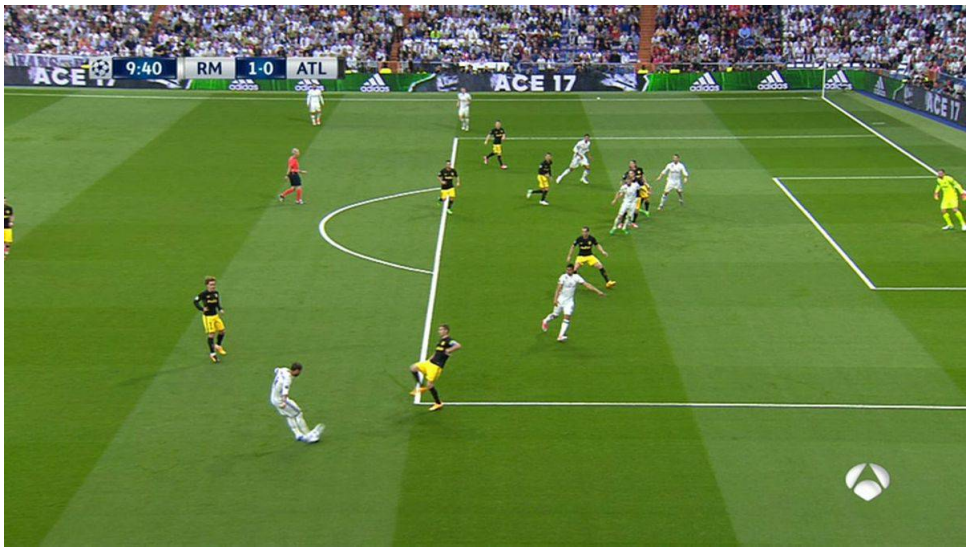


Figure 1.1: Offside

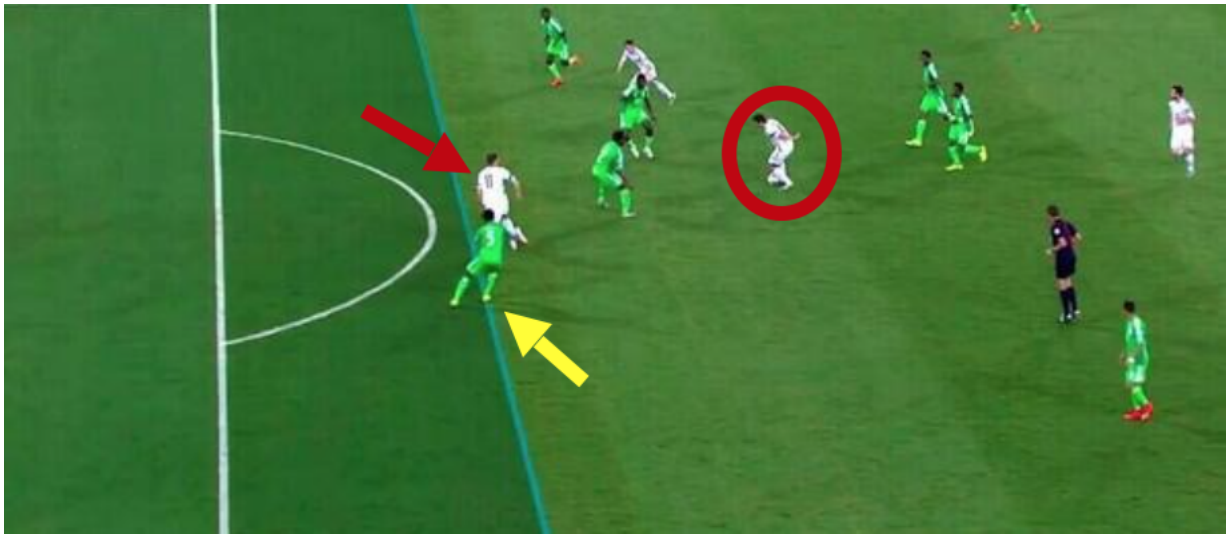


Figure 1.2: Not Offside

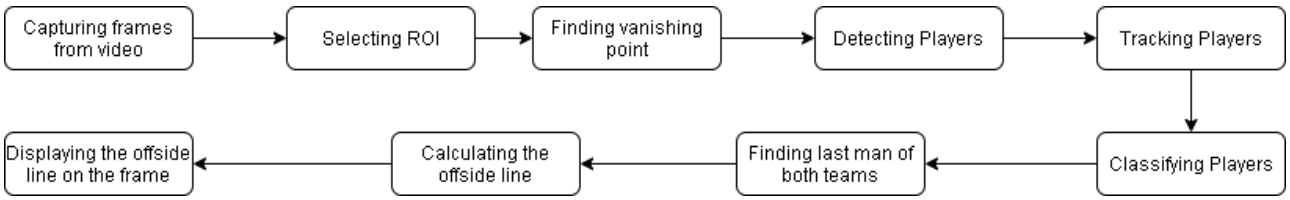


Figure 1.3: Sequence Chart

Chapter 2

Literature Review

A significant amount of work has been done in the field of football match video analytics in general. Player tracking, segmentation, Offside detection systems, On-screen virtual Offside line markers and above all the Video Assistant Referee (VAR) system being employed by FIFA in the premier tournaments. All the work done, except VAR, has addressed various specific problem statements with certain assumptions and limitations. Parth et al.3 have proposed a method to detect offside which takes information of the team jerseys and the marks ground's endpoints, during execution of the program as as inputs.This method assumes that the camera is absolutely still, which is rather impractical considering actual broadcast coverage of a game. Also, errors in offside detection, are subject to camera and video quality,as players in the frame may be running, causing them to be blurred in the video.

Jagjeet, et al. [4] have proposed a automatic method to detect offside which works on still images. They have used the pretrained CNN, MobileNet for player detection. It does not work well for all different kinds of videos which questions robustness of the system. It also requires high computational capacity, since Deep learning models are being used.

Another reference (et al.[2]), a paper by students of Michigan University, used Open CV and the YOLO(You Only Look Once) CNN trained on MS-COCO dataset, to detect players. It just draws an Offside line and leaves it to the user to view and take decision, which is a better alternative than an automated decision, subject to defects of the camera. Also, the system, can operate well even for a video with a moving camera, making it better suited for the purpose.

All, the above references processed TV broadcast footage. But FIFA, the governing body of Football (et al.1]), recently introduced Video Assistant Referee system (VAR), which enables referees to review decisions on fouls as well as off-sides, based on videos obtained from high- resolution, powerful cameras placed all around the stadium. This system has brought about a revolution in refereeing of matches. This system, marks offside lines on the video assistant referee's screen, provides slow motion videos from different angles, enabling him to review on field referee's decisions. But this technology is not open-source.

Chapter 3

Proposed Solution

In this project, we have used some clips of real time plays, with a standard broadcast camera. The main obstacles of this project are varying camera movements, different stadiums and different coloured kits for every team. This project, gives an offside line for every frame by using certain image processing techniques and player detection. It is tremendously important to accurately detect the play area and players. An ideal offside line detector should work perfectly fine under any camera movements, illumination etc.

3.1 Determining Play Area

We need to find our Region of Interest (ROI), for effectively detecting players and also detecting offside line. ROI is the part of data set which is useful for our purpose. We have our input video, which is a standard broadcast video. We use background subtraction to filter out the ROI. Background detection can also be called as foreground detection.For color filtering, we convert the default RGB space to HSV (Hue saturation value) color space. Converting to HSV has the advantage of detecting any specific color background from specific color codes, which we specify in the code. HSV, will further be helpful when segregating the nationalities of the players. It is actually a type of color plane representation (like RGB,). It is a representation format independent of device: The HSV colour representation is useful to detect specific color types(in our case, range) To find ROI, we separate out the green color from the frame (color filtering). We provide a low and a high range for green value, which separates the field from audience. Bitwise-and operation is specifically used for creating the mask of field(green colour). Bitwise operations are used in openCV to extract useful parts of the image.

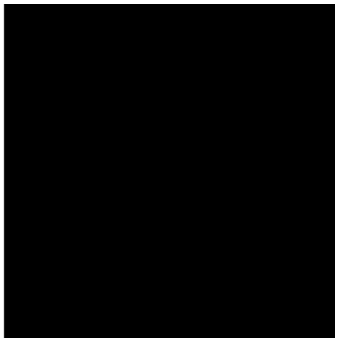


Figure 3.1: Source1

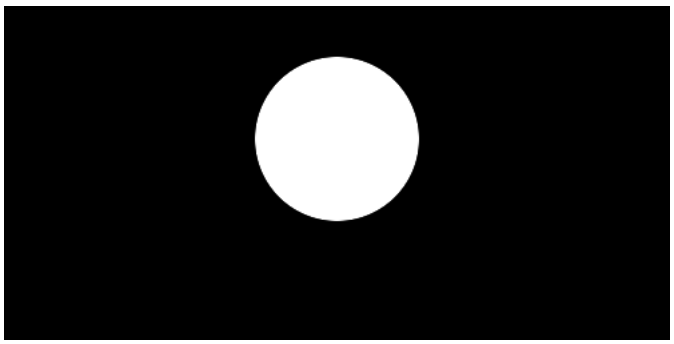


Figure 3.2: Source2

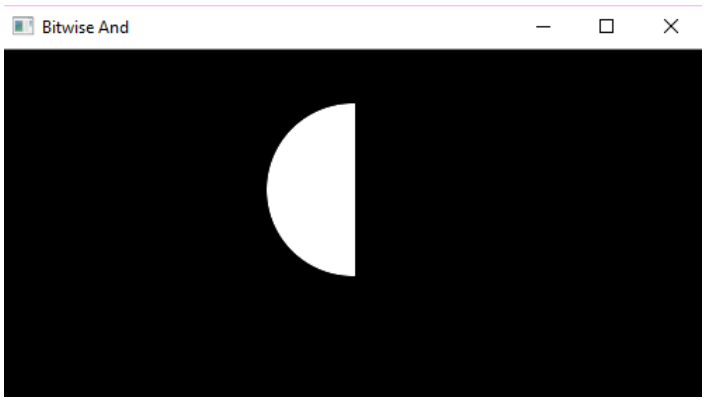


Figure 3.3: Bitwise operation performed on two source images

We also perform morphological operations on the images.It is performed, generally, upon binary images.Specifically, we apply closing and opening operations. Morphological operations are always applied in pairs. Opening operations discards the pixels near the boundary depending upon the size of the kernel. It is advantageous for removing small white noises. The opening operation fill out the audience, since some error might be present there(If someone in the audience is wearing green color, he/she might be detected). Closing operation, as it sounds, is the exact opposite of opening. It increases the white region in the image. Opening is always followed by closing, because opening removes white noises, which decreases area of the image. Now we have removed the noise, so closing is applied to retain the size of the original image. Thus, we have detected the play area effectively. After

separating play area, we can effectively detect and track players. Color filtering just keeps the green color and turns rest into black.

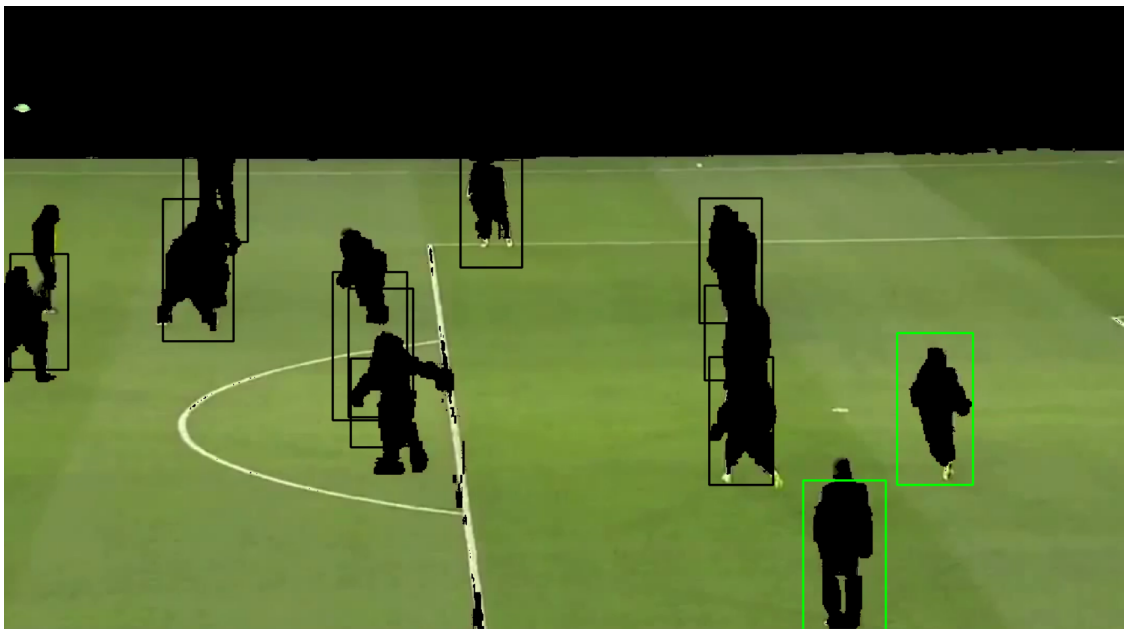


Figure 3.4: After separating ROI

Figure 3.5: After applying morphological operations

3.2 Player Detection and tracking

Now, since we have generated the region of interest. We can detect the players in a similar way. HOG descriptor is used to detect the people in the frame. HOG(Histogram of Oriented Gradients).
HOG is a feature descriptor. ”A feature descriptor is a representation of an image or an image patch that simplifies the image by extracting useful information and

throwing away extraneous information".Typically, a feature descriptor converts an image to array of length n. Shape, as we know is the most important characteristic while categorizing objects. HOG uses exactly the same concept. We used the built-in HOG + Linear SVM detector that OpenCV comes with, allowing us to detect people in images. HOG descriptor creates histogram of each gradient orientation in the specific part of the image. The directional changes in color is referred as gradient orientation.
HOG feature extractor can be used in following ways

We resize the image to make our code run faster

We convert the image to a particular color space

Using HOG(which is hog() in OpenCV

SVMs find a hyperplane in N dimension al space, to classify data points. Dimension of the hyperplane calculated, depends on the number of features present. We specifically used Daimler People Detector, since it is specifically trained to detect people. But the dimensions of the image should be 1:2, which makes the calculations significantly faster. We throw all the results under a threshold to only detect players on the field.In our case, its the area and height of the players, since the players detected on pitch must be quite large enough(comparing to the audience). Bounding boxes are cropped around the players. We have the range of jersey colors. We can separate out the two teams,easily based on their jersey color. We again apply color filtering. Before applying color filtering, we once again convert RGB to HSV. Thus, to remove small minuscule components we used an image opening morphological operation followed by image dilation to increase the accuracy of player detection. Noise inside the jersey can be in the form of sponsor's stickers. We apply color filtering for both the teams. Separate color bounding boxes are used to store the coordinates of players for both teams(which we found using HOG descriptor). We keep two arrays where we store the coordinates of the detected players of the respective teams. Since we need to know the real time information for all the detected players, we detect players every frame. Like this, we can store the coordinates of players at every frame, in their respective team's array. Using all these information, we can now proceed to detection of the offside line.

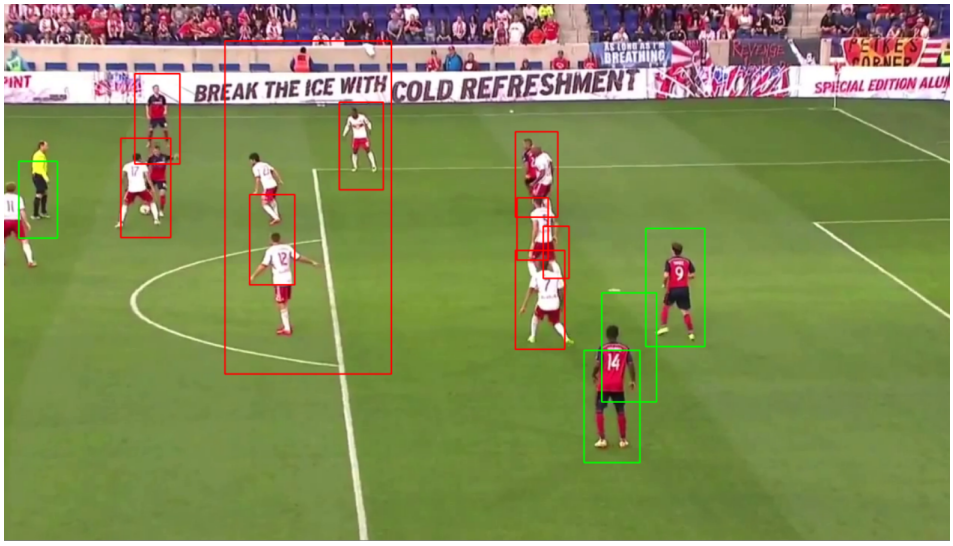


Figure 3.6: Separate coloured rectangles to separate the nationalities

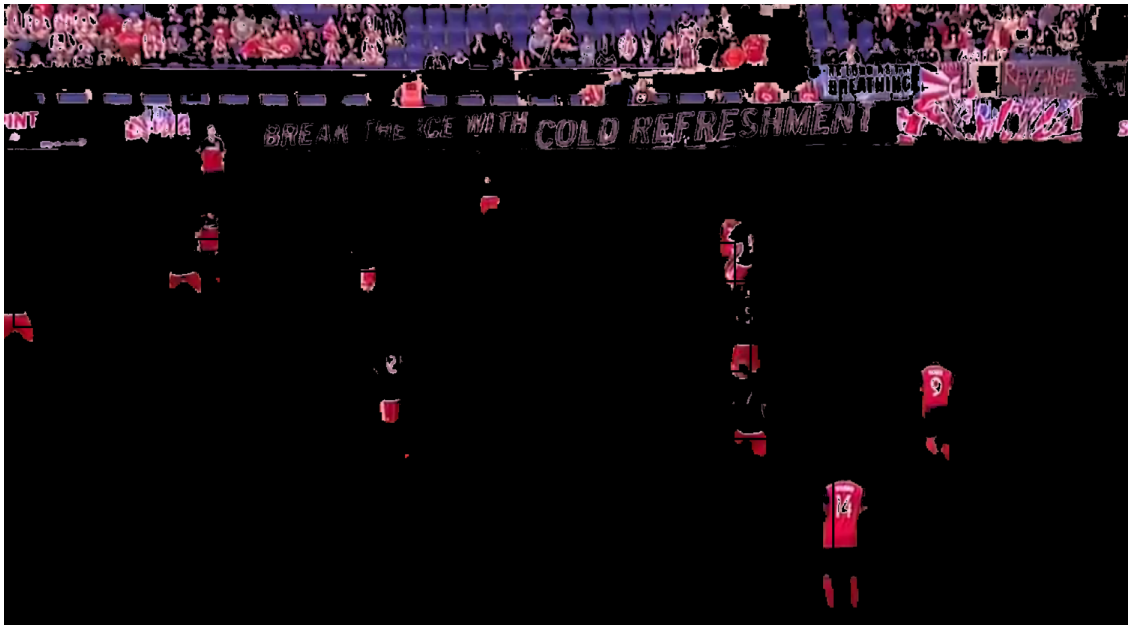


Figure 3.7: Extracting team2's players

Figure 3.8: Using opening and closing morphological operations to remove noise from crowd and as well as inside the player model

3.3 Drawing the Offside Line

Once detecting the players present on the field area, we now have to detect the offside line. We know that the offside line must pass through the player which is closest to the goal line on the defending side, the other point which this line should pass decides the orientation of the offside line which should be parallel to the vertical boundaries of the field with respect to the angle of the camera. We use the existing lines i.e, the penalty line and the goal line to determine the vanishing point. Vanishing point is a point on 2D plane of where two mutually parallel lines in 3D space seems to converge. For example, when we view a rail track from one end, it seems to converge after some distance.

We detect lines on the image by using CannyEdge and HoughLineP(Probabilistic Hough Lines transform) transform functions. Hough Lines Probabilty function directly gives us the coordinates of the lines. To apply the Transform, first we use canny edge as an edge detection pre processor.

For edge detection, we first convert the image colorspace from BGR colorspace to gray colorspace. Then we remove noise by convolving it with a low pass filter kernel, this process is called image smoothing (Blurring). We use gaussian filtering for this purpose as this is highly effective for removing noise from the image without getting the edges blurred.

We apply the Canny edge algorithm to these blurred images, the canny edge algorithm is a multistage algorithm. This algorithm first removes the noise from the image, noise reduction is followed by determining edge gradient. After a non-maximum Suppression, Hysteresis Thresholding is applied to finally detect strong edges in the image. Each stage of the algorithm is described here in detail.

1) Noise minimization : First stage involves noise reduction as edge detection

is highly sensitive to noise. It is done using a 5x5 Gaussian filter.

2) Determining edge gradient : To find direction and edge gradient for each pixel, we filtered smoothened image with a Sobel kernel in both vertical and horizontal direction to get the first derivative in vertical direction and horizontal direction. Gradient is rounded to one of four angles representing the two diagonal directions, horizontal direction and vertical direction.It is always perpendicular to the edge.

3) Non-max Suppression : To eliminate any undesired pixels which may not represent the edge, a full scan of the image is done after getting gradient direction and magnitude. For this, a pixel is checked if it is a local maximum in its vicinity in the direction of gradient at every pixel. If not, it is suppressed, otherwise, considered for the next stage. At last, we get "thin edges" as a binary image.

4) Hysteresis Thresholding : This stage confirms which of all edges detected in the previous stage are genuinely an edge. For this, we take minimum and maximum threshold values. Those edges who are above the maximum threshold value are confirmed as edges and those who are below minimum threshold values are not confirmed edges. Those who lie in between the threshold values are decided as an edge or not an edge based on their connectivity. If they are connected with an edge which is marked as a sure edge, they are regarded to be part of the edge and otherwise discarded. Therefore, it is very important to select threshold values accordingly to get accurate results. This stage works on the assumption that edges are long lines and eliminates tiny pixel noises. In this way, we detect strong edges in the images.

Next, after edge detection, we have to detect lines out of edges. For this, we used HoughLineP Transform function as it is able to detect any shape if it can be represented in a mathematical form. We know that there are various representation of a line, openCV uses parametric representation as = xcos + ysin where denotes the perpendicular distance of the line to the origin and denotes the angle between this perpendicular and horizontal axis measured in counterclockwise direction. Any line can be represented by these two values (,). HoughLineP generates a 2D accumulator or array initialized to all zeroes where rows denotes and columns denotes . The size of the array depends on the accuracy of the result one requires. Maximum value can be the diagonal length of the image and the maximum value can be 180. For any line present in the image, We consider its first point (x, y) putting it in the equation with = 0,1,2,3.....,180 and get the corresponding values. For every pair (,) we increment the value by one in its corresponding cell. We repeat the same procedure for every point on the line. This way some cells are voted more by the rest and the

position of the maximum value in the accumulator represents the line by the pair (,), which says line is present at the distance of with an angle of with the horizontal axis. In this way, we detected the lines present in the image.

We use the line with the maximum length as the penalty line and the line closest with the right boundary as the goal line.

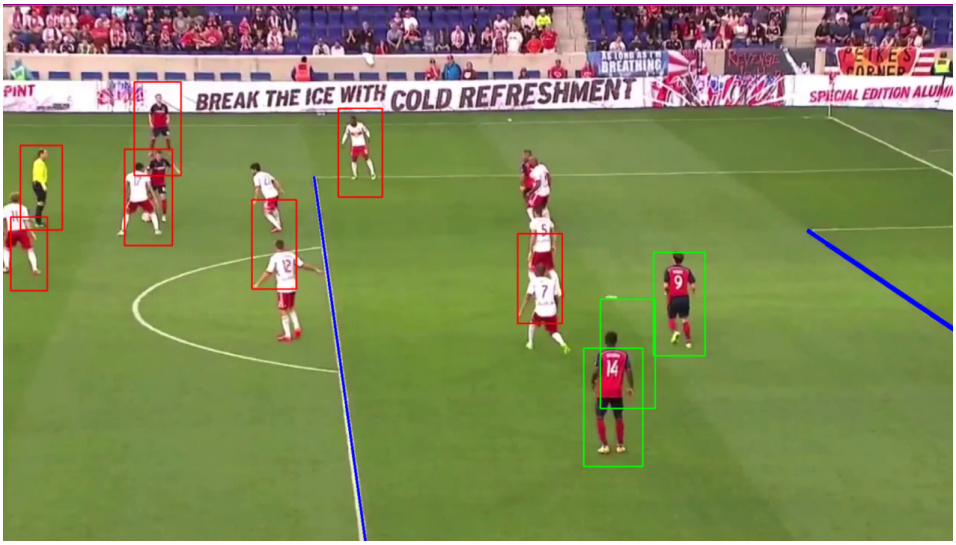


Figure 3.9: HoughLinesP used to detect 16 m and 5 m lines

After detecting, we extend these lines to compute the intersecting point as the vanishing point.

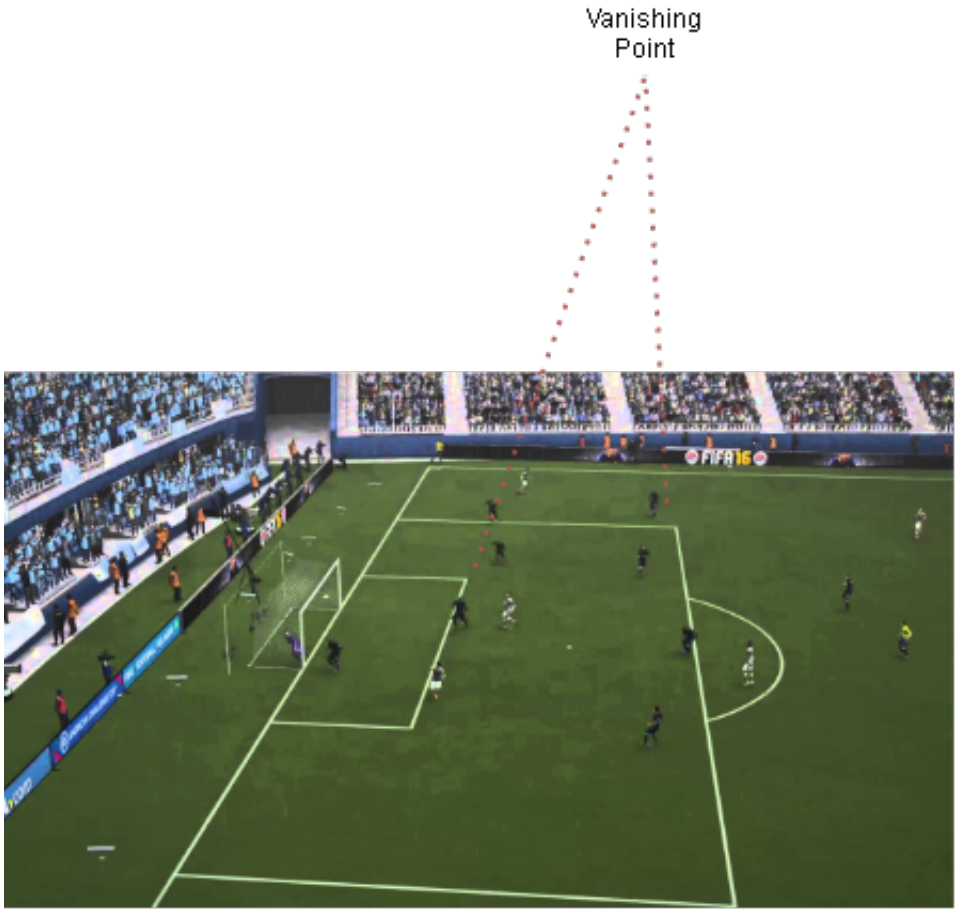


Figure 3.10: Vanishing point(intersection of 16 m and 5 m lines)

We, then extend a line from ”vanishing point” to every defender and then choose the line with the maximum slope. This way we determine an accurate line which can be used as an Offside line.

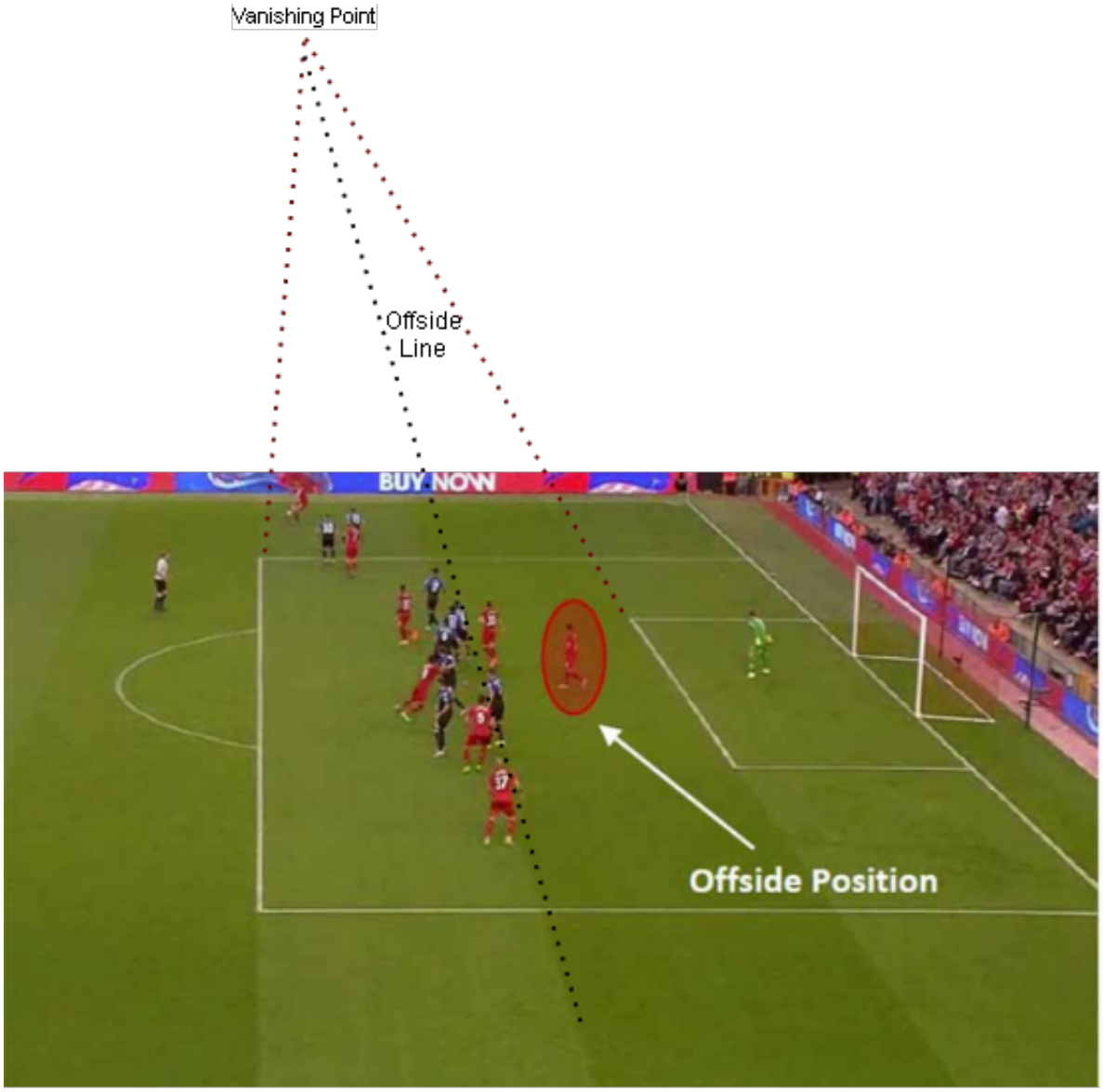


Figure 3.11: Offside Line drawn from vanishing point

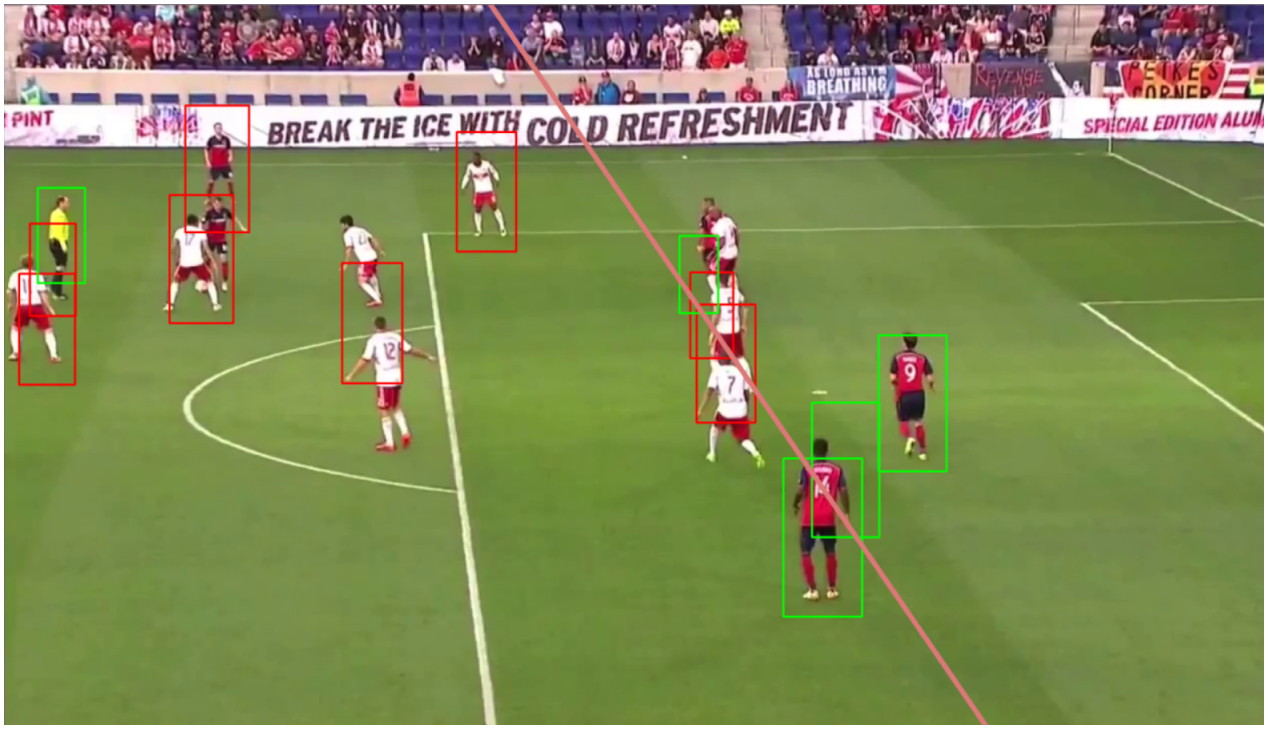


Figure 3.12: Offside Line drawn from vanishing point

3.4 Displaying the Top View

Our system also develops the top view(2D) of the game, with all players on the pitch, marked correspondingly to the pitch, we are displaying upon. The top view, further makes the offside detection easier for the referee. Also, if we are able to generate the top view of the field, calculation of many other things, such as heatmaps, pass detection, and other statistics become quite easy. Of course, all of those is beyond the scope of our project. In our case, it further simplifies detecting offside. To generate top view, first thing we need to realize is that one plane(football field) is stretched out and our pitch image is ”laid flat”. We therefore, need some kind of transformation to map the corresponding coordinates. We need homographic transformations, that can change a square into any kind of quadrilateral.We would be needing this transformation.

Mathematically, homographic transitions, can be represented as-

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \mathbf{H} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$$

Figure 3.13

x,y represents pixel coordinates.(x’,y’) represents pixel coordinates in another

plane. And H is the homography matrix.

$$\begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix}$$

Figure 3.14: Homography Matrix

If there is a point (x,y) on our field. We can easily get the coordinates on the 2D pitch, by multiplying (x,y) with H, which gives us (x',y'). We can now, easily plot the points on the pitch.To calculate H matrix, we need at least 4 points. But generally speaking, more the points, more accurate the result will be.

Figure 3.15: Point on a chess board

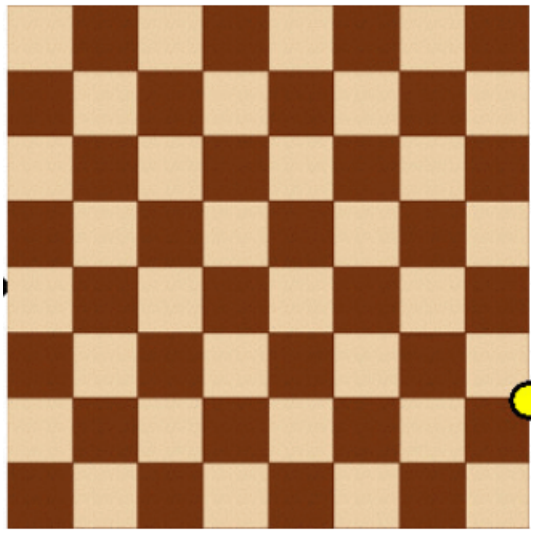


Figure 3.16: Point marked on 2D plane

We used GIMP to get the coordinates of the fields."GIMP is a free and open-source raster graphics editor used for image retouching and editing, free-form drawing, converting between different image formats, and more specialized tasks".GIMP

can be used to get the mouse coordinates quickly and easily. Finally,we have exploited OpenCV's library functions to get this done, because OpenCV provides all the necessary methods for our problem.findHomography() does this job for us by converting the coordinates on the field to the 2D pitch.

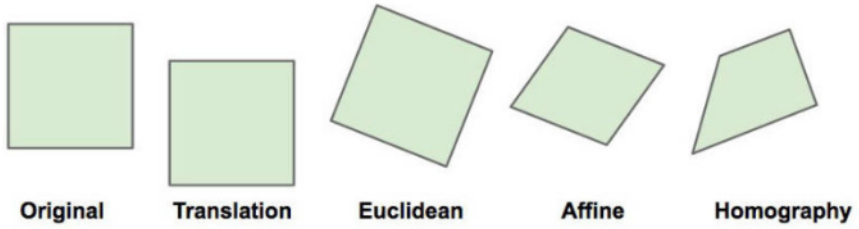


Figure 3.17: Different types of transformations

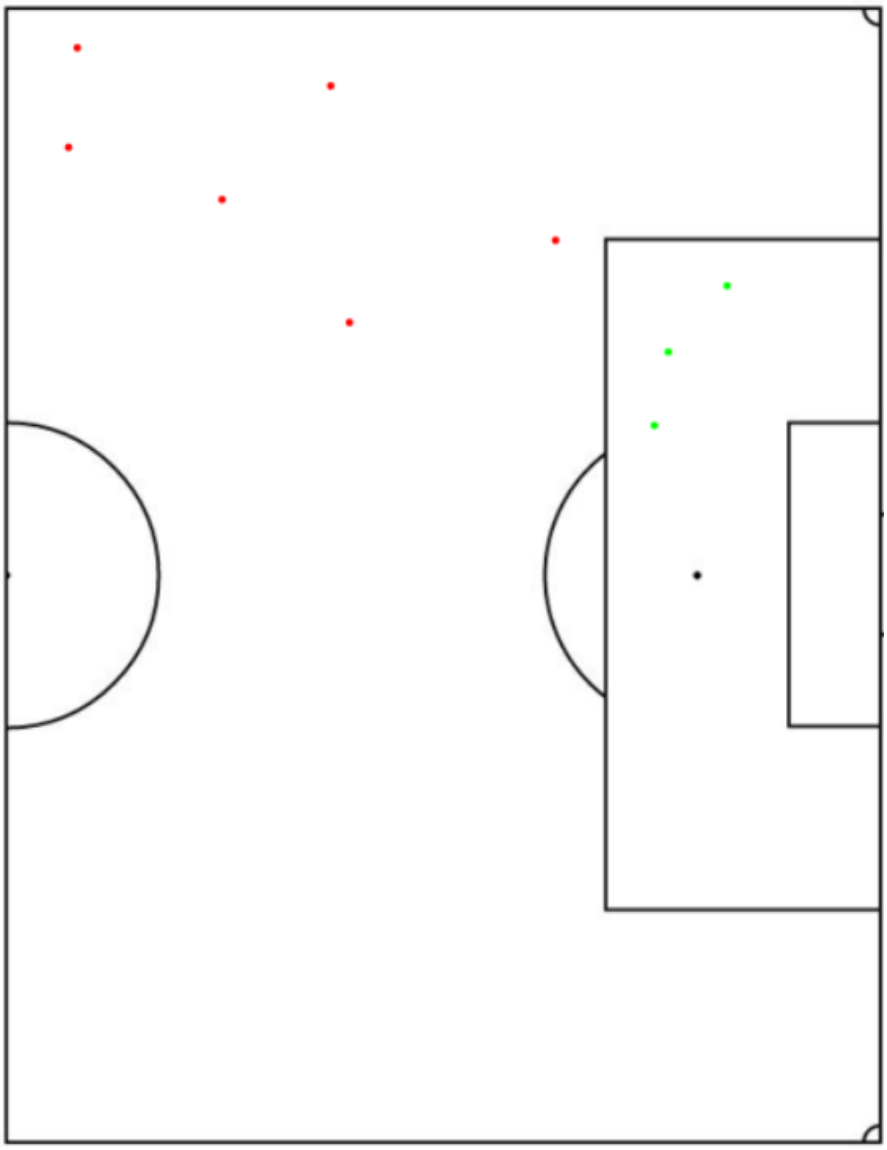


Figure 3.18: Top view of the field

Chapter 4

Results

Following our methods above, we now report the stepwise results after each operation.Figure 1.1 through figure 3.5 illustrate every step of our algorithm in detail. Same method is used in figure 3.6 through 3.8 to separate the two team's players , by applying different masks. Separate colour bounding boxes are used to separate the players of attacking and defending side. By using our offside detection algorithm, we calculate the final offside line in figure 3.12. Our method very efficiently works,even when audience is present, since it removes the noise very effectively.
In figure 3.18, we have successfully generated the top view of the field(by converting the coordinates on the field to map it on the 2D pitch), with the attacking and defending side players marked in different colours.

Chapter 5

Conclusion and Future Scope

In this project, we have developed and implemented an algorithm which:-

- Detects and tracks players in a football match broadcast video using HOG Descriptors and SVM Classifier.
- Classifies players into their respective teams based on colour.
- Draws the offside line in the broadcast video itself by finding the vanishing line through Hough lines.
- Projects the positions of the players on a virtual football pitch, from a top view, through homographic transformations. This simple, yet effective, project can help referees take the right decisions regarding the question Off-sides, with the help of On-screen Offside lines. But it can also prove to be quite helpful for teams to analyse their style of play and tactics, through the features of player racking and their position projections on the pitch.
- Our success in getting in the position of players from the top view, further enables us to make heatmaps of individual players or even track the passes.
- Other improvements would be improving accuracy of tracking system further.

A lot of work is presently going on the application of computers in the field of sports, something we are quite interested in. This project is an honest effort to learn and contribute to the same.

References

[1] https://football-technology.fifa.com/media/171902/var-iaap-technology-tests.pdf

[2] Karthik Muthuraman,Pranav Joshi,Suraj Kiran Raman,Vision Based Dynamic Offside Line Marker for Soccer Games,University Of Michigan

[3] https://github.com/kparth98/ITSP-Project

[4] https://github.com/jagjeet-singh/Vision-based-method-for-offsidedetection-in-soccer

[5] Ming Xu, J. Orwell and G.Jones, *Tracking football players with multiple cameras*Image Processing, 2004: 2004 International Conference

[6] Joseph Redmon, Santosh Kumar Divval, Ross B. Girshick and Ali Farhadi,*You Only Look Once: Unified, Real-Time Object Detection*,CoRR 2015

[7] `https://github.com/AndresGalaviz/Football-Player-Tracking`

[8] D'Orazio, T., Leo, M., Spagnolo, P., Mazzeo, P., Mosca, N., Nitti, M., Distante,*An Investigation into the Feasibility of Real-Time Soccer Offside Detection from a Multiple Camera System*,2009: IEEE Transactions on Circuits and Systems for Video Technology

[9] H., O., J.J., M.,*Player position detection system*,2001: Google Patents