University of Sheffield

Tracking and re-identification of players in team sports, with a focus on occlusions between players

Joshua B Turver

Supervisor: Yoshi Gotoh

A report submitted in fulfilment of the requirements
for the degree of BSC Artificial Intelligence and Computer Science

in the

Department of Computer Science

July 27, 2023

Declaration

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Name: Joshua B Turver

Signature: JBTurver

Date: 05/12/2022

Abstract

Object detection and tracking has been around and fairly reliable for a number of years now, however, one area it is still growing in is sports tracking and analysis. Therefore the aim of this project is to create a tracking system that is able to track football players in real time in a match. The system should be able to label the players, keeping track of them and re-identifying them as they experience occlusions. Different heuristics will be implemented to try and achieve this re-identification and keep the identities of the players consistent throughout the video. The heuristics used are the bounding box location of the players from frame to frame as well as the shirt colour of the players. When implemented and tested the bounding box location scored an average accuracy of 87.88% and shirt colour scored 82.48%. This is compared to the base level of no heuristic scoring 51.45%. Both of these heuristics performed better and worse under different circumstances on different videos.

Acknowledgements

I would like to thank my supervisor, Yoshi Gotoh, for guiding me through the process of this project, giving direction and support on how to proceed as well as reading over the work and offering advice.

I would also like to thank both my parents Craig and Michelle, in addition to Kaitlyn for supporting me with advice and encouragement throughout the project.

Contents

1 Introduction **1**
1.1 Aims and Objectives 1
1.2 Overview of the Report 2

2 Literature Survey **3**
2.1 Object Detection 3
2.2 Object Tracking 6
2.3 Re-identification 8
2.4 Tracking with a moving camera 9
2.5 Player Identity And Tracking In Sport 10

3 Analysis & Project Requirements **12**
3.1 Dataset 12
3.2 Platform 14
3.3 Ethical, Professional and Legal Issues 15

4 Implementation & Testing **16**
4.1 Dataset Modifications 16
4.2 YOLO Installation 16
4.3 System Classes 18
4.4 Methods Of Player Identification And Re-Identification 19
4.4.1 Re-identification Using Bounding Box Location 19
4.4.2 Re-identification Using Shirt Colour 19
4.5 Methods Of Testing 20

5 Results & Discussion **25**
5.1 Overview Of Results 25
5.2 Performance Of YOLO 25
5.3 Performance Of Identification Using Bounding Box Location 27
5.4 Performance Of Identification Using Shirt Colour 30
5.5 Review Of Aims And Objectives 31
5.6 Building On This Project 32

6 Conclusions **34**
6.1 Future Work 36

List of Figures

2.1 A frame can be input into the RCNN algorithm, from which around 2,000 candidate regions are generated. The regions are then warped and fed into the CNN. From this the region can be classified to say what object it contains. Based on the diagram found on [15] and using frames taken from the dataset intended to be used. 4

2.2 A visual representation of the YOLO algorithm for detecting objects. The input image is split into an S x S grid before bounding boxes are assigned to the possible detections along with the confidence of that detection. There is also a class possibility map to show the possible class of each square. Both of these are then combined to give the bounding boxes that meet a desired threshold with the attached classes of the objects they contain 5

2.3 The two frames shown at the top are consecutive and show the bounding boxes being detected and drawn on. In the bottom image the two frames have been overlayed to show that it is possible to calculate the distance between them. If this distance is less than a set threshold, then they can be considered to be the same player. 9

3.1 Wide angle camera view taken from the dataset. This is the most frequently used angle and usually has many players involved. This is good to test the robustness of the tracking and re-identification as there are lots of players for the system to manage at once and they frequently leave and enter the scene . 13

3.2 Side angle, semi close up shot taken from the dataset. This angle is seen regularly, and normally contains a handful of players. This is useful for me as it tests whether the re-identification and features used in it are reliable enough to correctly re-identify a player from a lower, closer view. 13

3.3 Close up shot taken from the dataset. Usually containing only one or two players. This angle further tests the abilities of the re-identification system as the features and methods used for re-identification will be heavily based on distanced camera shots, therefore it will be interesting to see how the system copes 14

4.1 Table of the results when finding an appropriate confidence level to use, finding a balance between false positives (FP) and false negatives (FN) 17
4.2 Method in the player class for getting the start (top left) and end (bottom right) Euclidean distance between that players bounding box and the bounding box of another player passed into the function 19
4.3 Method in the player class that crops the frame down to the bounding box of the player and aims to find the colour of their shirt 21
4.4 The image of a player as a result of cropping the frame down to the size of their bounding box (left) then applying a mask to the image to remove as much of the pitch as possible (right) 22
4.5 The full results of a single video clip run with the base of no heuristic and then followed by the bounding box location and shirt colour heuristics. Showing the full 'T', 'F', 'X' values for each frame when compared aginst the ground truths as well as the overall accuracy score for each run. 24

5.1 This frame illustrates all three reasons why YOLO sometimes struggles to detect both players in the scene. The image quality is very poor, both players are running quickly causing motion blur and the occlusion on the red player by the player in black means they are much less likely to be detected correctly. 28
5.2 This figure is taken from video 5 in 5.1 where the bounding box heuristic only scored 42.47%. This low score is due to the fact that just before half way through the clip there are 6 undetected frame, after this point the identities have been switched and the system is unable to recover. 29
5.3 This figure is taken from video 10 in 5.1 in which the colour heuristic was able to score 49.30%. This score is likely to be low due to the referee being in black. This causes the system to recognise another colour as the shirt colour which may be leading it to getting the identities mixed up. 31

List of Tables

2.1 A copy of the results of a comparison of different detection models used in the agricultural greenhouse study[7] 6
2.2 A copy of the results of a comparison of different detection models used in the pill identification study[13] 6

5.1 Full results of all 30 video clips run with no heuristic and the two implemented heuristics, with the mean average calculated in the final row 26

Chapter 1

Introduction

Computer vision and real time object tracking are becoming increasingly prevalent in modern life and are used in many fields from traffic monitoring and robotics to medical imaging. These types of technology have made huge advancements in the past couple of decades with the development of more efficient algorithms and faster machines to run them on, to the point where we now have very good accuracy and robustness in many domains. One domain where computer vision can be seen less though is in tracking and analysis of live sports. Especially in team sports, the fast paced and high energy games such as football, basketball or hockey, means that it can be much harder to accurately detect and track players. There are many aspects of sport that make it much harder to perform tracking, one example is that the speed at which players are able to change direction, accelerate or decelerate means it can be easy to lose track of players. In live sports there is also the use of multiple cameras and angles. This poses another problem where the object tracking would need to re-identify the player whenever the shot changed or they moved out of view of the camera. Therefore this project starts by researching the possible solutions to these challenges before implementing a system to track football players during a live match that uses multiple camera angles. This is a system that would be able to be used on different team sports such as rugby, basketball, etc, however for the purpose of this report the focus will be on football as it provided the best dataset.

1.1 Aims and Objectives

The aim of this project is to create a real time object tracking system that is able track the movement of players in sports videos as well as maintain their identities as they moves across the scene and encounter occlusions from each other. The objectives set to achieve this aim are:

- To use a data set of videos taken from Premier League football matches to develop and test the system.
- Implement an object detection system using a tool such as YOLO to correctly recognise

a player.

- Use a tool such as OpenCV to automatically draw a bounding box around the detected player.
- Implement a tracking algorithm or use a tracking by detection method to update the size and shape of the bounding box as it follows the player across the scene.
- Create a function that keeps track of the players identities and attempts to re-identify them after they have been occluded by another player.

1.2 Overview of the Report

The rest of the report will read as follows. Chapter two contains a literature survey of research into areas that are the backbone of the system to be created and the different algorithms and techniques that are used in them. In chapter three analysis of the aims and objectives of the project is performed in more detail. This includes considering some of the tools and platforms that are to be used to achieve them. This chapter also discusses any ethical, professional and legal issues that could arise. Chapter four describes how the system was implemented, from the installation and configuration of YOLO, to the construction of the algorithms behind the identification heuristics and methods of testing the system. Chapter five follows by discussing the results of the two heuristics and how they performed, outlining in which situations they performed well and poorly. This chapter continues with a review of the aims and objectives before moving on to discuss improvements and how the system could be expended in the future. Chapter six concludes the report.

Chapter 2

Literature Survey

2.1 Object Detection

Object detection is a field of computer vision that aims to identify semantic objects and label them in a video or image. Real time object detection takes this a step further by trying to identify them in a video that is being played in real time, while maintaining high levels of accuracy. There are different types of object detection algorithms that can be used in different applications. If accuracy is paramount, then a Region-based Convolutional Neural Networks (RCNN) or Single Shot Detector would be used, however, where speed is more important, then the You Only Look Once (YOLO) algorithm works better. When therefore used in a sports context with video and a fast paced game, an algorithm such as YOLO would be the better option.[3]

In the more accurate method such as R-CNN the image is split into different regions. When the image is split the algorithm should propose around 2000 regions, this could be more but that adds much more computation. These 2000 regions are proposed by using a selective search algorithm on the image. In this the image is split into many sub-regions. The greedy algorithm is then used to reduce the number of regions by creating larger regions from recursively combining similar small ones. This is done until there are the final 2000 left to be used. From this, the regions can be warped into a square and given to a convolutional neural network (CNN). The CNN can be used for feature extraction where the output is in the form of a 4096 dimensional vector containing the feature information. This feature vector can then be used in a support vector machine (SVM) to classify whether or not there is an object in each region. Although this approach gives accurate detection, it takes a very long time to train, having to classify all 2000 regions for each image and when testing, the detection could take around 50 seconds, which can not be used for real time detection.[15]

This algorithm was then improved upon and taken over by the development of Fast R-CNN in 2015. This approach starts in the same way as R-CNN by using a selective search to propose regions and then objects. It differs however by then taking the whole image and

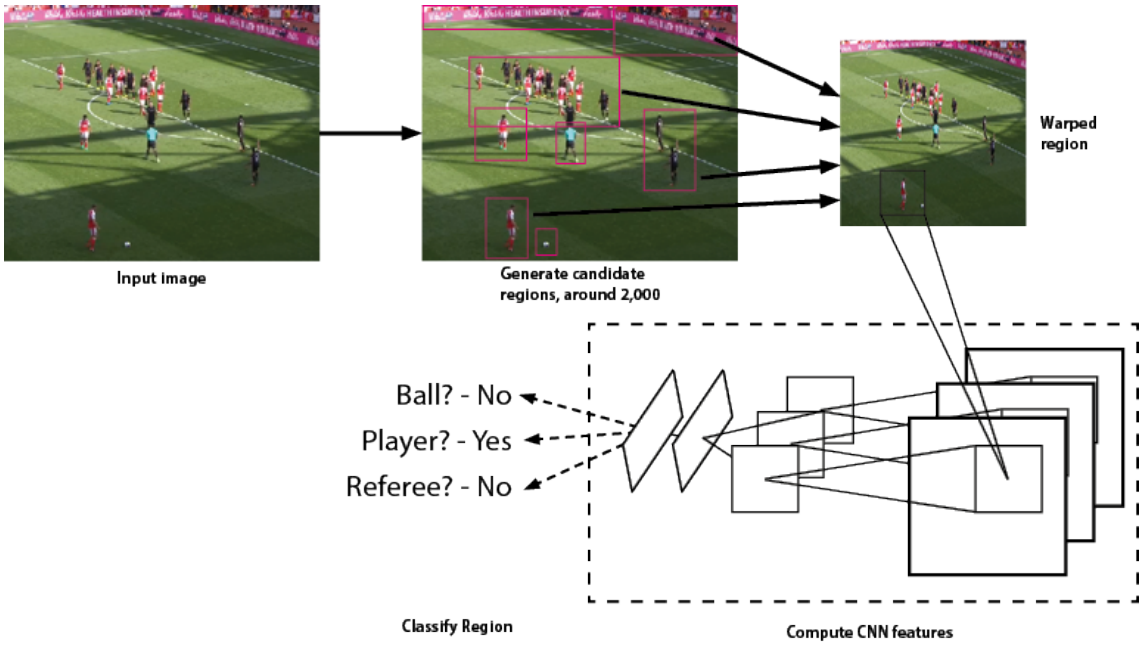

Figure 2.1: A frame can be input into the RCNN algorithm, from which around 2,000 candidate regions are generated. The regions are then warped and fed into the CNN. From this the region can be classified to say what object it contains. Based on the diagram found on [15] and using frames taken from the dataset intended to be used.

running it through a CNN with max pooling to create a convolutional feature map. The object proposals can then each be taken and run through a region of interest (RoI) pooling layer, which is used to extract a feature vector from the feature map. From this, each feature vector goes through a group of fully connected layers before finally reaching the two output layers, one of which gives the encodings for the final bounding box positions. This method brought the time down from 50 seconds to around 2 seconds. This is a huge improvement, however, still not realistic for real time detection.[4]

Faster R-CNN then came not long after, with many improvements being made since. The biggest improvement in this came from the replacement of the selective search based proposal system with another CNN called a region proposal network (RPN). Due to now using another convolutional network, it means that layers from this stage can be integrated better with next stages (detection). This improvement brings the time down dramatically again from about 2 seconds to just 10ms, which can now be considered in real time detection.[1]

You Only Look Once (YOLO) takes a different approach to the previous methods by splitting the image into a grid and treating it more as a regression problem. This means that the algorithm can move straight from the image to calculating bounding boxes and class probabilities. Using a regression method means that there is no complex route where data is

passed from one process or network to another multiple times, the pixel values from the grid are simply fed into a neural network. This is not only good for the speed of the algorithm but also generalisation. The fact that the image is split up less and not processed separately means that YOLO is able to use the whole image and can indirectly encode contextual information when training. This leads to YOLO being better at performing on new domains and more general images than its competitors, for example detecting a person in a slightly abstract piece of art.[12]

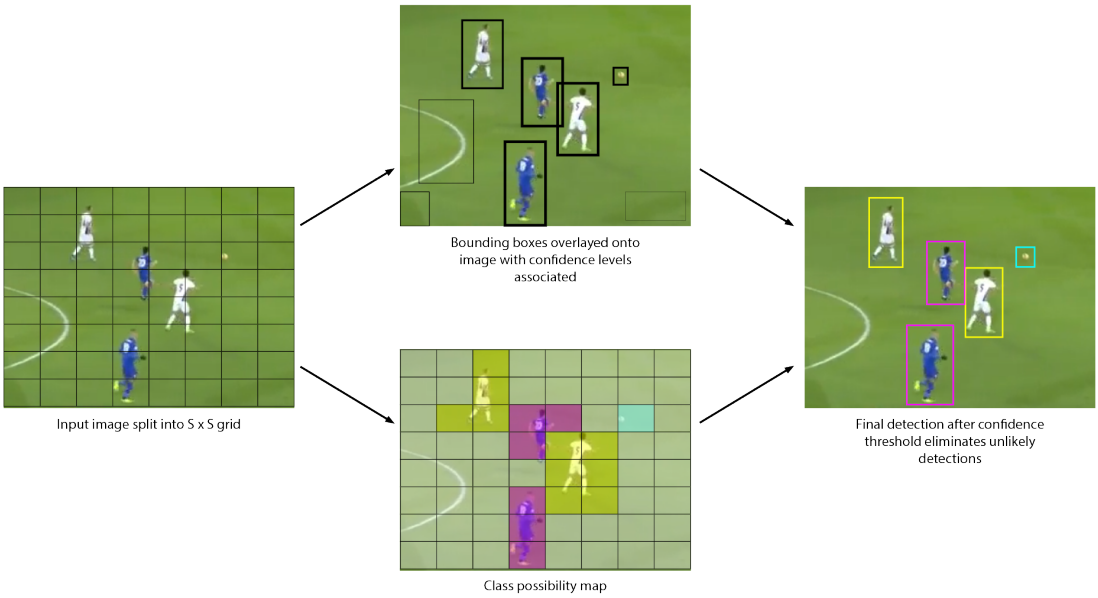

Figure 2.2: A visual representation of the YOLO algorithm for detecting objects. The input image is split into an S x S grid before bounding boxes are assigned to the possible detections along with the confidence of that detection. There is also a class possibility map to show the possible class of each square. Both of these are then combined to give the bounding boxes that meet a desired threshold with the attached classes of the objects they contain

The mAP metric stands for the mean average precision and is often used to compare detection algorithms. The average precision measured as the area under the precision recall curve, where precision is defined as the accuracy of the predictions a system has made, the percentage of predictions that are correct and the recall is defined as a measure of how well the positive predictions are found, how many predictions it takes to find a certain percentage of the positive cases.

When comparing the two approaches, YOLO and Faster R-CNN, there are many studies that have tried and tested them both for different applications. Lu Tan et al. [13] compared the algorithms in the context of detecting and classifying different pills. They implemented both methods as well as single shot detection (SSD) and discovered that R-CNN has a higher mean average precision percentage (mAP%) with 87.69% compared to YOLOv3 with 80.17%.

In the case of real time detection and classification however, (real time being considered to be running at more than 30 frames per second (fps)), YOLO came out on top by a substantial margin, YOLO ran at an average of 69 fps compared to 3 fps achieved by R-CNN. This shows that while R-CNN has better precision in labelling the correct pill, it is far from being able to work in real time. Similar was also seen in a comparative study by Min Li et al. [7] This study was making comparisons between Faster R-CNN, YOLOv3 and SSD when used to detect agricultural greenhouses in high resolution satellite images. The results from this showed YOLO to outperform R-CNN in both precision and speed. The mAP% for YOLO was 90.4% compared with 86.0% for R-CNN as well as YOLO running at 73 fps compared to the significantly slower 12 fps of the R-CNN network. The results seen in these studies, in addition to others [6], show that YOLO outperformed R-CNN by a considerable margin in speed but sometimes also precision, depending on the domain it is used in.

	Faster R-CNN	YOLO	SSD
mAP	86.0%	90.4%	84.9%
FPS	12	73	35

Table 2.1: A copy of the results of a comparison of different detection models used in the agricultural greenhouse study[7]

	Faster R-CNN	YOLO	SSD
mAP (normal samples)	87.69%	80.17%	82.41%
mAP (difficult samples)	79.63%	78.52%	78.69%
FPS	3	69	41

Table 2.2: A copy of the results of a comparison of different detection models used in the pill identification study[13]

2.2 Object Tracking

It is one thing to be able to detect an object, but the much bigger challenge is to be able to track it through a video scene. As the object/s in the scene move there are many things to consider that would be no problem for a human to process but could very easily throw off a machine. These could include problems like lighting, camera resolution, occlusion by other objects, changing scale of objects or even a slight motion blur due to the change in speed of the object or camera. Due to these challenges a system for tracking needs to be both robust and adaptable to these changes in the scene while balancing this with its ability to be used in real time.

Depending on the application of the tracking system, there are multiple methods that can be implemented. One of these is tracking using the features of the object, which could

include, colour, texture or the optical flow of the object. The analysis of a colour histogram can be used to locate distinctive colours in an image and then follow this pattern of colour across the scene. This however is not very accurate as it fails to take into account that two objects could have the same colour histogram but not be the same, while also neglecting the size and shape of the object. Similarly, the texture of an object can also be detected using a preprocessing technique to extract the texture feature. This can be used in conjunction with colour to better distinguish between similar objects[17].

Another method that can be used is segmentation. This is the process of trying to split the foreground from the background, the foreground being the moving objects that we want to track and the background being the stationary or irrelevant objects in the scene. This method can be split into different approaches with one being a bottom-up based approach. Bottom-up takes the process of segmentation and tracking separately with the image first being split into regions, with the regions containing foreground features subsequently being tracked. A problem with this method is that any errors in the segmentation of the image would be carried through to the tracking stage. This is remedied by using a joint based approach where the segmentation process and tracking method are carried out in tandem[17].

Estimation based methods can also be used in tracking, by representing the object as a state vector, with the vector showing the position and velocity of the tracked object. This method consists of two models that work together in tracking the object. Firstly a state model is used to predict the next position of the object based on the vector of its last position. Then an observation model is used to observe the current object and update the predicted position. These two models are used on each frame of the video, and the specific filtering techniques that could be used are a Kalman filter or a particle filter[17].

Learning based methods are also another choice for tracking an object. One of these sub-methods is to use a discriminative tracker which models tracking as a classification problem. It tries to classify different aspects or features of the scene into either being a target object we're looking for or part of the background. For this, deep learning can be used and this can be further split into the categories of feature extraction and end-to-end. In feature extraction the raw image is converted into meaningful and more easily worked on features. These could include converting to gray-scale and using raw pixel values as features or using a histogram of oriented gradients (HOG) as a feature to help filter out basic shapes[11]. These are only a few of multiple features that can be extracted and combined to optimise tracking for a specific domain. End-to-end then trains a network to do both the feature extraction and then candidate evaluation. An example of this type of method is a Siamese tracker, where the network will take two input images and measure the similarity between them. This is done using the features that have been extracted from the images and allows the network to determine whether an object is in both. If the target object is in both but has moved then the bounding box can be moved to the new location also to complete

the tracking. This process can be repeated for pairs of frames until the target object is no longer deemed to be in both images and the tracking stops.

2.3 Re-identification

When objects or people move across a busy video scene, they often become occluded. In the case of sport, this could be players running past each other and one player being temporarily obstructed. This can also happen when two players are close together like when challenging for the ball, potentially causing a slight occlusion and leading the detector and tracker to classifying it as one object. Therefore there must be a way to identify that a player is the same person that re-emerges from the occlusion as the one that went in.

With such great advancements in recent object detection, a common way to track objects is to use detection as the foundation. The object detection algorithm can be run on each frame of the video to determine a list of objects. Then all the tracking algorithm has to do is to match up the detected objects between the two frames in order to keep track of them.

The most important part of being able to re-identify objects is determining what sets them apart and how to distinguish between them. An early proposal of how to do this was to have a re-identification method that was separate to the detection and tracking methods. Once an object had been detected it could be cropped out and fed into the re-identification module, where its features could be extracted and identification checked. This however is time consuming and computationally intensive as the process is separate and needs to be run on every object in every frame. The re-identification module is usually trained to classify each object into one class. This however does not scale well. When there are many different identities that all need to be checked, training and testing times would be greatly increased. It also means that it is very difficult to find training data that is sufficient. The training needs to be supervised, so any data set needs to be well annotated and plentiful, and this is not easy to come by.[9]

Unsupervised methods have also been proposed to handle re-identification. This has been used in re-identifying vehicles in traffic cameras and the method consists of combining the detector and re-identification module into one network. The approach of Yuhao Xu et al [16] achieved this by using the faster RCNN algorithm and adding an extra track branch to handle re-identification. For the re-identification module to work however, a triplet is needed that consists of the input image, a positive example of the intended target and a negative image that does not contain the target. To get around this, the previous and subsequent frames of the video can be used instead. Due to having the method integrated into the detection and tracking network, the computational expense is less than if they were separate modules, thus increasing the speed at which the video can be processed. Another simple approach that was

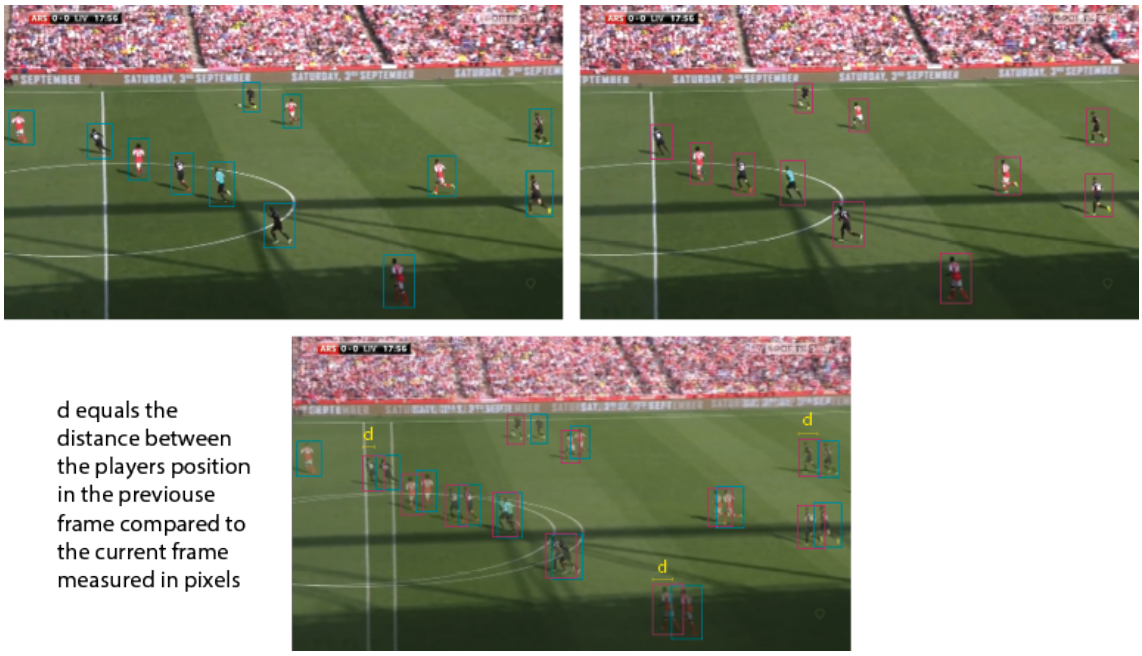

Figure 2.3: The two frames shown at the top are consecutive and show the bounding boxes being detected and drawn on. In the bottom image the two frames have been overlayed to show that it is possible to calculate the distance between them. If this distance is less than a set threshold, then they can be considered to be the same player.

taken to track and identify the objects (in this case vehicles) was to measure the distances between the detected objects from one frame to the next. If there is an object detected in the next frame that is only a few pixels away from one detected in the previous frame, it is more than likely the same object. This can be seen in figure2.3

2.4 Tracking with a moving camera

When using a static camera there are many techniques that can be used to detect and track an object. One of these, as mentioned before, is background subtraction. This relies on the fact that the camera never moves, meaning many of the background features stay the same and objects move across the scene relative to them. This however is not possible with a moving camera because all the background is also constantly changing. A moving camera also creates the issue that objects will appear to change shape more as the camera angle and perspective changes as well as changes in lighting, colour and blur. All these factors mean there needs to be a new approach taken to tracking using a moving camera.

Tao Liu et al [10] proposed the creation of a 3D tracking system that uses adaptive ground plane estimation to better track the background and in turn the objects moving in the scene. When the sequence of video frames enter the system, two parallel procedures occur, one being

for detection and the other for feature extraction to handle scene motion. The object detection branch is based on a pre-trained model so is a less significant change in this process, however as said before it does need to be robust and use appropriate features to be useful when used with a moving camera. On the other branch, feature extraction and more specifically the Harris corner feature detector is used. This feature extraction technique can be used in real time and is robust enough to aid in estimating the ground plane. Another key part needed to estimate the ground plane is the essential matrix. The essential matrix is a combination of the rotation matrix and the translation matrix, where both of these relate to the movement of the camera. With both the feature extraction and essential matrix calculated, the ground plane for that frame can be calculated. The parallelization is recombined at the pose estimation stage. The estimated ground plane is used with each of the detected objects to turn their 2D locations in the scene into estimated 3D locations. The next stage is to use constrained multiple kernel (CMK) tracking to update the new position of the target object. This is done by maximising the similarity between the target object and the candidate bounding boxes. A Kalman filter is used after each iteration of the CMK tracking stage to update the tracking result and improve reliability. An additional bonus to this method is that because the objects are tracked in an estimated 3D space, the problem of losing an object due to occlusion is reduced, due to there being an extra axis to track the object's motion on.

2.5 Player Identity And Tracking In Sport

There have been multiple attempts to overcome the problems in player tracking and classification, issues such as occlusion, varying size of players, similar player appearance, etc. One of these was by Jia Liu et al[8], who split players into three classes, team A, team B and referee. Their approach starts by filtering the input video, firstly by eliminating everything outside of the playing field. The second filter is to then use background subtraction to remove the field and create a mask to only show the players. These two steps are good for multiple reasons, one being it reduces the amount of processing needed on the region of interest later, therefore speeding up detection and two, it reduces the chance of false positives due to noise (people in the crowd, stewards, camera operators, etc). Once this is done, the detector is used to scan across the different regions of the image at multiple scales. By doing this, each player will usually be detected multiple times. Detections that are close to each other can then be merged into one, which allows for the discarding of false positives if there are single bounding boxes on their own. The next stage is to label each of the detections. The system uses a bag of features to model the players and is only trained using the top half of a player. This region of interest can then be converted into CIE-Luv colour space. A Gussian mixture model is then estimated to form clustering. The centres of these are called prototypes and neighbouring ones can be merged if their centre distance is less than a predefined threshold, these are called meta-prototypes. Using the upper body region of the player, the pixel values can be put into a histogram according to the meta-prototype they correspond to.

Ruiheng Zhang et al[18] went further than this and tried to identify and keep track of the individual players. This is very hard because commonly used features for differentiation can not be used in this domain. For example colour can not be used because players on the same team all wear the same kit. Unique movement style or gait can also not be used as the players change speed and direction too frequently for this to be relied upon. The proposed method for dealing with these problems is to create a system consisting of three parts. The first being a deep player identification model known as DeepPlayer. This consists of combining a cascade mask RCNN model and a PoseID model to identify players. The use of a coarse-to-fine region-based CNN model allows the RCNN to recognise both player's team and shirt number. The second aspect of the system is an Individual property occupancy map (IPOM) which is used to help solve the problems of occlusion and noise. This function can take both known and ambiguous player ID's and output the probability of the ID for each player, as well as their 3D location in the scene. The third branch of this system is a K-shortest path model. This model is designed to prevent the switching of identifications between players and works by using a player ID correlation coefficient to measure the similarity between two nodes in neighbouring frames.

Chapter 3

Analysis & Project Requirements

3.1 Dataset

The dataset chosen for this project was obtained from soccer-net.org, which is a website setup to be a constantly growing and evolving large-scale dataset of European football matches. The dataset currently consists of 500 complete football matches with a total duration of 764 hours. It has been used for programming challenges for multiple years in the fields of computer vision, object detection, tracking and re-identification and has also been used in multiple publications. Using this dataset means that the results obtained when building this system can be analysed and compared to that of other systems to assess how well it keeps the correct identity of a person through a scene, this could be through a percentage of how long the identification is correct during the video clip or until the player leaves the scene. However due to the large number of hours of footage, the amount of space needed to store the video dataset would be around 1TB. For this project that would be an unreasonable amount of video, that on the hardware available is not realistic to store or process. Therefore for the purposes of this project, the dataset will be reduced significantly to create a subset of the original. Some of the varying camera angles from the dataset can be seen in figures, 3.1, 3.2, 3.3

The dataset has also been used in papers closely related to this project before. "SoccerNet-Tracking: Multiple Object Tracking Dataset and Benchmark in Soccer Videos" is a paper focused on tracking player and ball movements in order to collect team statistics. This includes things like estimating player travel distance, possession and passing, as well as the formation the teams are playing[2]. Another paper, "SoccerNet: A Scalable Dataset for Action Spotting in Soccer Videos" uses the same dataset but to localise specific events such as goals, yellow cards and red cards. These can be sparse events in a 90+ minute game of football so this paper works on pinning down which minutes these events occur in[5].

Figure 3.1: Wide angle camera view taken from the dataset. This is the most frequently used angle and usually has many players involved. This is good to test the robustness of the tracking and re-identification as there are lots of players for the system to manage at once and they frequently leave and enter the scene

Figure 3.2: Side angle, semi close up shot taken from the dataset. This angle is seen regularly, and normally contains a handful of players. This is useful for me as it tests whether the re-identification and features used in it are reliable enough to correctly re-identify a player from a lower, closer view.

Figure 3.3: Close up shot taken from the dataset. Usually containing only one or two players. This angle further tests the abilities of the re-identification system as the features and methods used for re-identification will be heavily based on distanced camera shots, therefore it will be interesting to see how the system copes

3.2 Platform

The first step to creating this system is to utilise a robust detection system. For this the YOLO detection algorithm will be implemented using python. Python is a very widely used language, especially in the machine learning field, so there are more likely to be example implementations and information on the topic. YOLO is also a very developed and robust system at this stage, and is also built in a way that makes it more flexible to ambiguity and new domains. This is good as it should reduce the chance of missing detections or false detections due to motion blur and the movement of the camera.

In conjunction with using YOLO, OpenCV will also be used in python to handle video and image processing. This is a well developed python library that easily splits video up into frames to be used for detection and processing. There is also the ability to create masks and filter colour in OpenCV which could be very useful for background subtraction and feature extraction. The main reason for using OpenCV however is to draw and display the bounding boxes that have been calculated by YOLO.

The objective of being able to track the players across the field as they move could be done using a track by detection method. This would involve running the detector on each frame and then taking them in pairs. If there is a bounding box located in two consecutive frames

and the distance between them is less than a defined threshold then these two detections can be considered to be the same object. This would be done for each detection in each frame and therefore would track each player through the scene.

The final objective of this project is to create a function that keeps track of the players' identities. When a new player is detected, for example at the start of the video, they would be given an ID (possibly attached to a more meaningful label). If the player then goes out of shot and then returns, it would be ideal if they are then able to be picked up again as being the same player as before. This is something that can be built from the principles of tracking but taking it a step further. For example, calculating that bounding boxes that are close to each other in one frame to the next, could be taken a step further to say that a player that leaves shot last from the left, is the most likely to return back from the left, due to the distance they have travelled. Shirt colour would also play a small part in this process as it narrows the possibilities down by at least 50%, as well as many other possible features that could be used.

3.3 Ethical, Professional and Legal Issues

There are no ethical or professional issues that have arisen at this time. The project will make use of platforms, tools or code that are open source and are available to use for the purposes proposed, as to not break any copyright laws.

Chapter 4

Implementation & Testing

4.1 Dataset Modifications

When considering the size and scale of this project compared to that of the dataset, the decision was made to reduce the dataset in size. This was done by taking 10 games from the most recent available season which is the 2016-2017 Premier League season. From these 10 games, 3 lots of 1-3 second clips were taken specifically focusing on situations where players cross each other's path. This gives a total of 30 different video snippets for the system to be tested and analysed on. This is sufficient to to calculate an average for each of the proposed methods of re-identification so that they can be compared against each other. The use of 10 different games means there will be a variety of different teams with different coloured kits, thus making the system more generalisable. The system therefore has to be able to track players without relying solely on a hard coded colour of kit, but instead be able to detect the colour. Each video clip is cut down to 1-3 seconds in order to focus specifically on the task of re-identification after occlusion and prevent any interference from other players.

4.2 YOLO Installation

To detect players in the video the chosen method was to install YOLO, and more specifically the latest version at time of writing, YOLOv8 from Ultralytics [14] . It was easily integrated into python by using a pip install and importing it in the program as well as downloading their pretrained model that has been trained on the COCO dataset, a very well known and widely used dataset of images. The library was first implemented on static images in order to test its functionality, before then moving on to video. To do this OpenCV has been used to load a video, before taking each frame at a time and running it through the YOLO detector. When running the detection method the parameters it takes are, an image (or in the case of this project a single frame of the video) and a confidence level. Determining a suitable confidence level was done by trial and error in order to find a balance between too many false positives and too many false negatives. The detector was tested on single frames from 10

	0.3		0.35		0.4		0.45		0.5		0.55		0.6		0.65		0.7		0.75		Total people
	FP	FN	FP	FN	FP	FN	FP	FN	FP	FN	FP	FN	FP	FN	FP	FN	FP	FN	FP	FN	
Image 1	4	0	3	0	2	0	2	0	1	0	1	3	1	4	0	9	0	14	0	17	20
Image 2	2	2	2	2	1	2	1	2	0	2	0	4	0	8	0	11	0	14	0	16	16
Image 3	1	1	1	1	1	1	1	1	1	2	1	2	1	2	0	7	0	8	0	12	13
Image 4	0	3	0	4	0	5	0	5	0	5	0	5	0	6	0	7	0	9	0	12	16
Image 5	0	1	0	1	0	1	0	2	0	3	0	6	0	7	0	8	0	12	0	14	17
Image 6	0	1	0	1	0	1	0	1	0	1	0	1	0	3	0	7	0	8	0	11	14
Image 7	1	2	0	2	0	3	0	3	0	3	0	4	0	6	0	8	0	11	0	13	15
Image 8	1	1	1	1	0	2	0	2	0	2	0	3	0	5	0	7	0	12	0	14	15
Image 9	2	1	2	1	1	1	1	2	1	3	0	4	0	7	0	13	0	16	0	19	19
Image 10	1	2	0	2	0	3	0	4	0	5	0	7	0	8	0	12	0	14	0	16	18
Sum	12	14	9	15	5	19	5	22	3	26	2	39	2	56	0	89	0	118	0	144	163
Mean	2.4	2.8	1.8	3	1	3.8	1	4.4	0.6	5.2	0.4	7.8	0.4	11.2	0	17.8	0	23.6	0	28.8	32.6
Combined Mean	2.6		2.4		2.4		2.7		2.9		4.1		5.8		8.9		11.8		14.4		16.3

Figure 4.1: Table of the results when finding an appropriate confidence level to use, finding a balance between false positives (FP) and false negatives (FN)

different video clips before they had been cropped to create the test set. It was done on the uncropped image in order to include more players, giving more data points to show at what threshold players were more likely to be detected correctly. From these tests it could be seen that the optimum confidence level was between around 35-40%, therefore for the purposes of this project, a confidence level of 40% will be used at the start, monitoring the accuracy this yields when used in practice as the system is developed. The tests resulted in quite a low confidence level which is the result of the image quality of the video and the movement of the players. The video resolution being low and the varying sizes of the players due to their distance from the camera, in addition to some of the frames containing motion blur if the players are moving at speed, it can be hard for the detector to pick up on some of the players. Therefore 40% came out to be the optimum confidence level in order to detect the most players.

The YOLO algorithm was run on the cropped video clips and the bounding boxes of the two players were drawn onto the frame using OpenCV. When doing this and playing the clip through, it became evident that when one player completely covers another player, YOLO is only able to detect one person and therefore IDs would not be able to be given and the system would have to pick back up when two detections were present again. More on this in the next chapter.

The return value from the YOLO detector contains the top left and bottom right coordinate of the bounding box, the id number for the class of object the algorithm thinks it is (person, bicycle, etc) and a confidence level as to how sure the detector is that the object in that region belongs to that class. The system needed to be built around these values and therefore the following classes have been set up to create the foundations of the system.

4.3 System Classes

A class called Player is used to store the information about a single player detection in a frame. The class has 4 attributes, ID number to identify the player, the bounding box coordinates and colour which contains the rgb value of the players shirt. This class contains many helper methods for comparing players, for example comparing the difference in location between two player's bounding boxes, calculating and comparing the colour of player shirts and checking to see if two instances of a player object are equal.

Building on this, the class Frame has been created to hold all of the information about a single frame. This class has the attributes, index, to signify the frame's position in the video sequence, frame_image which contains the numpy array of rbg values that represent the image, frame_anot which is the same as frame_image but is annotated with the bounding boxes and predicted player ID numbers. The final attribute of this class is a list of player objects that hold the information about the players detected in that frame. This class also contains helper methods that are useful to check the equality of frames, annotate the frame with each players bounding box and ID as well as getting the highest ID number used so far and the number of players in the frame (which should always be two but can very with false or missed detections from YOLO) The main implementation of the focus of this project is contained as methods in the Frame class as well and run from a separate class called Run.

The Run class focuses on running the system on one video clip from start to finish, performing detection, identification and annotation on each frame. The information stored as attributes in this class are, 'model' is the specified model to be used by YOLO, 'vid_path' which is the path to the video clip the system is to be run on, 'frames' is a list containing objects of the type frame and is appended with new frames as the system processes the video. The attribute 'method' denotes the identification method that should be used (none, by bounding box difference or by colour) and the final attribute of the class is 'show' which is a boolean value to determine whether to output the annotated frames to the user.

When a new Run object is created, the video clip can be analysed by calling the 'auto_run' method on it. This uses OpenCV to open the video and then a loop to get the next frame and increment the index until the video is over. For each new frame a Frame object is created with its index and the numpy image array. On this new object the method 'run_detection' is called. This method runs the YOLO detection algorithm on the frame and iterates through the detected players, creating new Player objects containing the players bounding box and running the 'get_colour' method to get the colour of their shirt for later use. If the index of the frame is zero, meaning this is the first frame, the method will also set the players id numbers (starting from 0) in the order they appear in the params outputted by YOLO. These IDs will be what the system is trying to maintain and identify correctly in the following frames. The 'auto_run' method checks if the frame is the first frame and if so annotates and displays it, otherwise the method 'determine_ids' is run on the frame taking the argument of the

```
def bb_diff(self, sec_player):
    s_euclid = math.sqrt(((sec_player.bound_box[0] - self.bound_box[0]) * (sec_player.bound_box[0] - self.bound_box[0])) +
                         ((sec_player.bound_box[1] - self.bound_box[1]) * (sec_player.bound_box[1] - self.bound_box[1])))

    e_euclid = math.sqrt(((sec_player.bound_box[2] - self.bound_box[2]) * (sec_player.bound_box[2] - self.bound_box[2])) +
                         ((sec_player.bound_box[3] - self.bound_box[3]) * (sec_player.bound_box[3] - self.bound_box[3])))

    return (abs(s_euclid)), (abs(e_euclid))
```

Figure 4.2: Method in the player class for getting the start (top left) and end (bottom right) Euclidean distance between that players bounding box and the bounding box of another player passed into the function

identification method that should be used (see section 4.4). With IDs predicted, the frame can be annotated, displayed and appended to the list of completed frames before moving on to the next one.

4.4 Methods Of Player Identification And Re-Identification

4.4.1 Re-identification Using Bounding Box Location

The algorithm for re-identifying a player using their bounding box coordinates starts by finding the highest ID number that has been given to a player so far. For each of these IDs the system needs to find the player that was most recently assigned it so that their bounding box can be compared with those in the current frame. Once the most recent instance of a player with each ID is found and compiled into a list, the system loops through each player in the current frame and compares them to each player in this list. The method of comparison is to find the difference between the two bounding boxes of the players. To do this the Euclidean distance

$$difference = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

is found between the two top left corners of the bounding boxes and the two bottom left corners of the boxes, which are then added together figure 4.2. This distance returned by the comparison between the player from the current frame and the player from the past frame is used to find the smallest distance and therefore the closest match. Once all of the players have been compared, whichever returns as the closest match can be assumed to be the same player. Once this pairing has been made, this previous player instance can be removed from the list so as to not be paired with another player in the current frame and duplicate IDs be created.

4.4.2 Re-identification Using Shirt Colour

The algorithm implemented to re-identifying players using the colour of their shirt starts by determining the area of their shirt. An original approach to this was to assume that the

player's torso would be in the centre of the bounding box. Therefore it would be possible to find the centre pixel of the player's bounding box and take the colour value from this pixel or the average from the surrounding pixels. This however was found not to be the case in some circumstances depending on the players body position, for example if the player's leg is stretched out in front of them, their bounding box will be stretched to accommodate this and the centre of the box may therefore be part of the pitch and the colour be detected inaccurately. A second method was to start by cropping the frame to a specific region of interest, in this case the player's bounding box. A mask was also created to filter green and inverted so that when combined with the region of interest using a bitwise and operator from the OpenCV library the resulting image would no longer contain any green in the specified range. This was done to remove the pitch so that better focus was on the colours relating to the player. With the player highlighted and the background removed, kmeans clustering is used to find the most common colour. For this the number of clusters for this is set to two and was chosen because even though we only want the most common colour, once the image has been filtered, most of the pixel values are 0,0,0 (black). This would therefore be returned as the most common colour value which is not what is required, therefore the second most common colour returned is the colour of the players shirt. With this colour determined it can then be set in the player object so that it can be used to compare to players in previous frames. This comparison is done by the same method as used in the bounding box location method. For each ID the most recent player with that ID is found and compared to each player in the current frame. Whichever player in the current frame matches best is then given the ID associated with the player from the past frame. Shown in figure 4.3

4.5 Methods Of Testing

To carry out testing it is necessary to know the ground truth values for the player identities in each video clip. Therefore a class was created to be able to iterate through each frame and label each frame before recording these ID values in a file. When the YOLO detection is run on a frame a list is returned containing the different detections. Therefore when the first frame is run, the players can be given ID starting from zero in the order they appear in the list. Once these IDs have been established, these are what must be maintained throughout the video clip. When testing the system on a single clip, the run class maintains a list of frames and the players in them. This means that once the system has finished running each heuristic, the manual ID labels can be compared against those produced by the system and give a true or false value as to whether the system maintained the IDs correctly.

When comparing the IDs produced by the system to the ground truth IDs the output for each frame will be either 'T', 'F' or 'X'. 'T' represents true and that all IDs produced by the system match up to the ones that have been labelled manually. 'F' denotes false and that the IDs have been switched for this frame and do not match the ground truth values.

```python
def get_colour(self, frame):
    # Crop the frame image to the size of the players bounding box
    roi = frame[int(self.bound_box[1]):int(self.bound_box[3]), int(self.bound_box[0]):int(self.bound_box[2])]
    roi_hsv = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)

    # Green threshold and mask
    lower_green = np.array([40, 25, 25])
    upper_green = np.array([70, 255,255])
    mask = cv2.inRange(roi_hsv, lower_green, upper_green)

    # Invert the mask and apply it to the image
    inv_mask = cv2.bitwise_not(mask)
    result = cv2.bitwise_and(roi, roi, mask = inv_mask)
    result_rgb = cv2.cvtColor(result, cv2.COLOR_HSV2BGR)

    # KMeans used to put colours into two clusters
    clt = KMeans(n_clusters = 2, init='k-means++', n_init=1)
    perc, centers, palette = self.palette(clt.fit(result_rgb.reshape(-1, 3)))

    # Sort the cluster centers to find the most abundant colour and its percentage
    centers = np.array(centers)
    perc = np.array(perc)
    inds = (-perc).argsort()
    sorted_centers = centers[inds]
    sorted_perc = perc[inds]

    # Take the shirt colour to be the second most common colour
    # first most common colour is black due to filtering out the pitch
    colour = [int(value) for value in sorted_centers[1]]
    self.colour = colour
```

Figure 4.3: Method in the player class that crops the frame down to the bounding box of the player and aims to find the colour of their shirt

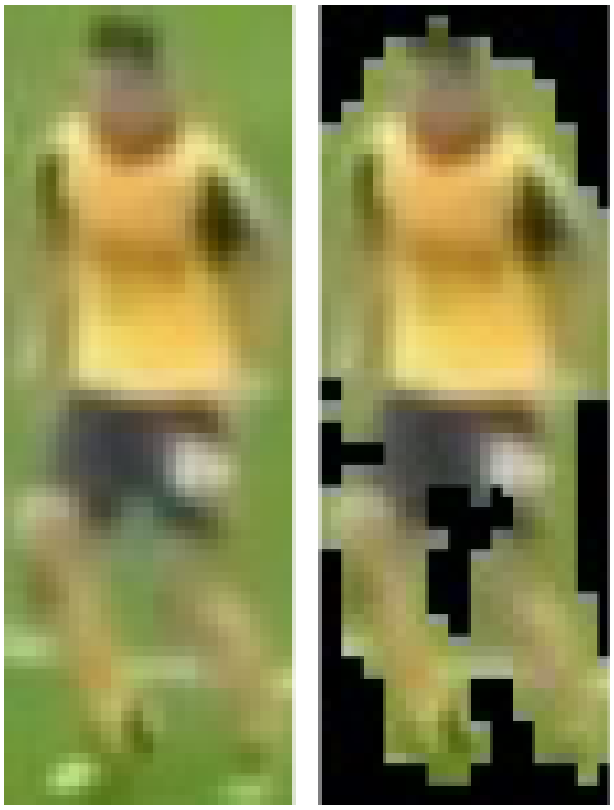

Figure 4.4: The image of a player as a result of cropping the frame down to the size of their bounding box (left) then applying a mask to the image to remove as much of the pitch as possible (right)

'X' denotes that the YOLO detection run on the frame (before any re-identification has been considered) has failed. This could mean either that one or both of the players in the frame has not been successfully detected, or that more than the two players in the frame has been detected, this could be the false positive detection of a shadow or other anomaly.

When testing the heuristics implemented there must be a base level test to compare them to and determine their effectiveness. This was done by running the system so that it assigned IDs to the players purely on the order in which they were detected and returned in a list by YOLO, the first in the list being ID number 0, and so on. As there are two players in the test clips it would be reasonable to assume that this method would get the identification of the players correct around 50% of the time. Each of the 30 clips will be run using this base method with no heuristic, before using both the bounding box location and shirt colour heuristics.

With the true/false value recorded for each frame, the 'T' values can be summed up and divided by the number of frames minus the frames in which detection failed.

$$percentage_maintained = \frac{total_true}{total_frames - dropped_frames}$$

This determines the percentage accuracy of the heuristic. This can also be done with the 'X's to find the percentage of frames that YOLO was able to correctly detect both players in.

```
Running with tracking method: NONE
Each frame status:
--
 ['T', 'T', 'F', 'F', 'T', 'F', 'F', 'F', 'T', 'F', 'T', 'F',
'F', 'F', 'F', 'F', 'F', 'F', 'X', 'X', 'X', 'F', 'F', 'F',
'F', 'F', 'F', 'F', 'F', 'F', 'F', 'F', 'F', 'F', 'F', 'T',
'F', 'F', 'T', 'X', 'F', 'X', 'X', 'X', 'X', 'X', 'X', 'X',
'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'T', 'T', 'T',
'T', 'T', 'T', 'F', 'F', 'F', 'T', 'T', 'T', 'T', 'T', 'F',
'T', 'F', 'F']
--
Total number of frames: 75
Percentage Of Yolo Complete Detections: 73.33%
Percentage Of Yolo Detections Where ID Maintained: 34.55%


Running with tracking method: BB_DIFF
Each frame status:
--
 ['T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T',
'T', 'T', 'T', 'T', 'T', 'T', 'X', 'X', 'X', 'T', 'T', 'T',
'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T',
'T', 'T', 'T', 'X', 'T', 'X', 'X', 'X', 'X', 'X', 'X', 'X',
'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'T', 'T', 'T',
'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'F',
'T', 'T', 'T']
--
Total number of frames: 75
Percentage Of Yolo Complete Detections: 73.33%
Percentage Of Yolo Detections Where ID Maintained: 98.18%


Running with tracking method: COLOUR
Each frame status:
--
 ['T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T',
'T', 'T', 'T', 'T', 'T', 'T', 'X', 'X', 'X', 'T', 'T', 'T',
'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T',
'T', 'T', 'F', 'X', 'T', 'X', 'X', 'X', 'X', 'X', 'X', 'X',
'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'F', 'F', 'T',
'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'F',
'T', 'T', 'T']
--
Total number of frames: 75
Percentage Of Yolo Complete Detections: 73.33%
Percentage Of Yolo Detections Where ID Maintained: 92.73%
```

Figure 4.5: The full results of a single video clip run with the base of no heuristic and then followed by the bounding box location and shirt colour heuristics. Showing the full 'T', 'F', 'X' values for each frame when compared aginst the ground truths as well as the overall accuracy score for each run.

Chapter 5

Results & Discussion

5.1 Overview Of Results

The results of all the tests carried out can be seen in full in table 5.1. The overall average percentage accuracy can be seen to be substantially higher when using either of the heuristics than when not using one. When running the system with no heuristic it can be seen that it has an average accuracy of 51.45%, this is as predicted in section 4.5 due to the players being labelled with the index in which they appear in the list of detections returned by YOLO, and that there are only two players in the frame so the chance of getting the IDs correct would be assumed to be 50% When using the heuristic of the difference in bounding box location between frames there was seen to be a significant improvement in maintaining player identities with an average accuracy of 87.88%. Using the heuristic of shirt colour was marginally less accurate that bounding box location but still a substantial improvement over no heuristic with an average accuracy of 82.48%.

5.2 Performance Of YOLO

As the YOLO detector makes the foundation for this project and which the focus is built upon, as in some cases affected by, it was important to test the accuracy of this. The average amount of frames that YOLO was able to correctly identify both players in came out to be 82.58%. This average came from a set of results that were close together with the scores being mostly between 75%-90%. There were however a few outliers with scores as high as 100% and some as low as 37%. There are multiple aspects that could cause these outliers in specific videos.

One struggle that YOLO faces when detecting the players is that the video clips are of very low quality and resolution. When cropping the videos down to include just the two players, this decreases the size of the players dramatically to only tens of pixels across, especially if they are on the opposite side of the pitch and further away from the camera. The further

Video Number	Num Frames	YOLO Detection %	Heuristic None	Heuristic BB Diff	Heuristic Colour
1	36	83.33%	66.67%	100.00%	96.67%
2	35	37.14%	69.23%	100.00%	76.92%
3	96	89.58%	59.30%	100.00%	97.67%
4	50	88.00%	47.73%	100.00%	52.27%
5	85	85.88%	43.84%	42.47%	69.86%
6	37	83.78%	61.29%	100.00%	58.06%
7	34	94.12%	40.62%	100.00%	90.62%
8	29	72.41%	61.90%	100.00%	95.24%
9	36	58.33%	14.29%	100.00%	80.95%
10	75	94.67%	40.85%	98.59%	49.30%
11	75	92.00%	46.38%	100.00%	84.06%
12	33	96.97%	46.88%	100.00%	90.62%
13	60	75.00%	33.33%	85.56%	97.78%
14	75	73.33%	34.55%	98.18%	92.73%
15	44	86.36%	65.79%	100.00%	89.47%
16	24	91.67%	63.64%	100.00%	81.82%
17	24	83.33%	65.00%	45.00%	80.00%
18	24	100.00%	41.00%	100.00%	100.00%
19	42	85.71%	58.33%	100.00%	100.00%
20	27	70.37%	63.16%	47.37%	94.74%
21	24	87.50%	61.90%	100.00%	100.00%
22	34	79.41%	18.52%	100.00%	85.19%
23	50	74.00%	51.35%	97.30%	97.30%
24	49	77.55%	84.21%	100.00%	52.63%
25	24	87.50%	52.38%	52.38%	90.48%
26	35	80.00%	53.57%	100.00%	53.57%
27	49	81.63%	72.50%	55.00%	70.00%
28	30	96.67%	55.17%	100.00%	75.86%
29	34	79.41%	51.85%	55.56%	88.89%
30	24	91.67%	18.18%	59.09%	81.82%
Mean	43	82.58%	51.45%	87.88%	82.48%

Table 5.1: Full results of all 30 video clips run with no heuristic and the two implemented heuristics, with the mean average calculated in the final row

a player is from the camera and the smaller they are, the harder it is for YOLO to give a detection with any certainty. The discovery of this meant that the majority of the cropped clips in the dataset used are of players that are at the near side of the pitch. The decision to do this was taken because as YOLO detection is not the main focus of this project but rather the heuristics for identification built on top of it, it was more advantageous to take clips that would yield the highest number of correct detections. If the project was to be extended to include more players or detections across the entire pitch, then problems may

arise from missed detections without the use of higher resolution video.

Another aspect to why in some frames players are not able to be correctly detected is motion blur from the movement of the players and the camera. Specifically in videos where players are moving at speed there is considerable motion blur which in addition to the low resolution can cause the players to become unrecognisable. Motion blur was also caused by the movement of the camera. When cropping the original videos to make the clips for testing, some of the clips are taken from when the camera is moving quickly away from the player and following the action. This means they had to be edited to keep the players in the centre of the frame. Despite this editing, if the camera moves very quickly away from the players it can blur them to the extent of YOLO no longer being able to detect them as being a person.

Yolo also suffers from dropped frames when a player is fully or almost fully covered by the other. This usually occurs in the frames at the middle of the video and the detection of both players can drop for a substantial amount of frames until they are far enough apart again to be recognised as separate people. It can also sometimes be the case that if one player is behind the other but only their legs are occluded, it is possible for YOLO to be able to detect the player just from the head and torso. This however can change the bounding box shape substantially, for example in this case decreasing the height by two, which can cause problems for the bounding box location heuristic.

5.3 Performance Of Identification Using Bounding Box Location

As shown in table 5.1 the average number of frames the heuristic was able to predict the correct identity over all video clips was 87.88%, which is a large improvement on the 51.45% scored when using no heuristic. Although this percentage is quite high it is the average of results that have a large range of 57.53. There are for example many video clips that the heuristic was able to keep the correct identities of both players in 100% of the frames. On the other hand there are also multiple clips where the heuristic scored substantially worse than the test using no heuristic.

From testing and the tests that scored highly it can be seen that the system performs well in certain scenarios. One of these is when players are moving across each other in opposite directions. As the bounding boxes start far apart there is very little chance of them getting switched as the difference between them is very high. If they are moving towards each other and then past each other there is little time when the bounding boxes are close enough together to get switched. Performance is also good if the bonding boxes stay roughly the same size and the occlusion is not full. As both the top left and bottom right of the bounding box is taken into account when calculating the difference between frames, if the occlusion is not full, then at least one of these two values will be different enough to differentiate between

Figure 5.1: This frame illustrates all three reasons why YOLO sometimes struggles to detect both players in the scene. The image quality is very poor, both players are running quickly causing motion blur and the occlusion on the red player by the player in black means they are much less likely to be detected correctly.

Figure 5.2: This figure is taken from video 5 in 5.1 where the bounding box heuristic only scored 42.47%. This low score is due to the fact that just before half way through the clip there are 6 undetected frame, after this point the identities have been switched and the system is unable to recover.

the two players and maintain the correct identities.

On the other hand the system can perform very poorly if players are moving past each other but in the same direction. In this scenario there will be more frames where the players are close together as they pass each other simply because it takes one longer to overtake the other when going in the same direction. This closeness in more frames increases the chances of the identities being switched. The size and shape of the bounding box changing can also negatively impact this heuristic. If one of the bounding boxes decreases in height, for example in the scenario described at the end of section 5.2, this increases the bounding box difference between frames and means that even if one corner of the box hasn't moved much at all, because the difference is calculated using both top left and bottom right, the difference between the the players bounding box in the last frame and the current frame many be higher than that of the other player in the previous frame, resulting in the players identities being switched.

Having dropped frames where the YOLO detection of one or both players has failed is one of the biggest reasons for poor results when using this heuristic. When the preceding frame does not have complete detections, the system goes back through previous frames to find the most recent with both players detected. However the more frames the system has to go back through, the more the players have moved and their bounding box locations changed during these missed frames. As mentioned in section 5.2 the majority of undetected frames are in the middle of the video as the players cross each other's path. This results in the player identities being switched as after these missing frames they are on the other side of each other and more closely match each other's bounding box.

5.4 Performance Of Identification Using Shirt Colour

When using the shirt colour of players as a heuristic for identifying them it can be seen that this also performs substantially better than not using a heuristic, with scores of 82.48% and 51.45% respectively. However, as with the bounding box location, this method also has a high range of percentage accuracy with some videos reaching 100% and others as low as 49.30%. This again is dependent on each specific video and its characteristics, who and where the players are, lighting, occlusions.

This heuristic performs well when the players are wearing very distinct or opposite colours, for example blue and red. This is because when the most common colour in each case is found, they will be far enough away from each other on the hsv scale to make comparison and differentiation much easier.

The system can however perform poorly when the colours are much more similar or the same. In many cases where players move across each other it is in the case of a tackle or chasing the ball and therefore they are on separate teams with different colours. It does however happen that players of the same colour cross each other's paths often too. This is one of the weaknesses of using this heuristic as the shirts being the same colour means that they are unable to be differentiated and the heuristic becomes as accurate as using none at all. Some of the video clips also include players crossing the path of the referee. In the clip the referee is wearing all black and this can also cause problems for detecting the colour. When filtering out the grass in the background the majority of the image becomes black and therefore the player or official is also included in this and the most common colour would be picked up as being their skin colour or boot colour. This can cause the identification to become mixed and switched as the colour detected is not consistent throughout the clip.

Lighting and orientation of the players can also cause this heuristic to perform badly. If a player moves from a lighter part of the pitch to a more shaded area or is shaded by the other player, this can cause the colour detected to change and not be picked up as being the same player, causing the identities to switch. The orientation of the player can affect which colour is chosen as the most common colour and mean it may not always be the colour of the players shirt. For example if the player is turned sideways with their arms by their side, it is likely that almost all of their shit will be blocked, this would result in another part of them, their skin colour, shorts and sock colour etc, being chosen as the most common. If this is closer to the other players shirt than it is to their own, this would again result in the two identities being switched.

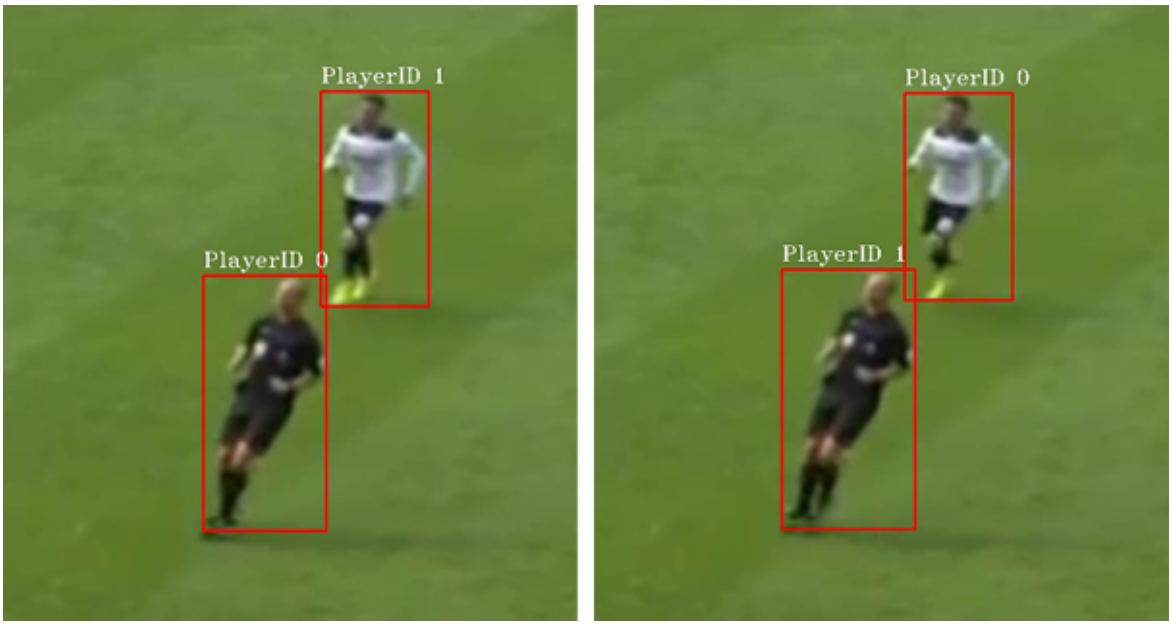

Figure 5.3: This figure is taken from video 10 in 5.1 in which the colour heuristic was able to score 49.30%. This score is likely to be low due to the referee being in black. This causes the system to recognise another colour as the shirt colour which may be leading it to getting the identities mixed up.

5.5 Review Of Aims And Objectives

The overall aim of this project was to create a real time object tracking system able track the movement of players in sports videos, maintaining their identities as they move across the scene and across each other. Although the system was set out to run in real time, this proved not to be possible with each frame taking around a second to be processed. This was due in part to using a larger YOLO model for player detection. Using a larger model increased the chances of detecting both players successfully and therefore increasing the number of frames that could be used in the heuristics. This was deemed more important to the project, to have greater accuracy of detection, than for the system to be able to process frames real time. Another reason for the system not running in real time is that the hardware available to run it on was sub-optimal. The system was run using only the CPU when YOLO does have the ability to run on a GPU with CUDA, this hardware was however not available for this project. If more advanced hardware was available it may be possible to have the system run in real time.

The objective to use a dataset of Premier League football videos that has been used in previous studies was achieved. The videos were however cropped to make them suitable for the specific requirements of this project, having only two players in the frame and being less than 3 seconds long.

Implementation of YOLO to facilitate player detection was also achieved. As discussed in previous chapters this did struggle to correctly identify both players in some frames however did fulfil its requirements in the most part. The objective to use OpenCV for video and image manipulation and to draw bounding boxes onto frames was fulfilled in its entirety. This tool was even more useful than originally thought with its ability to allow for the filtering of colours, making detecting the shirt colour easier.

Player tracking was also achieved. This was implemented by using the idea of track by detection. As a result of this it was more logical to integrate tracking into the methods for detection rather than being a stand alone module.

Two functions were set up to achieve the objective of keeping track of the players identities and re-identifying them as the same player after one had passed the other. Both re-identification by bounding box location and by shirt colour had their own strengths and weaknesses but both performed substantially better than re-identification without a heuristic, resulting in this objective being achieved.

5.6 Building On This Project

The project could be further developed by combining the two heuristics together. From testing it was shown that when one of the heuristics performed badly on a specific video clip, the other still achieved a reasonable score. This could be done by ways of a confidence system where whichever heuristic was most confident that the ID is correct, that is the ID that is chosen and used. This would mean that the system overall should score a higher percentage accuracy than without a heuristic and and when using the heuristics separately.

A third heuristic could be created. One that was considered was to use the players speed and direction. This would need to be built on top of the bounding box location as the difference in bounding box position from one frame to the next would be needed in the calculations for both the speed and direction, mainly in the first few frames with the first frame being unable to obtain this information at all. It was decided for this reason that it would not be implemented in this project as it was more likely to decrease the accuracy of identification rather than increase it, especially if it was thrown off within the first few frames. If it was possible to be sure the speed and direction could be calculated accurately in the first few frames this would be a good heuristic to use as when players are moving in opposite directions they could easily be distinguished by the direction their bounding box was moving. When moving in the same direction and one overtaking the other, speed could be used as for the overtake one player would need to be moving significantly faster than the other. This speed would be calculated by the number of pixels moved between the frames

and the direction based on the vector created by the difference in x and y of the bounding box between the two frames.

Another way this work could be taken further is to expand it to video clips that contain more players and then potentially the entire uncropped video containing many players. Some players may not be passing others and may be somewhere on their own but it would test how well the heuristics would work with additional detections. For example the bounding box location would still be assumed to work well and players on their own would be easily re-identified as their bounding box difference would be much smaller than that between them and other players. On the other hand it would be expected that the shirt colour heuristic would not perform as well. This is because introducing more players would mean more players having the same shirt colour and therefore not being able to differentiate between them. If the number of players was increased, this heuristic would need to be combined with another to make it viable, or only used on small sections of the video clip.

Chapter 6

Conclusions

Machine learning and computer vision is becoming mainstream in many aspects of our life, from detection in CCTV footage to self driving cars. An area that it is still growing in is the field of live sports and analysis. In team sports such as football and rugby there are many players on the field that are constantly moving past each other during tackle and other fast paced plays, making it difficult to accurately track individuals through a scene. This report has set out the project aim of creating a real time tracking and re-identification system that is able to follow players across the scene and maintain their identities as they move past and occlude each other. This was to be achieved using a data set of Premier League videos from the 2016/2017 season and implemented using python with the addition of tools such as YOLO, OpenCV and numpy.

Research was done into the variety of different tools, algorithms and techniques to help develop the system as well as finding out where tracking and re-identification have been used in sports in the past. This included ideas such as background subtraction to simplify the image, in addition to using colour histograms to differentiate between different objects of different colours. Estimation based methods were also found to be useful in tracking players. This includes using the position and velocity of players to analyse the objects previous movements and predict the position they should be in in the next frame. Research was also done into the different algorithms for detecting objects in a video, in order to determine which would be most suitable for this project. R-CNN was found to be the most accurate with the highest chance of correct detection, but was however a slow algorithm. Even with improvements in recent years to develop Faster R-CNN it was still not as fast as the YOLO algorithm which took a different approach to detection and processed the image in one pass rather than multiple. As the aim of the project was to create a real time tracking system, YOLO still seemed the best option.

Analysis was undertaken to determine what was needed for the project and how to move forward. A dataset was found that contained a vast amount of video footage of football matches from different countries and leagues. The dataset had been used in multiple other

research and papers on similar topics in the field of computer vision and player tracking. Using this dataset the decision was made to focus on videos from one season, specifically the most recent Premier League season available. To tailor the dataset to the purpose and scope of this project, small 1-3 second video clips were cropped out of the originals to be used for developing and testing the system. Analysis on how the objectives were going to be achieved was also done, outlining the tools that were to be used as well as discussing the steps needed in the algorithms for tracking and re-identification. Throughout the project it has been important to keep in mind any ethical, professional and legal issues that could arise. It was determined that this project has very little potential to have any issues as long as copyright laws are adhered to.

The implementation of the system included installing YOLO and tuning the confidence level to get the correct balance between the detections containing too many false negatives and too many false positives. Different classes were set up to contain the information about players and frames, in addition to the ones that ran the system using the different heuristics and ran the tests on them. The heuristic of bounding box location was implemented by finding the smallest difference in location between a player in the current frame and the players in the most recent available frame and giving them the corresponding ID. Using shirt colour as a heuristic was done by using KMeans clustering in order to find the most abundant colour in the region of the players bounding box. For each player this was compared to the colour of the players in the most recent available frame to find the closest match and give the new player that same ID. A function was implemented to allow for the creation of ground truth labels for each frame of each clip in the dataset. This was useful so that testing could be run automatically on each of the heuristics and then compared to the ground truths for analysis of performance.

The results obtained from these tests showed that using the heuristics greatly improved the accuracy of identification from the base of using no heuristic. The use of the bounding box location and shirt colour heuristics increased the accuracy from 51.45% to 87.88% and 82.48% respectively. It was found that both heuristics performed better and worse in different situations on different videos. The bounding box location performed better when players were moving in opposite directions and there were minimal missed frames by YOLO and the shirt colour heuristic performs best when the colours were bright and distinctly different without being too close to the colour of the pitch as to not get filtered out.

The majority of the aims and objectives of this project were fulfilled with the exception of being able to run in real time. This was compromised on to accommodate the use of a larger YOLO model, allowing for the best possible chance of correct detections in as many frames as possible. With more powerful hardware and running YOLO with a GPU could mean that the system may be able to run in real time, however, this hardware was not available for this project. Furthering and improving this project could be done by combining the two

heuristics as when one performs badly the other still performs well and vice versa. Increasing the number of players the system and heuristics can handle would also be a way to take the project forward and make it more useful to real world applications.

6.1 Future Work

Future work to this project would include further experimentation and testing on the current heuristics implemented to improve their accuracy further. For the shirt colour heuristic this could be achieved by adding checks to ensure that the detected colour is actually the colour of their shirt and not their skin, shorts or boot colour. Finding a solution or work around to the problems caused when a player or official is wearing black would also be a further improvement to the accuracy of this heuristic. Future improvements could be made to the bounding box location heuristic by experimenting with different distance measurements. The size of the bounding box could also be incorporated into the difference as well as potentially the speed and direction of the player. If there was more time allocated to the project these would be things that could be focused on to improve the system further.

Bibliography

[1] Ananth, S. Faster r-cnn for object detection. *Towards Data Science* (August 2019). Accessed: 8th November 2022.

[2] Cioppa, A., Giancola, S., Deliege, A., Kang, L., Zhou, X., Cheng, Z., Ghanem, B., and Droogenbroeck, M. V. *SoccerNet-Tracking: Multiple Object Tracking Dataset and Benchmark in Soccer Videos.* IEEE Computer Society, Los Alamitos, CA, USA, Jun 2022.

[3] .G, C., Jain, A., and Mohana, M. Real time object detection and tracking using deep learning and opencv. *Proceedings of the International Conference on Inventive Research in Computing Applications* (July 2018), 1305–1308.

[4] Gandhi, R. R-cnn, fast r-cnn, faster r-cnn, yolo — object detection algorithms. *Towards Data Science* (July 2018). Accessed: 8th November 2022.

[5] Giancola, S., Amine, M., Dghaily, T., and Ghanem, B. SoccerNet: A scalable dataset for action spotting in soccer videos. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (jun 2018), IEEE.

[6] Kim, J.-a., Sung, J.-Y., and Park, S.-h. Comparison of faster-rcnn, yolo, and ssd for real-time vehicle type recognition. In *2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia)* (2020), pp. 1–4.

[7] Li, M., Zhang, Z., Lei, L., Wang, X., and Guo, X. Agricultural greenhouses detection in high-resolution satellite images based on convolutional neural networks: Comparison of faster r-cnn, yolo v3 and ssd. *Sensors 20*, 17 (2020).

[8] Liu, J., Tong, X., Li, W., Wang, T., Zhang, Y., and Wang, H. Automatic player detection, labeling and tracking in broadcast soccer video. *Pattern Recognition Letters 30*, 2 (2009), 103–113. Video-based Object and Event Analysis.

[9] Liu, Q., Chen, D., Chu, Q., Yuan, L., Liu, B., Zhang, L., and Yu, N. Online multi-object tracking with unsupervised re-identification learning and occlusion estimation. *Neurocomputing 483* (2022), 333–347.

[10] Liu, T., and Liu, Y. Moving camera-based object tracking using adaptive ground plane estimation and constrained multiple kernels. *Journal of Advanced Transportation 2021* (July 2021).

[11] Naiyan Wang, Jianping Shi, D.-Y. Y., and Jia, J. Understanding and diagnosing visual tracking systems.

[12] Redmon, J., Divvala, S. K., Girshick, R. B., and Farhadi, A. You only look once: Unified, real-time object detection. *CoRR abs/1506.02640* (2015).

[13] Tan, L., Huangfu, T., Wu, L., and Chen, W. Comparison of yolo v3, faster r-cnn, and ssd for real-time pill identification.

[14] Ultralytics. Ultralytics yolov8, developed by ultralytics, is a cutting-edge, state-of-the-art (sota) model that builds upon the success of previous yolo versions and introduces new features and improvements to further boost performance and flexibility. yolov8 is designed to be fast, accurate, and easy to use, making it an excellent choice for a wide range of object detection, image segmentation and image classification tasks., 2022.

[15] Verma, Y. R-cnn vs fast r-cnn vs faster r-cnn – a comparative guide. *Developers Corner* (September 2021). Accessed: 8th November 2022.

[16] Xu, Y., and Wang, J. A unified neural network for object detection, multiple object tracking and vehicle re-identification. *CoRR 62* (2019).

[17] Zahra Soleimanitaleb, Mohammad Ali Keyvanrad, A. J. Object tracking methods:a review.

[18] Zhang, R., Wu, L., Yang, Y., Wu, W., Chen, Y., and Xu, M. Multi-camera multi-player tracking with deep player identification in sports video. *Pattern Recognition 102* (2020), 107260.