AUTOMATIC OFFSIDE DETECTION

A PROJECT REPORT

Submitted By

RUPESH A. K. 312216104088

PRASHANT KUMAR DIXIT 312216104078

VIBHUTI KUSHWAHA 312216104120

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

SSN COLLEGE OF ENGINEERING

KALAVAKKAM 603110

ANNA UNIVERSITY :: CHENNAI - 600025

May 2020

ANNA UNIVERSITY : CHENNAI 600025

BONAFIDE CERTIFICATE

Certified that this project report titled **'AUTOMATIC OFFSIDE DETECTION"** is the *bonafide* work of "**RUPESH A. K. (312216104088)**, **PRASHANT KUMAR DIXIT (312216104078)**, and **VIBHUTI KUSHWAHA (312216104120)**" who carried out the project work under my supervision.

Dr. CHITRA BABU
HEAD OF THE DEPARTMENT
Professor,
Department of CSE,
SSN College of Engineering,
Kalavakkam - 603 110

Dr. K. LEKSHMI
SUPERVISOR
Associate Professor,
Department of CSE,
SSN College of Engineering,
Kalavakkam - 603 110

Place:
Date:

Submitted for the examination held on. . . . . .

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENTS

We thank GOD, the almighty for giving me strength and knowledge to do this project.

We would like to thank and deep sense of gratitude to our guide **Dr. K. LEKSHMI**, Associate Professor, Department of Computer Science and Engineering, for his valuable advice and suggestions as well as his continued guidance, patience and support that helped us to shape and refine our work.

My sincere thanks to **Dr. CHITRA BABU**, Professor and Head of the Department of Computer Science and Engineering, for her words of advice and encouragement and I would like to thank our project Coordinator **Dr. S. Sheerazuddin**, Professor, Department of Computer Science and Engineering for her valuable suggestions throughout the project.

We express our deep respect to the founder **Dr. SHIV NADAR**, Chairman, SSN Institutions. We also express our appreciation to our **Dr. S. SALIVAHANAN**, Principal, for all the help he has rendered during this course of study.

We would like to extend our sincere thanks to all the teaching and non-teaching staffs of our department who have contributed directly and indirectly during the course of our project work. Finally, I would like to thank our parents and friends for their patience, cooperation and moral support throughout our life.

RUPESH A. K. **PRASHANT KUMAR DIXIT**

. **VIBHUTI KUSHWAHA** .

ABSTRACT

In offside detection for the duration of a football match, afterimage and occlusion confuse the linesman and the referee, and this confusion can reason judgment errors. This study proposes a technique of football-offside detection the use of one or more cameras in an automatic linesman assistance system. In this method, a static camera is hooked up as a hardware device, and offside will be alerted while the offside detection unit detects offside. In an offside detection unit, when the offside role is decided, offside is straight away judged inside the case wherein the ball is surpassed from an attacker to some other attacker who is positioned closer the opponent-goal line than the most-back defender and the ball. The vertical lines that constitute those attackers, defenders and ball are used as their positions for logically judging the offside. Experiments had been performed the usage of short videos from 3 different plays. The consequences confirmed 95% accuracy in comparison with the professional.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Offside in Football

The offside law is précised by way of FIFA. The law states that if a participant is in an offside position when the ball is played by means of a teammate, he/she may additionally no longer become actively involved in the play. An offside position is taken when the participant is nearer to the goal line than both the ball and the second to final member of the defensive team. The foregoing statement solely applies when in the opposition half of the field. Offside is a specific rule defined by the association of football, which states that a player is in an offside position if any of their body part except the hands and arms is in the opponents half of the pitch and closer to the opponents goal line than both the ball and the second-last opponent.

1.2 Motivation

Offside detection in football has emerged as one of the most important decisions with an average of 50 offside decisions every game. Most of football matches end with a close score so every decision counts. The offside rule in football is one of the most controversial throughout football matches. Errors when judging an offside position are very common and they are always attributed to human causes. Numerous scientific papers have contributed in a very necessary way to

comprehend why these errors happen. The human error has been related to optical errors based on the incorrect angle of view, and eye movements. A human error means to carry out incorrectly, an action we are physiologically qualified for. The trouble is to find if the error has a human origin or not. In our project, our dataset has been taken manually by taking videos of real-time players playing in the football ground. Three separate videos were recorded where we took different cases of offside rule.

1.3 Statement of the Problem

Offside decisions are one of the most controversial decisions in the game of football.It is currently handled by human referees. Being handled by humans makes the process instantaneous but is prone to a lot of false decisions. Football being a game of very close margins makes the decisions very more important and relevant to a game. Hence the project demands to create an automated system capable of making correct offside decisions in negligible amount of time.

1.4 Goals of the study

The key goal of this exploration is to provide referees with objective and accurate decision of offside rule detection. Traditionally offside are called by the human sideline referees who keep moving along the sidelines continuously tracking the last defender. Not only is the work of the sideline referee taxing, but this technique is prone to human errors at multiple levels due to the limited precision

of human eye. Even though there are definite rules explaining the foul, offside call decisions are determined at the split second. Offside calls remain highly subjective to human referees and remains as one of the most complicated rules in all of sports. Technology in sports has paved way for lot of improvements with respect to increased quality, accuracies and better decisions across multiple sports right from ball, player tracking to virtual simulations. Computer vision algorithms can be exploited efficiently for this problem of offside detection. Specifically in this problem of offside detection, techniques from Computer Vision can be used effectively to develop an end to end system of automatically determining offsides or can be used in assisting referees in making educated calls in almost real time. In this work, we present an assistive technology to determine the offside line marker, in all frames of the real time video feed which could help in better decision making and accurate offside calls. Some of the previous approaches are GPS based approach and RF based approach in which GPS chips were mounted on the jerseys and studs of the players to obtain the co-ordinates of the players and then compute their distance from each other or RF signals were used to determine the distances of players from reference points. The current technology being used is Video Assistant Referee (VAR) in which assistant referee reviews decisions made by the head referee with the use of video footage. There is still ongoing research in detection of offside using image processing.

1.5 Objective

The objective of this project is to create a working offside detection system that can accurately detect offside passes. This system will rightfully evaluate the position

of players and the football to provide an objective analysis of offside passes. By taking the images frame by frame from a video in a sequential manner will analyze the ball trajectory and consequentially the offside pass.

1.6 Outline of the study

In this project, we process the video frame by frame and apply image processing techniques like background subtraction, contour detection, histogram back projection to identify the players, the ball and which of the two teams each player belongs to. After detecting players and the ball, the passer (player who passes the ball) and the receiver (player who receives the ball) are identified. Then rules are applied to find out whether this particular situation is offside or not after applying perspective view transformations.

CHAPTER 2

REVIEW OF LITERATURE

2.1 Background Subtraction

2.1.1 A Comparison between Background Modelling Methods for Vehicle Segmentation in Highway Traffic Videos By L. A. Marcomini, A. L. Cunha [1]

The objective of this paper is to compare the performance of three background-modeling algorithms in segmenting and detecting vehicles in highway traffic videos. All algorithms are available in OpenCV and were all coded in Python. Seven videos were analyzed. To compare the algorithms, 35 ground- truth images were created, five from each video, and three different metrics were used: accuracy, precision rate, and processing time. By using accuracy and precision, the aim was to identify how well the algorithms perform in detection and segmentation, while using the processing time to evaluate the impact on the computational system. The paper's objective is to compare the performance of three background-modeling algorithms namely 1. GMG 2. Mixture of Gaussians (MOG), 3. Mixture of Gaussians 2 (MOG2) available in Python 2.7, on OpenCV 3.2, when applied to vehicle segmentation on highways. In total, seven videos were analysed, with different luminosity conditions and camera angles. To compare the methods, a database with ground- truth images of each analyzed video was created. An image with all objects properly segmented,

i.e., where all pixels belonging to the foreground are correctly marked, is a ground-truth image.

A) GMG

The GMG algorithm models the background with a combination of Bayesian Inference and Kalman Filters. The first stage of the method accumulates, for each pixel, weighted values depending on how long a color stays on that position. For every frame, new observations are added to the model, updating these values. Colors that stay static for a determined amount of time are considered background. The second stage filters pixels on the foreground to reduce noise from the first stage. The Python implementation of the GMG method has input parameters that may be modified. The default values are presented as:

```
cv2.bgsegm.createBackgroundSubtractorGMG(
initializationFrames=120,
decisionThreshold=0.8
)
```

The parameter initializationFrames indicates how many frames the algorithm is going to use to initialize the background-modeling method. During the initialization, the resulting frame is always black. The more frames used on this phase, the more stabilized the initial model is. The parameter decisionThreshold determines the threshold in which pixels are classified as background or foreground. In the first stage, when the algorithm accumulates values based on the time a color remains static, every pixel with a lower weighted value then the threshold is considered part of the background. Choosing high values for this parameter may result in loss of object detections.

B) Mixture of Gaussians (MOG)

Mixture of Gaussians, on this method, a mixture of k Gaussians distributions models each background pixel, with values for k within 3 and 5. The authors assume that different distributions represent each different background and foreground colors. The weight of each one of those used distributions on the model is proportional to the amount of time each color stays on that pixel. Therefore, when the weight of a pixel distribution is low, that pixel is classified as foreground. On OpenCV, the MOG implementation has input parameters that calibrate the behavior of the method. These parameters and their default values may be observed as:-

```
cv2.bgsegm.createBackgroundSubtractorMOG(
history=200,
nmixtures=5,
backgroundRatio=0.7,
noiseSigma=0
)
```

The parameter history is responsible for the number of frames the method will use to accumulate weights on the model, throughout the entire processing period. Low values result in increased sensitivity to sudden changes of luminosity. The parameter nmixtures indicates the method how many Gaussians distributions it should during the whole video. Higher values drastically increase processing time. The parameter backgroundRatio defines the threshold weight for the differentiation between foreground and background. Lower values may incur in false objects. Finally, the parameter noiseSigma defines the accepted noise level.

Low values create false objects.

C) Mixture of Gaussians 2 (MOG2)

The MOG2 method was based with the objective to solve one of the limitations that MOG had: the fixed amount of used distributions. By using a variable amount of Gaussians distributions, mapped pixel by pixel, MOG2 achieves a better representation of the complexity of colors in each frame. On its OpenCV implementation, MOG2 has three input parameters that may be changed to calibrate for each different video. The parameters and their default values may be seen as:-

```
cv2.createBackgroundSubtractorMOG2(
history=200,
varThreshold=16,
detectShadows=True
)
```

The history parameter functionality is analogue to the first MOG parameter. It denotes the number of frames to be used to model the background. The parameter varThreshold correlates the value of the weight of the pixels on the current frame with values on the model. Lower values on this parameter tend to create false objects. The parameter detectShadows enables or disables shadow detection. Enabling this parameter increases processing times.

Merits/ Demerits

Results show small differences between algorithms, so better option can not be based only on accuracy.

Precision, on the contrary to accuracy, shows performance differences between algorithms. On GMG, results appear distributed along all intervals, with the lowest value at 30%, and concentration of occurrences between 60% and 80%. With MOG and MOG2, results were superior. Both had precision rates near 100%, ie, with pixels correctly classified as foreground. Therefore, based on these results, GMG can be discarded as an inferior option forvehicle segmentation, based on our conditions and videos.

MOG2 performance results are 3 times better than MOG, and 10 times better than GMG.

This paper had the objective to compare three of the available background-modelling algorithms on OpenCV, in Python, and determine which one would be better suited for our needs, based on videos and conditions, to segment vehicles.

A novel technique was utilised, using objective parameters and results to select one algorithm. This technique, as a whole, can be replicated to analyse any number of algorithms, needing only ground-truth images of videos.

As a result, tests show that MOG2 has a better performance, with accuracy rates equivalent to other tested algorithms, but superior precision rate and lower processing times.

2.1.2 Background subtraction in Video Surveillance By Shanpreet Kaur[2]

The aim of this thesis is the real-time detection of moving and unconstrained surveillance environments monitored with static cameras. This is achieved based on the results provided by background subtraction. For this task, Gaussian Mixture Models (GMMs) and Kernel density estimation (KDE) are used. A thorough review of state-of-the-art formulations for the use of GMMs and KDE in the task of background subtraction reveals some further development opportunities, which are tackled in a novel GMM-based approach incorporating a variance controlling scheme. A system for performing the testing of algorithms was developed, written in C++ and making use of the library OpenCV for the image processing. Two trackers that were chosen Gaussian Mixture model (GMM) and Kernel Density estimation (KDE).They were evaluated to determine whether they benefit from background subtraction. In this thesis, we used the concept of background subtraction. The resulting background mask from the background subtraction is used in different ways for the trackers. For GMM and KDE a new image is created from the original image by setting background pixels to black.

The performance of the background subtraction is evaluated on different cases to see what impacts its performance. Two cases are constructed. The first is a simple sequence, taking a single large image and creating a video by sweeping over it with a smaller window. In this case, all pixels should be classified as background since there are no moving objects in a static image. The second case is to evaluate the result of the subtraction when there is no error in the data for the camera

movement. This was done by recording a sequence without moving the camera and then constructing a new video using small parts of the original sequence (moving a window over it, simulating a moving camera). By doing this vibrations are minimized and perfect knowledge of the per frame movement is obtained. Finally, the background subtraction is evaluated on sequences from the camera under the conditions: only pan motions only tilt motions, and both pan and tilt motions. For this background model, small angle rotations are assumed and the camera movement is approximated as a translation. **Merits/ Demerits**

Experiments are conducted on different video sequences, which demonstrate that GMM approach outperforms the other algorithm and is robust to outliers coming from inaccurate motion estimation and pixel misalignment, when registering consecutive images. This paper works well for moving objects, which pass by through the observed scene, and static objects, which are added or removed from the scene.

2.1.3 Object Motion Detection and Tracking for Video Surveillance By M. Sahasri , C. Gireesh[3]

An Automated Video Surveillance system is used to monitor security at sensitive areas such as banks, highways, crowded public places, borders, forest and traffic monitoring areas. The system aims at detecting and tracking a moving object. The task of object detection in surveillance video first is the background subtraction and the second one is the Foreground mask sampling.

The surveillance system presented in this paper can detect and track moving objects in a video sequence, and is resilient against temporal illumination

changes. The foreground is extracted from the video scene by learning a statistical model of the background, and subtracting it from the original frame. The background model learns only the stationary parts of the scene and ignores the moving foreground. The system uses the Gaussian Mixture Model for modeling the background adaptively. OpenCV was used for implementation for background subtraction.

Merits

In this experiment background subtraction algorithm was used for the detection of the moving object in the surveillance area. The demonstration system has the set up for the implementation of proposed system in the OpenCV software. Here the reference image is initialized in the code and then the subtraction of the current frame is done. And after the subtraction of the both frame the subtracted image is displayed on the screen. In the real time extraction the background image processing approach is used to provide the most complete feature dataset. The system finds its applications where real time surveillance is required such as bank, traffic monitoring, forest etc.

Demerits

But it is extremely sensitive to dynamic scene changes due to lighting and extraneous events. Background image is not fixed and it must adapt to motion changes like tree branch move, changes in background because of objects entering in a scene, stay for longer period without motion .object detected and classify.

2.2 Contour Detectcion

2.2.1 Traffic Sign Detection and Recognition Using OpenCV By Mrs. P. Shopa, Mrs. N. Sumitha, Dr. P.S.K Patra[4]

The aim of this project is to detect and recognize traffic signs in video sequences recorded by an onboard vehicle camera. Traffic Sign Recognition (TSR) is used to regulate traffic signs, warn a driver, and command or prohibit certain actions. This paper presents a study to recognize traffic sign patterns using openCV technique. The images are extracted, detected and recognized by pre-processing with several image processing techniques, such as, threshold techniques, Gaussian filter, canny edge detection, Contour and Fit Ellipse. Then, the stages are performed to detect and recognize the traffic sign patterns. The project is divided into different sections. The first section is Image Extraction and Sign Detection and Extraction parts. The video images have been taken by a video camera, and Image Extraction block is the responsible for creating images. The Sign Detection and Extraction Stage extracts all the traffic signs contained in each image and generates the small images called blobs. The second section is Traffic Sign Detection. This stage is the image processing procedure. Image input from video sequence which is the natural background viewing image fed into the system. The image data is read both in color, and black and white mode Due to the black and white mode image is the base image that used to find the threshold of this image, this threshold is the criterion to change image from black and white to binary image. Moreover, the binary image is used to find contour and the interested region later on. Before the black and white image change to binary, the technique of smooth image with Gaussians filter and Canny edge detection to

enhance image. Therefore, it shows that the smooth technique has potential to enhance the image to obtain the required region. After obtaining the binary image, it is processed to retrieve contours by find contour function for binary image and to return the number of retrieved contours which stored in the chain format. The OpenCV library uses two methods to represent contours. The first method is called the Freeman method or the chain code. For any pixel all its neighbors with numbers from 0 to 7 can be enumerated. The 0-neighbour denotes the pixel on the right side, etc. As a sequence of 8-connected points, the border can be stored as the coordinates of the initial point, followed by codes (from 0 to 7) that specify the location of the next point relative to the current one. The fit ellipse method was used to find ellipses. This is done for each contour. Then this ellipse is checked whether if it is a traffic sign or not.

Merits

The experiments show consistency results with accurate classifications of traffic sign patterns with complex background images. The processing time in each frame of image is provided which is satisfied to apply in the real application.

2.2.2 Image Handling and Processing for Efficient Parking Space Detection and Navigation Aid By Chetan Sai Tutika, Charan Vallapaneni, Karthikeyan B[5]

This paper aims to develop a robust and flexible algorithm for vacant parking space detections using image processing capabilities of OpenCV. It removes the need for independent sensors to detect a car and instead, uses real-time images

derived from various sources and servers to consider a group of slots together. The foremost stage of the paper is Image Acquisition and Conversion in which images are acquired and converted to gray scale images. The second stage is Image Segmentation and Smoothing where image acquired from the first stage is then subjected to Gaussian blurring using a 3x3 kernel to smooth the image and reduce high intensity value pixels. The Gaussian smoothing operator is a 2-D convolution operator that is used to 'blur' images and remove detail and noise. Then, this stage is followed by Morphological Transformation and Contour Detection where morphological transformations such as erosion and dilation are performed on figure to enhance the existing contour features. The erosion of a figure by a structuring element produces a new binary image with ones in all locations (a, b) of a structuring element's origin at which that structuring element fits the figure. While dilation of the eroded image by a structuring element produces a new binary image with ones in all locations (a, b) of a structuring element's origin at which that structuring element hits the input image, repeating for all pixel coordinates (a, b). Dilation has the opposite effect to erosion as it adds a layer of pixels. The image after the enhancement is used to detect the contours. The contours of an image are connected components which are either 8 connected or 4 connected pixels. The retrieval method used, RETR_EXTERNAL, extracts the extreme outer contours present and the contour approximation algorithm, CHAIN_APPOX_NONE stores absolutely all the contour points. The last stage is False Contour Removal and Module Classification. In this stage, it can be observed that while the use of Gaussian blurring can reduce the amount of noise in the Binary image, there is still a considerable amount of noise in the image. The first step includes classifying contours based on the area and Y-axis positions. Contours with area less than 70 units and Y-positions greater than 270

are made zero. The image is then classified based on the elliptical angles. Contours with angles between 80 and 100 are assumed to be noise and made zero. The algorithm is designed to extract the position of the vehicle contours from previous step and design the appropriate bounding boxes. The angle, height, and width are the extracted information from the contours; the number of iterations is also calculated depending on the extracted information for efficient processing.

Merits

The results produced are accurate, testing with existing autonomous is yet to be done.

Demerits

The manual setting during empty parking spaces can be replaced with dynamic algorithm with support for much larger ground. The angle and height of the image can be adjusted for more space availability and slot detections. The segmentation can be further improved for detection of complete vehicle contours.

2.2.3 Detection of Potholes using Image Processing Techniques By Akshata Bhat , Pranali Narkar[6]

Roads are considered to be the main mode of transportation. But due to this heavy use of roads and environmental factors, these roads need a scheduled maintenance. Often this maintenance is not performed since it is not possible to monitor each and every place or simply because of ignorance. The proposed system has been implemented under a Windows environment using OpenCV library. Simple image processing techniques

like canny edge and contour detection with Hough transform is used for effective pothole detection. Proposed Method Video has been captured using a camera module interfaced with raspberry pi. Frames of the video are extracted and the individual frame is considered as an image which is further processed. The first step after frame extraction was the conversion of the RGB image into grayscale using standard techniques to make processing of image faster. After gray scaling three different blurs on the image were performed. The image is firstly blurred using averaging then with Gaussian filter and lastly with median blur so to remove unwanted noise from the image. To achieve more accurate edge detection from a depth image was modified by the process using morphological operations. Erosion after blurring operations which is followed by two iterations of dilation were performed. The pothole detection is utilizing canny edge detection technique. The detection technique is a multi-stage method to detect a wide range of edges in images. Otsu's method for reduction of a gray level image to a binary image was used. The algorithm assumes that the image contains two classes of pixels following bi-modal histogram (foreground pixels and background pixels), it then calculates the optimum threshold separating the two classes so that their combined spread (intra class variance) is minimal, or equivalently (because the sum of pair wise squared distances is constant), so that their inter-class variance is maximal. The system then uses contour detection technique. For better accuracy, binary images were used. Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition. Thus, it is very useful in pothole detection system.

Merits/ Demerits

The system provides satisfactory results for collected video samples. The system will be installed in a fixed position on the light poles which ensures less handling. Also, this system keeps a track of the negligence and delay. The system also detects potholes in time without damaging the cars for potholes detection. Thus, this approach makes the system more feasible and favorable.

2.3 Offside Based Papers

2.3.1 Vision Based Dynamic Offside Line Marker for Soccer Games By Karthik Muthuraman , Pranav Joshi , Suraj Kiran Raman[7]

Offside detection in soccer has emerged as one of the most important decision with an average of 50 offside decisions every game. False detections and rash calls adversely affect game conditions and in many cases drastically change the outcome of the game. The human eye has finite precision and can only discern a limited amount of detail in a given instance. Current offside decisions are made manually by sideline referees and tend to remain controversial in many games. This calls for automated offside detection techniques in order to assist accurate refereeing. In this work, computer vision and image processing techniques like Hough transform, color similarity (quantization), graph connected components, and vanishing point ideas are extensively used to identify the probable offside regions. In order to effectively track and identify players, it is important to

segment out the field-area exactly. To be specific, the top and side boundaries of an image is needed to segment out the play-area. The presence of the line-boundary (white line) was exploited after identifying it using the Hough transform by carefully selecting only horizontal lines. The regions below this detected line is only considered as the field area. With the team kit information and the processed field area from the above step,the color information of the team jerseys were used, and only those blobs which satisfy the color properties were extracted.Image opening morphological operation was performed to remove small insignificant components which is followed by image dilation to increase the strength of player detection. The main objective is to make real-time offside calls. Hence it is very important to have less complex models that can be implemented real-time. In this work, functions like Harris features were used where in iteration through all bounding boxes of the players, determining Harris corner features in each local bounding box region was calculated. For each Harris corner, the motion (translation or affine) between consecutive frames is computed and they are linked in successive frames to get a track for each Harris point. This way algorithm was optimised and speed up the process by switching between detection and tracking.After determining the player

bounding boxes for each frame (either via direct detection or tracking), the offside line was determined. To detect this, first which defending player is closest to the goal along a vanishing line was found.

Merits/ Demerits

The system is cost effective because it doesn't use any additional hardware excluding cameras.Even though the results were produced as required, for most

videos the ball was not identified properly.This required multiple cameras. **There were lot of false detections for players.**

2.3.2 Offside Detection in the Game of Football Using Contour Mapping By Pratik N Patil , Rebecca J Salve , Karanjit R Pawar[8]

This paper is about offside detection in the game of football using contour detection in computer vision.Contour detection method to decide whether the player is in offside position or not was used. The purpose of contour detection method in this project is to find the position of furthermost defender and attacker, while the attacker is receiving the pass from his teammate. Their horizontal positions are compared with respect to each other to give the final results. Various filtering techniques are used to obtain the desired output. The main objective is to determine offside in football matches so that the officials can improve the accuracy of decision making. Firstly, the video containing offside is acquired. After obtaining all the frames, the frame in which offside is occurring is selected. This frame is read as an image for operations. The image is then converted into HSV image for colour extraction. The colours of jerseys are known before the match, hence the values for hue, saturation and intensity before the match can be set. This image is then converted into grayscale image because contours can be located only on binary images in OpenCV. The next step is to locate the contours in the image. Contours in selected region of interest are obtained. After the filtering, contour of the furthermost attacker and defender is chosen such that the contour represents the goal scoring part, that is contours for hand is neglected

because only goal scoring parts are considered for offside. After determining the contours of furthermost attacker and the last defender, coordinates of their contours are obtained. From these coordinates, leftmost or rightmost coordinates are chosen depending on the side of the field. The x coordinates of these extreme points are compared. If the x coordinate of attacker is greater than the x coordinate of defender and the goal is on the right side of the field, then attacker is in offside position.

Merits/ Demerits

This paper provides satisfactory results but requires multiple cameras to work properly. It also suffers from occlusion.It requires input patch for jersey to work. The system is cost effective because it doesn't use any additional hardware excluding cameras.

2.3.3 A modern approach to determine the offside law in international football BY Alexander Henderson , Daniel Lai , Tom Allen (2014)[9]

A system has been proposed that uses player tracking technology to quantify players' positions and runs an algorithm to determine which players are offside. The likelihood of algorithm error is dependent on the accuracy of player tracking technology. It was found that algorithm accuracy is improved by increasing the sampling rate and precision of player tracking technologies. The most suitable technology form for use in the proposed system is camera based player tracking. No device is required to be worn by players and body segment positions can be determined to ensure the offside law is completely adhered to. A prototype watch

been developed to prove the systems concept. It houses: a lithium ion battery, Xbee wireless transceiver, LCD display and two buttons to select the team in question. The Xbee module is configured in transparent mode to allow any wireless data packet received to be directly transferred to the serial LCD monitor. Another Xbee is located in the 'base station' on the side of the field. Here a computer is fed with the wireless Xbee data as well as player positions. Team selection is achieved by pressing one of two buttons located on the watch's top surface. To determine which players are offside at a given time an algorithm is run to firstly find the second last defender and then compare every player from the attacking team against this reference. Players are identified by their jersey numbers and by their team. The reference is created using a bubble method in which the largest of a series of values rises to the top. The second largest is then taken for comparison. Each player from the attacking team is compared against the defending team's reference. Those players who exceed the reference position have their jersey number added to an array. Once all players have been compared, the jersey numbers of offside players are sent as a wireless data packet back to the referee's watch to be displayed on the LCD monitor.

Merits/Demerits

The veracity of this system is directly proportional to the accuracy of the player tracking technology. Camera tracking mitigates the need for players to wear any devices to monitor their location. Although still in its infancy stages the proposed system is theoretically capable of providing more accurate offside decisions. There is a delay in transmission between the base station and the referee making the system slightly slower than real time decision.This system places a lot of responsibility on the referee which makes the system not fully automated.

2.3.4 An Investigation into the Feasibility of Real-time Soccer Offside Detection from a Multiple Camera System (2009) BY T. D'Orazio (Member IEEE), M. Leo, P. Spagnolo, P. L. Mazzeo, N. Mosca, M. Nitti, A. Distante[10]

In this work the feasibility of multiple camera system for automatic offside detection was investigated.Six fixed cameras were proposed, properly placed on the two sides of the soccer field (three for each side) to reduce perspective and occlusion errors. The images acquired by the synchronised cameras are processed to detect the players position and the ball position in real time; a multiple view analysis is carried out to evaluate the offside event, considering the position of all the players in the field, determining the players who passed the ball, and determining if active offside condition occurred. The whole system has been validated using real time images acquired during official soccer matches, and quantitative results on the system accuracy were obtained comparing the system responses with the ground truth data generated manually on a number of extracted significant sequences. The hardware architecture consists of six high resolution cameras that have been placed on the two sides of the field, with the optical axes parallel to the goal line plane to reduce perspective errors. The acquired images are synchronised and transferred to six nodes by fibre optic cables. First, a background subtraction algorithm based on the local energy evaluation along a temporal window allows the detection of moving areas that can be associated both to players and ball candidate regions. Second, the player team classification is carried out without any human intervention by using an unsupervised clustering algorithm that process the images when players enter the stadium generating

automatically the five classes corresponding to the strips of the two teams, the two goalkeepers and the referees. Normalised colour histograms are used to generate the class prototypes and are continuously updated during the match to handle the varying lighting conditions. The ball detection algorithm, starting from candidate moving areas whose dimensions are compatible with the expected ball size, compares the selected region with different sets of reference examples of ball. A probability map, based on the past information about the ball motion, is used to evaluate the likelihood that any point of the image represents the ball centre.

Merits/ Demerits

A good percentage of active offside events were correctly recognised by the system. The same can be said for passive offside events. experiments demonstrated that the use of multiple cameras with a high frame rate allowed the reduction of perspective errors and the detection of quick events that happened in distant parts of the field. The opposite cameras solved many cases in which the decision was difficult by using only one view. It did not work in cases where there was the complex configurations among players, the difficult interactions between the ball and the players and in some cases it did not interpret the offside rule properly.

S NO	TITLE WITH YEAR	METHOD	FEATURES	MERITS	DEMERITS
1	A Comparison between Background Modelling Methods for Vehicle Segmentation in Highway Traffic Videos (2018)	Seven videos were inputted to the python code which used functions provided by OpenCV to produce background-subtractor. These were checked with the ground truth images.	-	-	-
2	Background subtraction in Video Surveillance (2017)	The footage was processed by two trackers GMM and KDE using their implementatio-ns provided by OpenCV	It is robust to outliers coming from inaccurate motion estimation, when registering consecutive images	Works well for moving objects	-
3	Object Motion Detection and Tracking for Video Surveillance (2017)	The foreground is extracted from the video scene by learning a statistical model of the background, and subtracting it from the original frame. The system uses the Gaussian	The setup cost is cheap	Works well in real time application	Extremely sensitive to dynamic scene changes

FIGURE 2.1: Literature Survey Content

		Mixture Model for modelling the background adaptively.			
4	Traffic Sign Detection and Recognition Using OpenCV (2014)	Image input is obtained from video sequence and this image is processed to find contours and fit ellipse was used to find ellipses	Classifies complex backgroun-ds properly	Classifies Patterns Accurately	It does not notify the driver about the signal
5	Image Handling and Processing for Efficient Parking Space Detection and Navigation Aid (2018)	Morphological transformations are applied to the acquired image and then the image is processed to find contours using OpenCV.	Handles noise which leads to false detection properly	Results produced are accurate	Image segmentation needs development
6	Detection of Potholes using Image Processing Techniques (2017)	Video was captured by raspberry pi. Morphological transformations are applied to the frames.Then contour detection is used to find potholes.	The system has a low cost and high compatib ili-ty with other interfaces	The system detects potholes in time without damaging the cars for potholes detection	A considerable amount of false positives are generated
7	Vision Based Dynamic Offside Line Marker for Soccer Games	Image processing techniques like Hough transform, colour similarity (quantization), graph connected	Works well even when crowd comes into play	It does not use additional hardware	Multiple cameras are required

FIGURE 2.2: Literature Survey Content

	(2018)	components, and vanishing point ideas to identify the probable offside regions.			
8	Offside Detection in the Game of Football Using Contour Mapping (2018)	Contour detection method is used. It finds the position of furthermost players.Their horizontal positions are compared with respect to each other to give the final results. Filtering techniques were used to improve the output.	It is very fast and produces satisfact ory results.	System is cost effective	It requires input from the user to work
9	A modern approach to determine the offside law in international football (2014)	The referee wears a Xbee watch which is used to inform the system that a potential offside event is happening. Players are identified by their jersey numbers and by their team. The reference is created using a bubble method in which the largest of a series of values rises to the top.	Works in real time	Removes the need of players having to wear a tracker	System is not fully automated

FIGURE 2.3: Literature Survey Content

10	An Investigation into the Feasibility of Real-time Soccer Offside Detection from a Multiple Camera System (2009)	Six cameras are used, three on each side to capture the footage. Then a background subtractor algorithm is used to find out players and ball.The detected players are classified by using an unsupervised clustering algorithm that process the images when players enter the stadium generating automatically the five classes corresponding to the strips of the two teams, the two goalkeepers and the referees.	Reduced number of perspective errors and the detection of quick events that happened in distant parts of the field.	A good percentage of active offside events were correctly recognised by the system	It did not work in cases where there was difficult interactions between the ball and the players

FIGURE 2.4: Literature Survey Content

CHAPTER 3

METHODOLOGY

We have divided the problem of detecting offside into multiple sub-problems namely, player identification, ball-tracking, drawing offside lines, detecting passes and finally applying the offside rules.

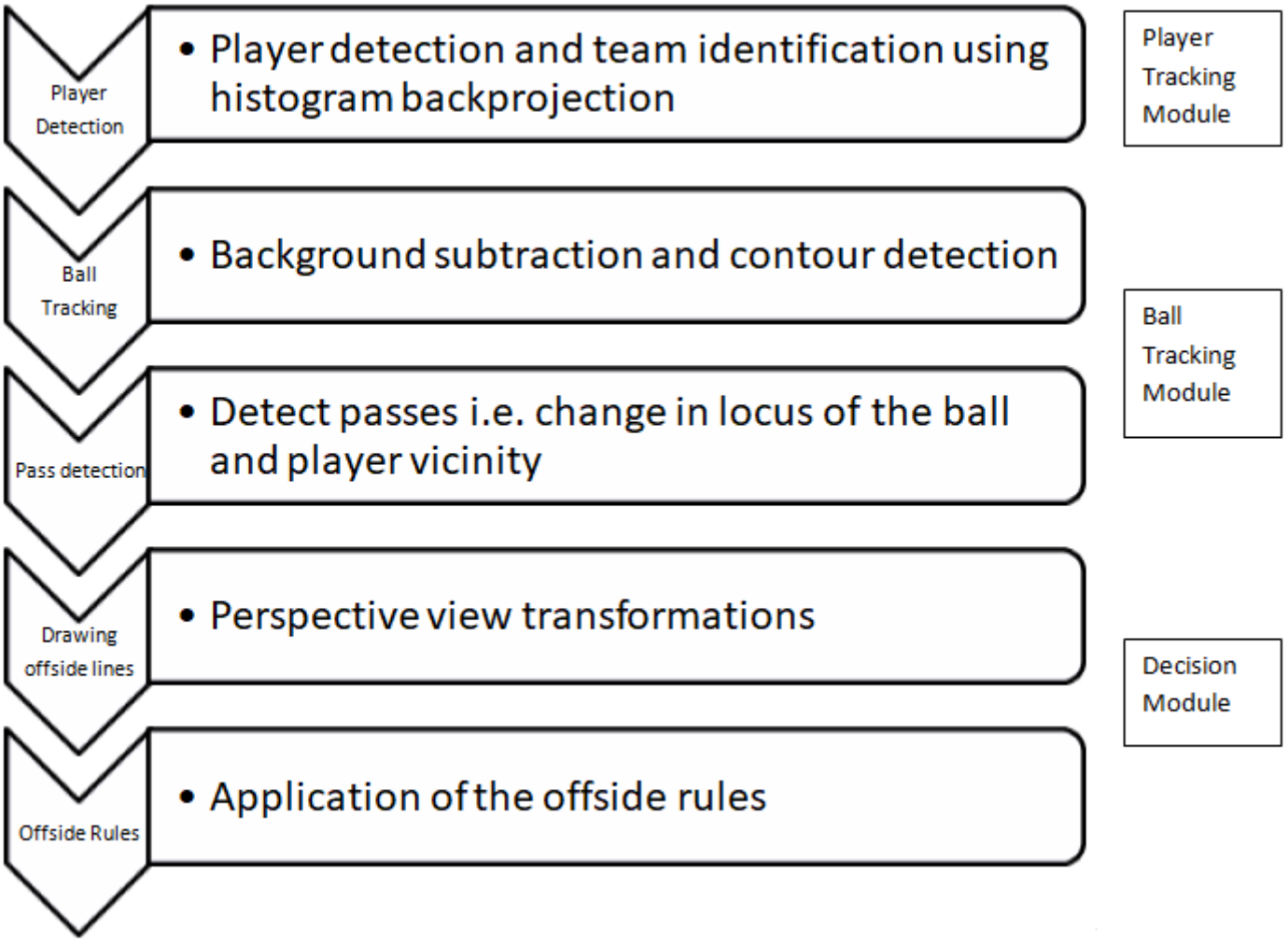


FIGURE 3.1: Problem Overview

The ball tracking module would take care of detecting the ball, tracking it and detecting whether a ball pass has occurred.

The player tracking module would detect the players of each team, attacking and defending, and get an approximate location of the foot of the players. Finally the two would be integrated into one program. The ball pass is detected only when it is passed from one player to different player.

If the player of the attacking team receiving the pass(when he receives the pass) is behind the last player of the defending team then offside is called. Note that offside is NOT called if the attacking player is behind the offside line but doesn't receive the ball.

3.1 Player Detection

3.1.1 Histogram Backprojection[11]

Back Projection is a way of recording how well the pixels of a given image fit the distribution of pixels in a histogram model. To make it simpler: For Back Projection, the histogram model of a feature is calculated and then used to find this feature in an image.
If the histogram of flesh color is obtained (say, a Hue-Saturation histogram), then it can be used to find flesh color areas in an image.

Let us explain this by using the skin example:

Lets consider that a teatm-1 jersey patch is obtained as input from the user. After that histogram (Hue-Saturation) based on the image is built. The histogram is going to be our model histogram (which we know represents a sample of jersey tonality).

Now, let's imagine that another image is obtained which includes a field, multiple players and ball(Test Image). Its respective histogram is also derived.

The goal is to use our model histogram (that we know represents a jesrey tonality) to detect the same jersey areas in our Test Image. Here are the steps

1. the patch of the player(Region of Interest) from team A using getROIvid function.

2. Convert the input RGB frame to HSV frame using OpenCV's cvtColor() function.

3. Construct a Histogram(Hue-Saturation) from the input patch with the help of calcHist() function by passing the HSV frame(Region of Interest) as parameter.This returns the Histogram of ROI.

4. Normalise the Histogram using cv2.normalize() and pass the ROI.

5. Now the same is done to player from team B.

6. If $\text{roi}_h ist_A (which is the histogram of ROI of team A) is not none then$

   $a) Apply cv2.calcBackProject()$

   $b) After applying back$ –

   $projection find contours in the frame using cv2.findContours() and store the result in the variable$

   $c) If the length of the contours is not zero then$

   $c.1) The bounding rectangle of the contour is found by using cv2.boundingRect() and returns x, yc$

   $c.2) Using cv2.moments(), the centre of the contour is found out and with the help of that foot of ap$

   Applying the steps above, the following BackProjection image is obtained for our Test Image:

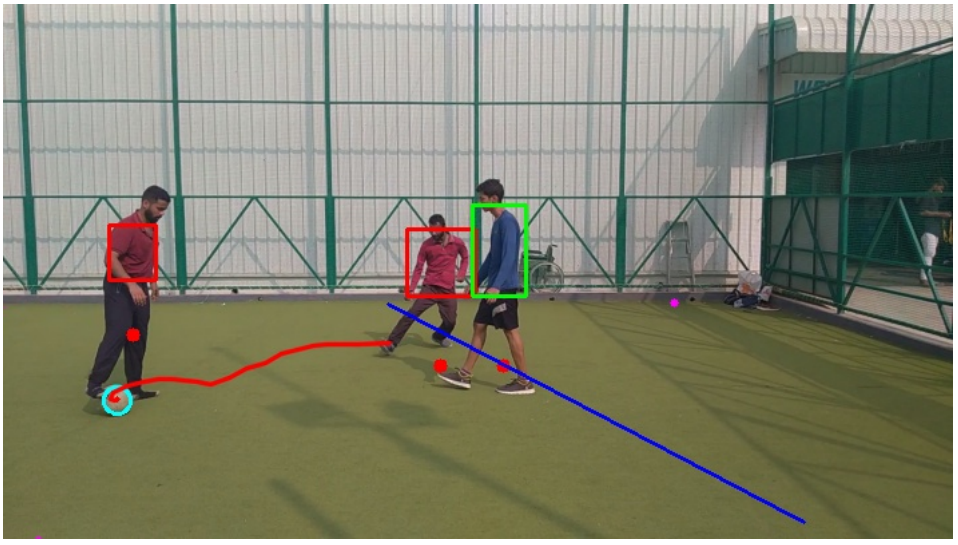


FIGURE 3.2: An Image Frame In a Video

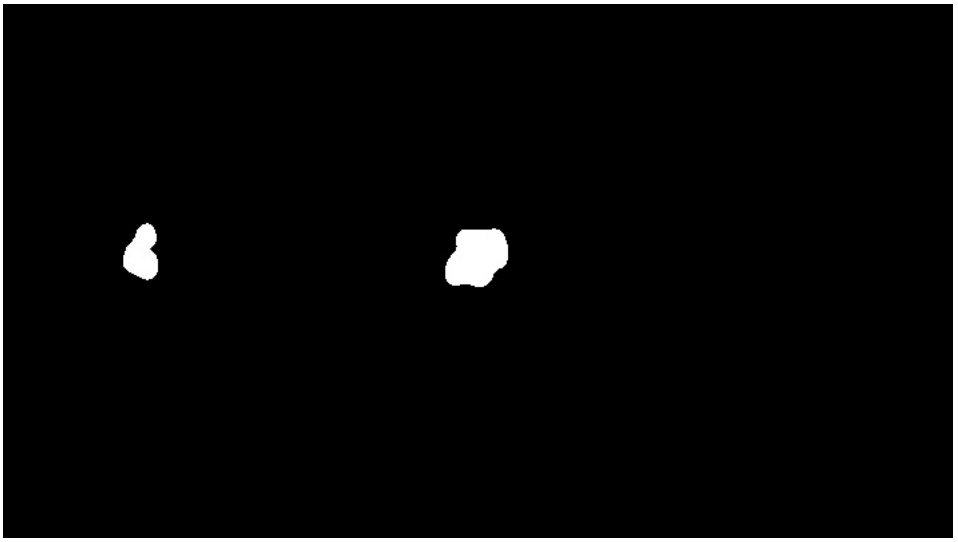


FIGURE 3.3: Team A players' jerseys recognised

In terms of statistics, the values stored in BackProjection represent the probability that a pixel in Test Image belongs to a jersey area, based on the model histogram that was used.

Also, in our case it helps us differentiate the two teams as, it identifies the part of the images based on color.

3.2 Ball Tracking

3.2.1 Background Subtraction[2,3]

Every frame of a video can be divided into two different areas: foreground, which groups pixels that are part of the objects of interest (in this case, players of different teams and ball); and background, which groups all pixels that are not part of an object of interest, such as field, sky, stands etc. Based on this division, any system

with the objective to detect objects automatically, with segmentation and tracking, must be able to differentiate between these two areas.

Mixture of Gaussians 2 (MOG2)[1]

The MOG2 method was based with the objective to solve one of the limitations that MOG had: the fixed amount of used distributions. By using a variable amount of Gaussians distributions, mapped pixel by pixel, MOG2 achieves a better representation of the complexity of colors in each frame. On its OpenCV implementation, MOG2 has three input parameters that may be changed to calibrate for each different video. The parameters and their default values may be seen as:-

```
cv2.createBackgroundSubtractorMOG2(
history=200,
varThreshold=16,
detectShadows=True
)
```

The history parameter functionality is analogue to the first MOG parameter. It denotes the number of frames to be used to model the background. The parameter varThreshold correlates the value of the weight of the pixels on the current frame with values on the model. Lower values on this parameter tend to create false objects. The parameter detectShadows enables or disables shadow detection. Enabling this parameter increases processing times.

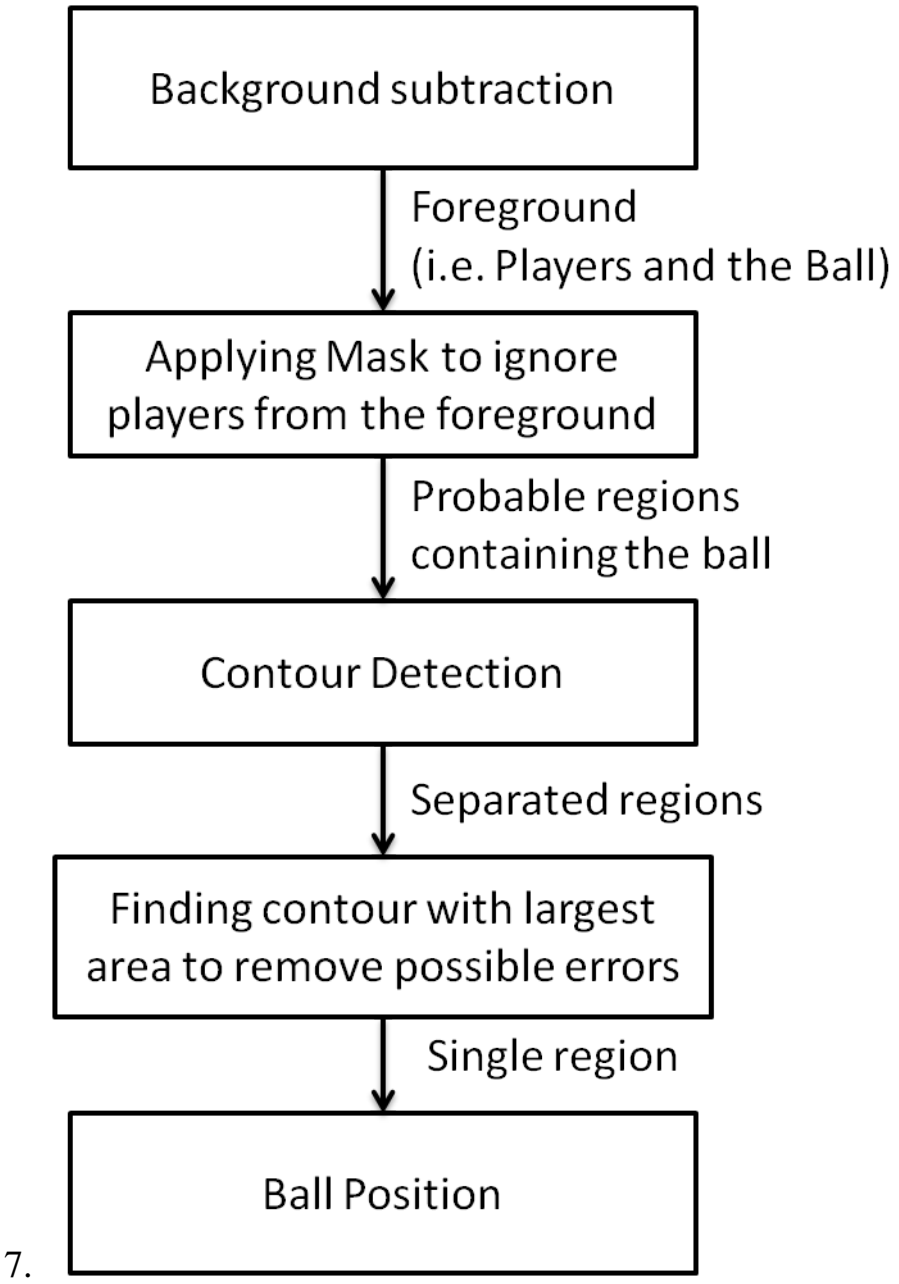
We have used MOG2 in our problem instead of GMG or MOG as it is the most recent, and gives better performance rate and is more precise compared to the other two. (as mentioned in the literature survey)

3.2.2 Contour Detection[4,5,6]

The idea of contour tracking, also known as boundary following or edge tracking, is to traverse the border of a region completely and without repetition.The result of the contour tracking is to obtain a boundary points sequence. Just as its name implies, contour tracking is through the order to find out the boundary of the edge points to track. The key of the contour tracking is to judge the next boundary point according to the adjacent points in binary image. Contour tracking is widely used in the fields of image processing, image restoration and image recognition. In the image processing, there are a lot of contour tracking algorithms, some of them have high accuracy and efficiency. Opencv offers cvFindContours function to realize contour tracking, which retrives contour from the binary image, and return the number of the detected contour. By using this method, we can clearly distinguish the boundary of the copper core, so we can accurately determine the exact number of copper core.

We call the openCV cvFindContours after background subtraction is done, which only selects the foreground i.e. the ball and the players as well. Also we apply color based masks obtained from the input patch to differentiate the ball and the players before calling the function.

Due to noise, the program still finds several contours, and we identify the ball as the the one with the largest area of those contours. Also, a threshold is applied to



7.

FIGURE 3.4: Ball Segmentation

the contour area so as to avoid false detection.

Here is the algorithm for ball detection:

1. Create a background subtractor using cv2 function, createBackgroundSubtractorMOG2() and store it in a variable called

foregroundbackground.

2. Now this is used to to do background subtraction and the background is removed.

3. Convert the input RGB frame to HSV frame using OpenCV's cvtColor() function.

4. Find contours in the frame using cv2.findContours() and store the result in the variable contours.

5. For every contour

   a) Find the minimum enclosing circle for the particular contour using cv2.minEnclosingCircle().

   b) Store the radius in a variable.

   c) If radius is above a particular threshold(5 in this case) break out of the loop.

6. Using cv2.moments(), the centre of the contour is found out. With the centre and the radius of the contour a circle is create using cv2.cicle() to show the ball in the output.

FIGURE 3.5: Contour detection after background subtraction and color based masking

FIGURE 3.6: Segmented ball

3.3 Pass Detection

The program we wrote has specifically assigned variables for the direction of the ball. If the deflection in the locus of the ball is more than a threshold value of the angle, it is considered as the beginning of a new pass.

The next step is to check the vicinity of the ball to detect the passer and the receiver.

Here's the algorithm for the same:-

1. Ball is detected in each frame and the center point of the circle is aquired.
2. An array, ballPoints, is initialised for the position of the ball in the last 20 frames. If the ball is detected in a frame, it's center is appended to the array. If the array is full, dequeue and store the next point.
3. The gradient of the motion of the ball is calculated as grad=arctan2((ballPoints[9][1] - ballPoints[0][1]), (ballPoints[9][0] - ballPoints[0][0])) (for the last 10 points), where arctan2 is a predefined function in numpy array.
4. The current gradient is compared to the previous gradient calculated.
5. If the difference is greater than the threshold(which we have kept as 20 degrees), update the passer as per the following steps. Else, continue.
6. . Two different arrays are initialised to contain the positions of detected players of both the teams.
7. To detect the passer, vicinity of each player with the ball is found.

8. Minimum distance players from both the teams are found. Distance is calculated using Euclidean Distance Formula,

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \text{.....} (Eq3.2)$$

.

9. Thus, we get teamAmin and teamBmin as the nearest players of each team from the ball.

10. If teamAmin is less than teamBmin, the passing team is set to be team A and passerIndex is set to teamAmin. Else the passing team is set to be team B passerIndex is set to teamBmin.

3.4 Drawing offside lines

Once the player tracking and ball tracking modules are done with their job, it's time to finally draw the offside line i.e. line with respect to the last defender. The line drawn must be parallel to the edges of the field.

When seen from a point, the view is called a perspective view. In a perspective view, even parallel lines meet at a point. For example, a straight road when seen in a picture converges at a point.

Thus, in our case, we could not simply draw a line perpendicular to the screen; rather we were required to use the perspective view and its inverse transformations to achieve the desired lines in our frames. To do so, we input two points for each edge of the field from the user in the beginning of the program execution.

3.4.1 Finding Corner Points

3.4.1.1 Finding line equation

These edges are further used to find the corner points of the football field. Firstly, slope of each edge is found out as:

$m = (y_2 - y_1)/(x_2 - x_1)$

Equation of a line is given by:

$y - y_1 = m(x - x_1)$ .....(Eq 3.4)

Equation of each edge is found out. Representing the equation in the form of "ax + by = c", we get,

a = m,

b = -1,

$c = mx_1 - y_1$

Now it is clear that any line can be represented in the form of ax + by = c, given any two points on the line.

3.4.1.2 Solving for Intersection Point

Given two line equations l_1 and l_2, represented by three values a_1, b_1, c_1 and a_2, b_2, c_2 respectively, the intersection point can be found out as follows.

The two equations can be written in the form of matrix as:

$$\begin{pmatrix} a1 & b1 \\ a2 & b2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} c1 \\ c2 \end{pmatrix}$$ .....(Eq 3.5)

(OR) AX=C

Multiplying by A^{-1} both sides we get $A^{-1}AX=A^{-1}C$

which implies, $X = A^{-1}C$

Thus, $\begin{pmatrix} x \\ y \end{pmatrix} = A^{-1}C$ .....(Eq 3.6)

The values of x and y give the intersection point.

3.4.2 Perspective View

A perspective projection is used to draw a three dimensional picture on a 2D screen, as it appears to the human eye. For example, a photograph taken involves perspective projection.

In computer graphics, perspective projection is a concept employed to generate images or photographs that look so natural. In perspective projection farther away object from the viewer, small it appears. This property of projection gives an idea about depth. Two main characteristics of perspective are vanishing points and perspective foreshortening. Due to foreshortening object and lengths appear smaller from the center of projection. More we increase the distance from the center of projection, smaller will be the object appear.

Vanishing Point is the point where all lines will appear to meet.

3.4.3 Generating the top view and Drawing Offside Line

1. The user first inputs the endpoints(corners) of the field. In our case, the corners were not visible, hence we take 2 points along each edge and solve

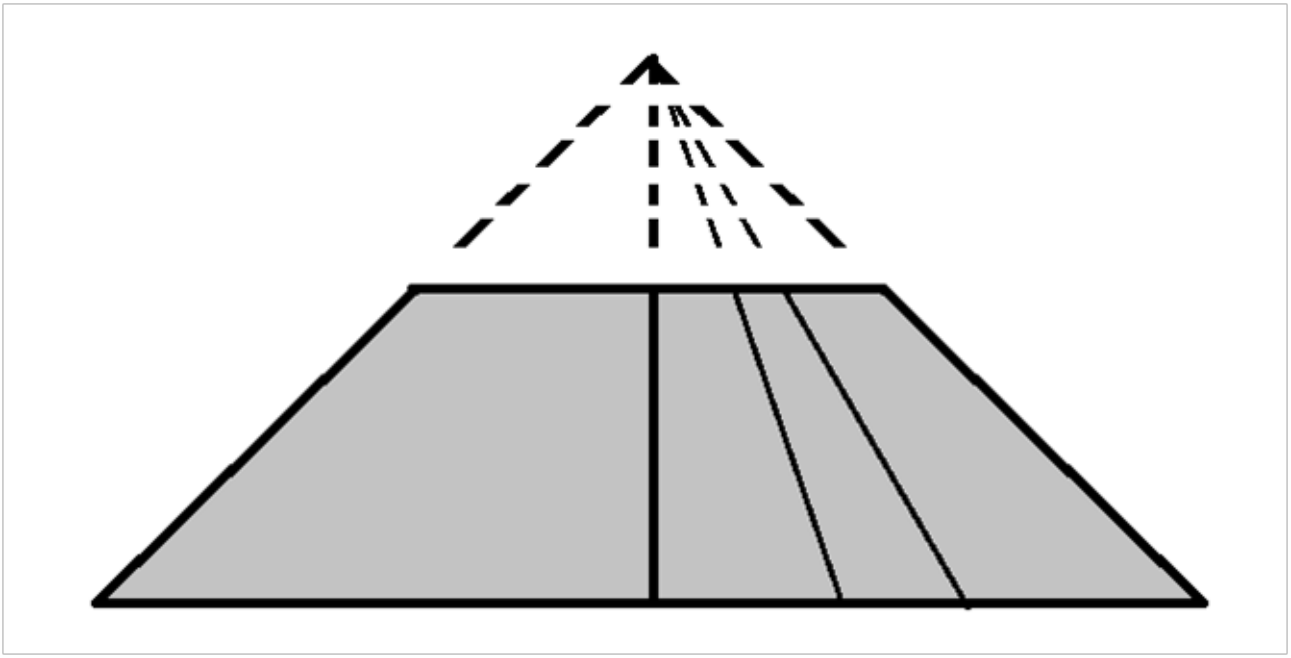


FIGURE 3.7: Contours after background subtraction and color based masking

the equation of line to get the 4 intersection points as given in section 3.4.1.2.

2. These 4 intersection points act as src_points for the transformations. And the dest_points are the corners of a rectangle in a top view.

3. cv2. getPerspectiveTransform (src, dst) is called to get the homography matrix, M.

4. Inverse of M is calculated using np.linalg.inv() function.

5. The new positions of each team player is calculated by multiplying by M matrix, i.e, transformation matrix.

6. These new positions are stored in separate arrays called newTeamA and newTeamB and are a representation of the top view.

7. The position of the last defender is found out as the min of the defending team, i.e. newTeamB array. last_def = np.argmin(newTeamB[:,0])

8. Offside line is to be drawn perpendicular to the screen in the top view and passing from last_def. i.e. A line from p1(last_def[0],0) to p2(last_def[0], bottom most pixel in the image)

9. Points p1 and p2 are inverse mapped from top view to the original view by multiplying them with M inverse matrix.

10. A line is drawn using the resultant points.

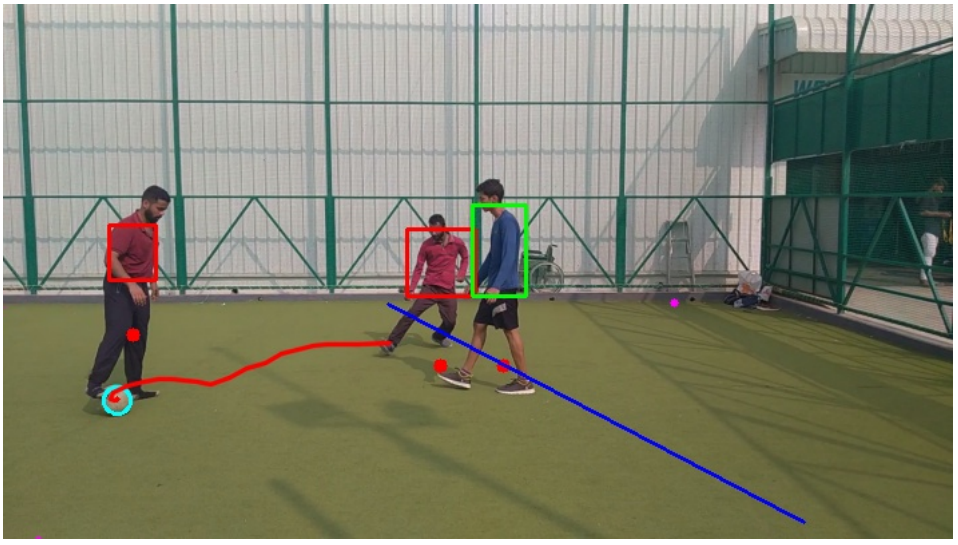


FIGURE 3.8: Drawing offside line

3.5 Offside Rule Application

Here we are, done with all the video processing required to claim whether offside has occurred or not. We should check the offside for each frame or approach it at a particular time interval, or use another intelligent way to say that now we must check for offside irrespective of when we checked for it the last time.

We know that an offside offence can occur when someone passes the ball to someone of his own team. Thus we only check for the offside conditions when a pass is detected and the passer and the receiver belong to the same team, i.e. attacking team.

We compare the offside line (the line of the last defender) with the position of the receiver and that of the passer. If the x-coordinate of the position of the passer is lesser than that of the offside line and x-coordinate of the position of the receiver is more than it, it is given as an offside offense.

1. If both passing and receiving teams are attacking teams, check next conditions. Else mark onside.

2. If X_position(Offender_i) $\leq$ X_position(Offside) $\leq$ X_position(Offender_j), where Offender_i and Offender_j are passers and receivers respectively of the attacking team, then it is an offside offence.

3. If X_position(Offside) $\leq$ X_position(ball)
   and X_position((Offender_i) $\leq$ X_position(Offender_j), where Offender_i and Offender_j are passers and receivers respectively of the attacking team, then also it is an offside offence.

Offside Offence:

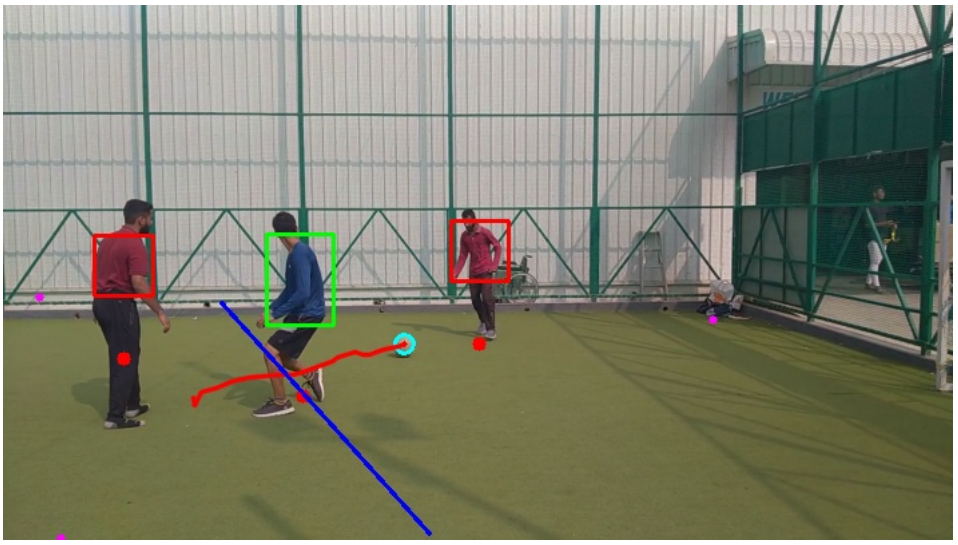


FIGURE 3.9: offside

Onside:

FIGURE 3.10: not an offside

CHAPTER 4

CONCLUSION AND FUTURE WORK

In this thesis, we have discussed in detail the decision algorithm (and the related tool) that detects off side offences in a football game efficiently. We have noted that given a static camera view, and the required input patches, the system tracks the players and the ball and accordingly decides if a pass made was an off side offence or not.

We learnt and implemented the Histogram BackProjection to track players and at the same time differentiate their teams. This algorithm uses histograms to match the sample image onto the target image to find areas with the same color patterns. Background subtraction was used to bring out the regions of interest in the given frames, which after applying contour detection, color based masks and noise filtering helped tracking the ball.
Concept of Homographic transformations was learnt and applied to achieve the top view from the perspective view and vice-versa.
Off side rules were applied once the video processing was done with.

Future Prospects of Improvement

The player's and ball's location can be detected with much greater accuracy by combining data from multiple cameras. In our application, there is still the issue of occlusion i.e. when one player blocks the view of another player. This can be resolved by using multiple viewpoints. Using multiple cameras to cover entire field. Using better algorithms to detect players and ball as which does not depend on colour of jersey.

REFERENCES

1. Marcomini, L.A. and Cunha, A.L.. (2018) 'A Comparison between Background Modelling Methods for Vehicle Segmentation in Highway Traffic Videos.' .

2. Shanpreet, Kaur (2017) 'Background Subtraction in Video Surveillance', University of Windsor.

3. Sahasri, M., and Gireesh, C.. (2017) 'Object Motion Detection and Tracking for Video Surveillance', International Journal of Engineering Trends and Technology (IJETT) .

4. Shopa, P,Sumitha, N., and Patra, P.S.K.. (2014) 'Traffic Sign Detection and Recognition Using OpenCV', ICICES2014 - S.A.Engineering College, Chennai, Tamil Nadu, India.

5. Chetan, Sai, Tukita ,Charan, Vallapaeni, Karthikeyan, B.. (2018) 'Image Handling and Processing for Efficient Parking Space Detection and Navigation Aid' ,VIT University,India.

6. Akshata, Bhat, Pranali, Narkar,Divya,Shetty and Ditixa,Vyax. (2018) 'Detection of Potholes using Image Processing Techniques' In IOSR Journal of Engineering (IOSRJEN),ISSN (e): 2250-3021, ISSN (p): 2278-8719 ,Volume 2, PP 52-56.

7. Karthik, Muthuraman, Pranav,Joshi and Suraj,Kiran,Raman. (2018) 'Vision Based Dynamic Offside Line Marker for Soccer Games' .

8. Pratik,N.,Patil, Rebecca,J.,Salve,Karanjit,R.,Pawar and Atre,M.,P. . (2018) 'Offside Detection in the Game of Football Using Contour Mapping' In

International Journal of Research in Engineering and Science (IJRES) ISSN (Online): 2320-9364, ISSN (Print): 2320-9356 Volume 6 Issue 4 ,Ver. II ,PP. 66-69

9. Alexander Henderson, Daniel Lai, Tom Allen,(2014) A modern approach to determine the offside law in international football

10. T. D'Orazio (Member IEEE), M. Leo, P. Spagnolo, P. L. Mazzeo, N. Mosca, M. Nitti, A. Distante, (2009) An Investigation into the Feasibility of Real-time Soccer Offside Detection from a Multiple Camera System

11. Michael Wirth , Ryan Zaremba, (2010), Flame Region Detection Based on Histogram Backprojection in 2010 Canadian Conference on Computer and Robot Vision