A project report on

FOOTBALL: DEEP-LEARNING BASED POSITIONAL ESTIMATION OF AGENTS, AUTOMATED REAL-TIME SPATIOTEMPORAL TRACKING, & PLANAR VISUALISATION

Submitted in partial fulfillment for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

by

MAKESH SRINIVASAN (19BCE1717)

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CHENNAI

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April, 2023

ABSTRACT

Football is one of the most played and watched sport in the world with over a billion fans spanning across the seven continents. With the advent of Artificial Intelligence, the way the sport is enjoyed is changing - the most prominent examples being the addition of Video Assistant Referee (VAR) system and Semi-Automated Offside Technology (SAOT), which have improved fairness in the sport, and automated decision making process in detecting offsides. There is still a great scope and potential for many implications of artificial intelligence in this sport.

In this research, I explored and built a novel way of visualising and analysing the games using deep learning and computer vision. Firstly, players, referees and ball are tracked in real-time with broadcast-feeds, and visualised on a 2-dimensional plane with a top-down view using state-of-the-art techniques. Subsequently, the tracked data is synthesised into a spatiotemporal dataset used to make positional estimates of players with respect to time, even when he/she is not visible within the camera's field of vision. This novelty is achieved using advanced recurrent neural networks such as LSTM and gradient boosting models such as XGBoost. In addition, major events such as game pauses, throw-ins, passes, and goals are automatically detected and recorded as logs for future data analysis.

While the scope of this research is to incorporate artificial intelligence to enhance audience's viewing experience, the spatiotemporal data and deep-learning prediction and tracking models can be used by football clubs, game developers and analysts to acquire deeper insights into gameplays, strategies, tactics and much more.

CONTENTS

Chapter 1
Introduction

This chapter serves to lay the groundwork in exploring the problem statement "Football: deep-learning based positional estimation of agents, Automated real-time Spatiotemporal tracking, & planar visualisation" within the domain of computer vision and deep learning.

1.1. INTRODUCTION

Football (also called soccer in some countries) is one of the most popular sport in the world with audience size ranging in billions across the planet. Most recently in the FIFA 2022, about 1.5 billion people watched the live broadcast on television. However, this was not the most interesting aspect of the tournament.

According to FIFA, there was a significant reliance on Artificial Intelligence (AI) tools such as the Video Assistant Referee (VAR) system and the Semi-Automated Offside Technology (SAOT) to improve fairness in the sport, and automate decision making process in detecting offsides and inspecting fouls. AI has the potential to revolutionise the way football is played and managed, providing coaches, players, and fans with a wealth of data-driven insights that were previously unavailable. By leveraging the tools of AI in various fronts such as visualisation, tracking and positional predictions, we can acquire data-driven insights for planning tactics, strategies and in general, improve the way the sport is played and visualised.

In this study, I developed a novel approach to visualising and analysing soccer games using cutting-edge deep learning and computer vision techniques. Initially, real-time tracking of players, referees, and the ball is conducted using broadcast feeds, and their movements are visualised in a two-dimensional plane from a top-down perspective, using state-of-the-art techniques. Subsequently, the tracked data is transformed into a spatiotemporal dataset that enables positional estimates of players with respect to time, even when they are not visible within the camera's field of vision. This innovative approach is made possible through the use of advanced recurrent neural networks, such as Long Short-term Memory networks (LSTM), and gradient boosting models, such as XGBoost. Additionally, the system automatically detects and records major events such as game pauses, throw-ins, passes, and goals as logs for future analysis.

1.2. RESEARCH OBJECTIVES

While the larger picture of the research is fundamentally divided into three stages outlined in the previous section (tracking and visualisation, spatiotemporal estimation, and synthetic data generation), the research objectives can further be divided into multiple sequential objectives which are as follows.

(i) **Detection of Agents:** The first step is to identify the players (including goalkeepers), football, and referees during football matches. I used advanced computer vision techniques to extract visual features from the broadcast feeds of the matches. Then, I employed state-of-the-art deep learning algorithms to identify the players, football, and referees from the video feeds, which were acquired from freely available online resources.

(ii) **Spatiotemporal Tracking:** After the identification of players, football, and referees, the next step was to track their movements spatiotemporally. I used advanced computer vision techniques to extract the spatial coordinates of each player and the ball in every frame of the video. The temporal component (frames) was then used to track their movements over time or frames. This spatiotemporal tracking enabled me to monitor the positions and movements of the players, referees and the ball throughout the game, which is crucial for subsequent stages of the research.

(iii) **Image Recognition (Jersey Number Detection):** To identify individual players playing the game, the number on the back of their jerseys are detected. I trained several deep learning models to recognise the numbers on the jerseys to be associated them with the players.

(iv) **Perspective Transform from Broadcast to 2D Plane Visualisation:** The broadcast feeds of football matches are often captured from different angles, which can make it difficult to visualise the movements of players and the ball. To overcome this, I used a perspective transform to convert the broadcast feeds into a two-dimensional plane visualisation. This allowed the movements of players and the ball to be visualised from a top-down view, making it easier to track their movements in cartesian coordinates system.

(v) **Generating Spatiotemporal Dataset and Estimating Agents' Positions:** Using the previous step, I generated a spatiotemporal dataset that was used to make positional estimates of players with respect to time, even when they were not visible within the camera's field of vision. This was achieved using advanced recurrent neural networks such as LSTM and gradient boosting models such as XGBoost. If a player moved out of frame, the model could still estimate their position accurately.

(vi) **Collection of Statistical Data or Events:** Finally, I collected statistical data or events such as passes, possessions, game pauses, and goals throughout the game. While the scope of this project is only to collect and synthesise the data in this part of the project, we can further use this information to analyse the game and gain insights into the performance of individual players and teams, which is beyond the scope of this research and would be explored in the next iteration of this research.

1.3. RESEARCH CHALLENGES

There are some research challenges associated with the design, development and implementation of the automated spatiotemporal tracking of agents and visualisation in football.

(i) **Data acquisition and processing:** One of the primary challenges is to collect high-quality and comprehensive data from multiple moving cameras in real-time, ensuring precise and accurate information on the positions of players, ball, and referees throughout the game. While the free available clips of football footages in high quality is freely available, it is crucial to find clips that are long enough for continued tracking and must not be interrupted by sudden camera switches or replaying of highlights or advertisements, as these can hinder the ongoing tracking of agents.

(ii) **Robustness and scalability:** The proposed system must be resilient to various conditions, such as lighting variations, occlusions, and camera lens distortion, to maintain consistent results. Additionally, scalability should be considered to accommodate different numbers of players, football pitch sizes, and diverse competitions. Furthermore, the footages can be parallel to the length or the width, and the system must be capable to performing perspective transformation appropriately.

(iii) **Integration of deep learning and computer vision techniques:** Developing a seamless integration of deep learning algorithms and computer vision methodologies to accurately track and predict the location and movements of players and game elements is a complex task that requires precise calculations and fine-tuning.

(iv) **Real-time performance and efficiency:** The ideal tracking system must not only be accurate but also operate at high efficiency, with minimal latency or delay to provide real-time analysis and insights during live football broadcasts. While this limitation is constrained by the hardware, it needs to be efficient enough to run smoothly on large processors used in broadcasting services.

(v) **Validation and evaluation:** A crucial challenge lies in establishing appropriate performance metrics and evaluation methods to assess the validity and reliability of the system in comparison to existing manual tracking and limited camera angle methods. While the models can be evaluated using standard metrics, the effectiveness of certain tools within computer vision needs to be evaluated through qualitative measures rather than quantitative ones as it involves human insights, comprehension, and holistic and comprehensive evaluation.

While the challenges outlined here are not the most exhaustive list of points, they outline some of the major ones that affect the overall progress of the development. Addressing these research challenges and exploring innovative solutions will contribute to a faster development of a more accurate and efficient system capable of unique ways of analysing and visualising the game.

1.4. PROBLEM STATEMENT

In recent years, football analytics have experienced significant advancements; however, there is still a deficiency in the precise and dependable real-time tracking and analysis of players, ball, and referees during football matches, particularly when it comes to their incorporation in tournament broadcasts. Presently, existing approaches largely depend on restricted camera angles and manual tracking techniques, which frequently result in incomplete and imprecise datasets. Consequently, there is a pressing need to create an automated and efficient system that can detect and track

players and other game components in real-time, produce a spatiotemporal data set, estimate player locations, and collect statistical data or events.

Moreover, there is a limited body of research dedicated to the probabilistic estimation of player positions in situations where cameras are also in motion. This study aims to bridge this research gap by investigating and developing an innovative method of visualising and analysing football matches using deep learning and computer vision techniques. The results of this study will contribute to the growing field of football analytics and provide a basis for more accurate and efficient methods of player and game element tracking in the future. Therefore, I define the problem statement as "Football: deep-learning based positional estimation of agents, Automated real-time Spatiotemporal tracking, & planar visualisation".

1.5. SCOPE OF THE PROJECT

The scope of this research project can be structured into the following components:

(i) **Data acquisition and preprocessing:** An extensive data collection involving various football matches will be conducted, ensuring representation of diverse teams, playing styles, and environmental conditions. The acquired data will be preprocessed to eliminate noise and highlight essential game components. While the need for a large dataset is not necessary in the tracking stages, it is very important during the model development stages and jersey number detection.

(ii) **System development and integration:** The research will focus on integrating cutting-edge deep learning and computer vision techniques to develop a novel player and game element tracking system. Existing models and methodologies will be analysed to identify potential improvements and adaptations to better suit the application in football analytics within the domain of computer vision and deep learning techniques.

(iii) **Real-time performance optimisation:** Development will prioritise the real-time functionality of the tracking system, with the objective of ensuring efficiency and speed to support live broadcasts and rapid analytics generation. This means sometimes, even a well performing models will be passed over for a lower performing one if the latency is very significant.

(iv) **Validation and evaluation:** A rigorous validation and evaluation phase will involve comparing the novel system's performance against established methods, utilising performance metrics and external validation approaches to ensure accuracy, reliability, and robustness. Although standard metrics can be used to evaluate the models, the effectiveness of some tools in computer vision requires qualitative measures rather than quantitative ones. This is because it involves human insights, comprehension, and a holistic and comprehensive evaluation.

(v) **Analysis of results and potential applications:** The research will include design and development of the proposed architectures and evaluation of its performance through a variety of quantitative metrics. The tracking and spatiotemporal estimation of player positions will be implemented in the most robust way. Finally, the dataset synthesised will be comprehensive but the exploratory data analysis (EDA) will not be within the scope of this research but would be included in the subsequent iterations of revisions of the research in the future.

(vi) **Future developments and recommendations:** Based on the findings and experiences acquired throughout the research process, suggestions for potential improvements, extensions, and future research opportunities will be provided, laying the groundwork for continuous advancements in the field of football player and game element tracking systems.

Chapter 2
Background

Chapter 2 expands the background studies of various developments in the related fields within the scope of this research. By laying a foundation in the progress and evolution of certain technologies and algorithms, we can better understand the reasoning behind my choices of methodology and proposed workflow in the subsequent chapters.

2.1. INTRODUCTION

In a broad sense, the research delves into the domains of multi-object tracking, image recognition, computer vision, time series forecasting and various forms of bagging and boosting algorithms. Therefore, I have done background survey on these areas by focussing on object tracking (individual and multi-object), perspective transformation, image recognition techniques, and several time series forecasting techniques to lay a foundation to understand the reasoning behind my methodology adopted (explored in detail in Chapter 3).

2.2. LITERATURE SURVEY

Football is one of the most popular sport with over 1.5 billion people around the world actively following the world cup and other international tournaments. With such a large global impact of football sporting events, it is vital that the fairness and governance of the sport be improved as well.

In the most recent FIFA tournament, the VAR and SAOT systems played crucial role in ensuring fairness and quick decision making throughout the game. Therefore, it is patent that the value and importance of the incorporation of AI in sports analytics, especially in football, is crucial in providing holistic and comprehensive analysis system, designing a strategy, analysing performance of teams and players, and last but not least, enhancing spectators's and remote audiences's viewing experience.

In the subsequent sections of this chapter, I have identified and explored the various areas of research such as object tracking, perspective transformation, image recognition, and time series forecasting, that were useful in designing my system in this research project.

objects between consecutive frames in a video sequence. Optical flow calculated the apparent motion of pixels and was able to track the movement of players and the ball across frames. Despite its success, it struggled with issues such as noise sensitivity and the aperture problem, which led to inaccurate motion estimates under certain conditions.

The development of feature-based detection methods, such as Scale-Invariant Feature Transform (SIFT) (Lowe, 2004) and Speeded-Up Robust Features (SURF) (Bay et al., 2006), marked a significant step forward in object detection. These methods used unique features extracted from images to identify and match objects across different frames, providing more reliable and robust object detection compared to earlier techniques.

The emergence of machine learning techniques, particularly deep learning, revolutionised object detection. The introduction of Convolutional Neural Networks (CNNs) (LeCun et al., 1998) and later improvements like R-CNN (Girshick et al., 2014) and Fast R-CNN (Girshick, 2015) enabled the creation of powerful models that could detect and localise multiple objects within a single frame with high accuracy. These advancements have been particularly useful in sports applications, including football, where the detection and tracking of players and the ball are crucial for game analysis.

YOLO (You Only Look Once) (Redmon et al., 2016) emerged as a groundbreaking object detection algorithm that offered real-time performance while maintaining high detection accuracy. YOLO framed object detection as a single regression problem and processed the entire image in one forward pass through the neural network, making it significantly faster than previous methods. This real-time performance has made YOLO an attractive choice for applications in football, where being able to detect and track objects in real-time is essential for live game analysis.

In summary, the evolution of object detection techniques, from early methods like image subtraction and optical flow to modern deep learning approaches such as YOLO, has resulted in significant improvements in both accuracy and efficiency which prove crucial for its applications of these techniques in football analysis.

2.2.2. OBJECT TRACKING

Object tracking has been a significant area of research within the computer vision community for several decades. Over the years, numerous techniques have been proposed to address the challenges of tracking objects in various scenarios, ranging from simple methods relying on colour histograms and optical flow to sophisticated algorithms leveraging machine learning and deep learning techniques. This literature review highlights the chronological development of object tracking approaches, culminating in the most recent advancements such as YOLO+DeepSORT, and BYTE, which have been used in the implementation of my system.

In the early years of object tracking research, simple techniques were employed to track objects in video sequences. In the 1980s, methods such as Mean Shift, Kalman Filter, and Particle Filter were proposed for tracking single or multiple objects. Mean Shift, introduced by Fukunaga and Hostetler (1975), is a gradient-based technique that iteratively shifts the position of an object towards regions of higher density, providing an efficient way to track objects in simple scenarios. The Kalman Filter, developed by Kalman (1960), is a recursive mathematical algorithm that estimates the state of a dynamic system based on a series of noisy measurements, offering a robust way to track objects in the presence of noise. The Particle Filter, proposed by Gordon et al. (1993), is a probabilistic approach that represents the possible positions and states of an object using a set of weighted particles, providing robust object tracking under noisy and uncertain conditions.

As the field of object tracking evolved, more sophisticated methods were developed to handle increasingly complex scenarios. In the 1990s and 2000s, researchers began to explore the use of machine learning techniques for object tracking. Avidan (2001) proposed a support vector machine (SVM)-based method for tracking objects in video sequences, demonstrating the potential benefits of using machine learning algorithms for object tracking tasks. In the early 2000s, researchers also began to investigate the use of feature-based methods for object tracking, such as the Scale-Invariant Feature Transform (SIFT) developed by Lowe (1999) and the Speeded-Up Robust Features (SURF) introduced by Bay et al. (2006). These methods extract unique features from images and use them to identify and match objects across different frames.

Another prominent technique is optical flow. Optical flow is a prominent technique in computer vision for tracking objects in videos and estimating motion between consecutive frames. The concept of optical flow was first introduced by Horn and

Schunck in their seminal work in 1981 (Horn & Schunck, 1981). Optical flow computes the apparent motion of pixels based on spatiotemporal intensity changes, providing valuable information about object movement. Since its inception, numerous variants and improvements have been proposed, including the Lucas-Kanade method (Lucas & Kanade, 1981) and Farnebäck's algorithm (Farnebäck, 2003).

During the late 2000s and early 2010s, the focus of object tracking research shifted towards the development of online learning algorithms that could adapt to changes in object appearance and motion patterns during tracking. These methods, such as the Online AdaBoost algorithm proposed by Grabner et al. (2006) and the Tracking-Learning-Detection (TLD) framework introduced by Kalal et al. (2011), were designed to update the tracking model on-the-fly, allowing for more robust tracking in challenging scenarios.

In recent years, the emergence of deep learning techniques has revolutionised the field of object tracking. In 2013, Wang et al. proposed the Deep Tracking method, which employed deep neural networks to learn hierarchical representations of objects for tracking. Since then, numerous deep learning-based tracking methods have been developed, such as Siamese networks introduced by Bertinetto et al. (2016) and the MDNet tracker proposed by Nam and Han (2016). These methods leverage the powerful feature extraction capabilities of deep learning models to achieve more accurate and robust object tracking, even under challenging conditions such as occlusions, rapid motion, and changes in object appearance.

One of the most significant recent advancements in object tracking is the development of the YOLO (You Only Look Once) object detection algorithm by Redmon et al. (2016). YOLO is a real-time object detection system that can detect multiple objects within a single frame with high accuracy, making it well-suited for tracking applications. The YOLO+DeepSORT strategy, which combines YOLO with the Deep Simple Online and Realtime Tracking (DeepSORT) algorithm by Wojke et al. (2017), offers a powerful tracking system that provides both high detection accuracy and robust tracking performance. DeepSORT extends the SORT (Simple Online and Realtime Tracking) algorithm by incorporating deep learning features to improve tracking accuracy.

Another recent advancement in object tracking is the BYTE (Back-propagated Temporal YOLO Ensemble) tracking method, which improves upon the

YOLO+DeepSORT strategy by leveraging an ensemble of YOLO detectors trained on consecutive video frames. In BYTE tracking, multiple YOLO detectors are trained on the temporal context of video sequences, allowing the model to learn the appearance variations and motion patterns of objects across frames. During the inference process, BYTE tracking combines the object detections from the ensemble of YOLO detectors using a back-propagation-based fusion strategy, resulting in improved detection accuracy and reduced false positives.

In conclusion, the field of object tracking has seen significant advancements over the years, with approaches evolving from simple techniques such as Mean Shift and Kalman Filter to sophisticated deep learning-based algorithms like YOLO, YOLO+DeepSORT, and BYTE tracking. The development of these advanced tracking methods has greatly improved the accuracy and robustness of object tracking systems, enabling more effective analysis of motion in various applications, including sports analytics.

2.2.3 PERSPECTIVE TRANSFORMATION

The concept of perspective transformation has its roots in projective geometry, dating back to the work of Renaissance artists and mathematicians, such as Brunelleschi and Alberti (Kraus, 1994). However, the formalisation of perspective transformation and its application to computer vision and image processing began to take shape in the mid-20th century.

In the 1960s, Roberts (1963) proposed a method to recover three-dimensional information from two-dimensional images, laying the foundation for the development of perspective transformation techniques in computer vision. Following this, the Direct Linear Transformation (DLT) method was introduced by Abdel-Aziz and Karara (1971), providing a straightforward approach to compute the transformation matrix between image coordinates and object coordinates given a sufficient number of correspondences.

Later, Tsai (1987) proposed a widely-used camera calibration technique, which further advanced the field of perspective transformation. This method enabled the estimation of intrinsic and extrinsic camera parameters, which are essential for transforming and rectifying images captured by a camera.

In the 1990s, researchers began exploring robust methods for estimating the transformation matrix in the presence of noise and outliers. Hartley (1997) introduced the method of normalised eight-point algorithm for computing the fundamental matrix in epipolar geometry, which is closely related to perspective transformation. This algorithm improved the accuracy and stability of the transformation matrix estimation by normalising image coordinates before applying the DLT method.

Around the same time, Lowe (1991) introduced a technique called the "image alignment" method, which used local image features and their geometric constraints to compute an accurate perspective transformation. This method laid the groundwork for the development of feature-based transformation techniques, such as Scale-Invariant Feature Transform (SIFT) by Lowe (2004), which has since become a popular approach for perspective transformation and image matching.

In summary, perspective transformation has evolved significantly over the past few decades, with advancements in computer vision and image processing techniques contributing to its development. While the roots of the strategies remain the same, the approaches and methodologies evolved over time. In my research, perspective transformation would play a crucial role in displaying of live game broadcasts in 2-dimensional cartesian coordinate system using the library of OpenCV in Python.

2.2.4 IMAGE RECOGNITION

Image recognition has been a critical area of research within the field of image processing for several decades. The development of advanced image recognition techniques has led to significant improvements in various applications, such as object detection, segmentation, and classification.

One of the earliest image recognition techniques was the perceptron, a linear classifier proposed by Rosenblatt (1958)[1]. The perceptron marked the beginning of research on artificial neural networks, which were later extended to multi-layer perceptrons (MLPs) and back-propagation learning algorithms by Rumelhart, Hinton, and Williams (1986)[2].

The 1990s saw the emergence of Support Vector Machines (SVMs), a powerful discriminative classifier introduced by Cortes and Vapnik (1995)[3]. SVMs demonstrated remarkable performance in image recognition tasks, particularly when combined with feature extraction methods such as Scale-Invariant Feature Transform

(SIFT) by Lowe (2004)[4] and Speeded-Up Robust Features (SURF) by Bay, Ess, Tuytelaars, and Van Gool (2008)[5].

The development of Convolutional Neural Networks (CNNs) by LeCun, Boser, Denker, Henderson, Howard, Hubbard, and Jackel (1989)[6] marked a significant milestone in the field of image recognition. CNNs proved to be highly effective in image classification tasks, as demonstrated by Krizhevsky, Sutskever, and Hinton (2012)[7], who proposed the AlexNet architecture and achieved state-of-the-art performance in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC).

Subsequent advancements in CNN architectures led to the introduction of VGGNet by Simonyan and Zisserman (2014)[8], which achieved top performance in the ILSVRC 2014 competition. VGGNet introduced the concept of using smaller convolutional filters to reduce the number of parameters, while still maintaining a deep network structure.

Continuing the trend of deeper networks, He, Zhang, Ren, and Sun (2016)[9] proposed the ResNet architecture, which introduced residual connections to alleviate the vanishing gradient problem in deep networks. ResNet demonstrated exceptional performance in image recognition tasks and won the ILSVRC 2015 competition.

In the context of image segmentation, Ronneberger, Fischer, and Brox (2015)[10] introduced the U-Net architecture, which combined an encoding and decoding structure to produce high-resolution segmentation maps. Later, Zhou, Canu, and Ruan (2019)[11] proposed the UNet++ architecture, which improved upon the original U-Net by introducing nested skip connections and deep supervision, resulting in more accurate image segmentation.

HarDNet, a highly efficient deep learning architecture, was proposed by Chao, Chiang, Natarajan, and Wang (2019)[12]. HarDNet introduced a densely connected structure with channel-wise connections, reducing the computational complexity while maintaining high performance in image recognition tasks.

Finally, the EfficientNet architecture, developed by Tan and Le (2019)[13], represents a significant advancement in image recognition. EfficientNet employs compound scaling to optimise the model's depth, width, and resolution, resulting in improved performance with fewer parameters and reduced computational overhead.

EfficientNet has demonstrated state-of-the-art performance in various image recognition tasks, including object detection, segmentation, and classification.

In conclusion, the field of image recognition has evolved significantly over the years, with major advancements in architectures such as ResNet, UNet++, HarDNet, and EfficientNet. These developments have enabled remarkable improvements in image recognition performance, leading to more effective and efficient solutions for various image processing applications.

2.2.5. TIME SERIES FORECASTING

Time series forecasting has been a significant area of research for decades, with various techniques and methodologies being developed and refined over time. This literature survey aims to provide a chronological overview of the key advancements in time series forecasting, focusing on methods such as ARIMA, moving averages, exponential averaging, and deep learning-based approaches like RNNs, LSTMs, and Transformers.

In 1950, Brown introduced the concept of exponential smoothing, a technique that assigns exponentially decreasing weights to past observations, creating a smoothed time series for forecasting purposes (Brown, 1950). Holt later expanded upon Brown's work by incorporating a linear trend component, giving rise to the Holt-Winters method for time series forecasting (Holt, 1957; Winters, 1960).

Box and Jenkins further contributed to the field of time series forecasting by developing the Autoregressive Integrated Moving Average (ARIMA) model in the 1970s (Box & Jenkins, 1970). ARIMA combines autoregressive (AR), differencing (I), and moving average (MA) components, providing a robust and flexible framework for modelling and forecasting time series data. ARIMA remains a widely-used method for time series forecasting to this day.

In the 1980s and 1990s, researchers began exploring the use of artificial neural networks for time series forecasting (Tang et al., 1991; Zhang et al., 1998). These early works demonstrated the potential of neural networks for capturing complex relationships and nonlinearity within time series data, setting the stage for future innovations in the field.

The introduction of Recurrent Neural Networks (RNNs) by Rumelhart et al. (1986) marked a significant advancement in the field of time series forecasting. RNNs are designed to handle sequential data and can capture long-range dependencies in time series, making them well-suited for forecasting tasks. However, RNNs were initially hampered by the vanishing gradient problem, which limited their ability to learn long-range dependencies (Bengio et al., 1994).

Hochreiter and Schmidhuber addressed this issue by developing Long Short-Term Memory (LSTM) networks in 1997 (Hochreiter & Schmidhuber, 1997). LSTMs are a type of RNN architecture that can effectively model long-range dependencies in time series data, thanks to their unique gating mechanisms. Since their introduction, LSTMs have become a popular choice for time series forecasting and have been successfully applied in various domains.

Another variant of RNNs is the Gated Recurrent Unit (GRU), introduced by Cho et al. (2014). GRUs are similar to LSTMs in terms of their ability to model long-range dependencies, but they have a simpler architecture with fewer parameters, making them computationally more efficient in some cases.

Recently, attention-based mechanisms have gained popularity in the field of time series forecasting. Vaswani et al. (2017) introduced the Transformer architecture, which employs self-attention mechanisms to capture global dependencies within a sequence. Transformers have demonstrated strong performance on various time series forecasting tasks, including electricity consumption and traffic flow prediction (Li et al., 2019).

In summary, time series forecasting has evolved significantly over the past decades, with advancements in statistical methods and the emergence of deep learning-based approaches. From the early work on exponential smoothing and ARIMA models to more recent developments in RNNs, LSTMs, and Transformers, time series forecasting continues to be a critical area of research with broad applications. Specifically to my research, traditional strategies and LSTM will be used as Transformers are more applicable in scenarios where a longer term of memory is to be stored. As players tend to change motion in instantaneous ways, and their velocities may be eclectic, it is not important to store data on multiple historical sequences of time steps, only a few sequences would suffice.

Chapter 3
Methodology

In this chapter, I have outlined the modules within the research, proposed experiments to plan and implement each modules, and expanded on the relevant algorithms and architectures.

3.1 INTRODUCTION

In this study, I propose a comprehensive methodology to effectively visualise and analyse football games using deep learning and computer vision techniques. The methodology involves the identification of key entities, object tracking with unique IDs, jersey number detection, perspective transformation for 2D visualisation, collection of statistical data, and player position estimation. This comprehensive approach will enable us to comprehend the complex underlying principles of a game of football, ultimately providing valuable information for coaches, players, and sports enthusiasts alike.

3.2 PROPOSED MODULES

While there is a general flow in the sequence of modules, there is no absolute rigidity that one must be complete in order to pipeline into the next module. As some of the modules are relatively independent of one another (except for modules (i), (ii), (iii), and (v)), there is relatively low dependency between the other modules which enables the modules to be developed simultaneously as well.

The methodology consists of the following steps:

(i) **Dataset:** The primary dataset will consist of videos from football game broadcasts from major tournaments like Bundesliga or LaLiga. These videos will serve as the basis for the analysis, providing the raw data necessary for the detection and tracking of players, goalkeepers, referees, and the ball. Furthermore, another dataset with images of jersey number on the players' back during a game will be used to train a model to recognise player numbers. It is important to note that the dataset does not necessarily need to be of football players; it can consist of any players with similar numbering formats in any sports such as rugby, American football, basketball, cricket, etc.

(ii) **Detection of non-noisy humans and the ball:** In this step, advanced image recognition algorithms will be deployed to accurately identify the ball, players and referees (non-noisy humans) within the game broadcast footages. This process will allow me to distinguish between the different agents present on the field and eliminate the noisy parts (audience and miscellaneous elements), including the background entities to establish a clean detection for further analysis.

(iii) **Tracking agents/objects:** Once the players, referees, and the ball have been identified, we will track them throughout the game using unique IDs known as "tracker_id". This will allow me to monitor their movements and interactions in real-time, providing valuable insights into the gameplay.

(iv) **Jersey number detection:** To recognise individual players, we will implement image recognition algorithms to detect the jersey numbers displayed on their players' backs. In order to achieve this step, I used a synthetic dataset by taking images from various games of football games, National Football League games, and basketball games.

(v) **Landmark detection:** In this step, the landmarks such as the centre of the circle, corners, penalty box, etc. will be identified and marked to perform perspective transformation. These will serve as absolute coordinates using which the relative coordinates of the players and the ball can be calculated.

(vi) **2D visualisation:** In this step, I will perform a perspective transform to convert the broadcast footage into a two-dimensional cartesian plane visualisation using 4 non-collinear points. This top-down view will provide a clear and comprehensive representation of the game on a virtual field, facilitating more effective analysis and interpretation of the data that would be used in the subsequent steps.

(vii) **Collect statistical data:** Various statistical data such as possession, passes, goals, ball-outs, and game stoppages will be stored as a synthetic data for further analysis (and as mentioned before, EDA is not part of this iteration of research). This information will help us gain insights into the strategies employed by the teams and players, and the overall progression of the game.

This module will be done during the synthetic data generation phase in conjunction with the previous module to create a new CSV file with time series data with virtual positional coordinates of each players within the field.

(viii) **Player position estimation:** To account for players who move out of the camera's field of view, we will utilise the spatiotemporal data generated in previous "t" time steps to predict their next positions. This estimation process will be used to make predictions even when players are not visible on the screen.

3.3 PROPOSED WORKFLOW

The sequence of steps outlined above is presented as a diagram in the Figure 1. Once the data is processed, it is pipelined into the next module - detection and tracking of agents. Once the non-noisy agents are acquired, subsequent modules can use this

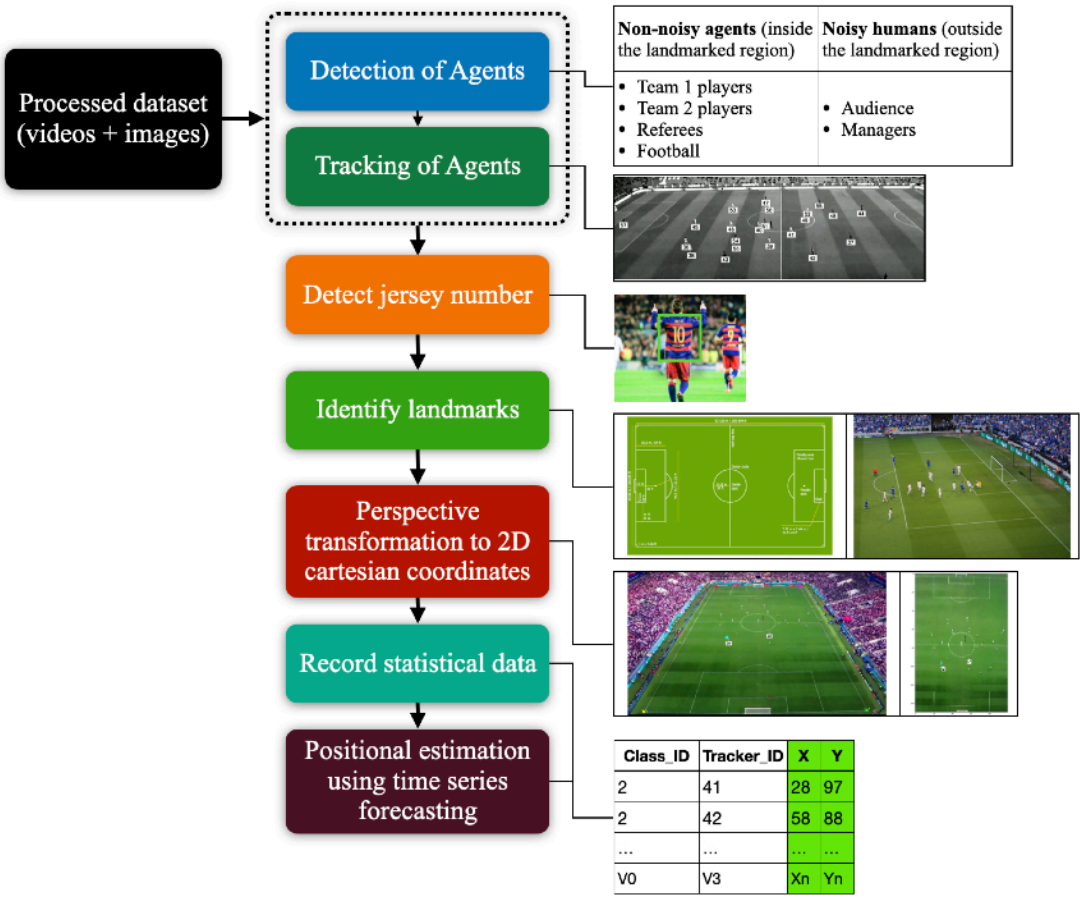


Figure 3.1: Proposed workflow

phase's output. In the identify landmarks stage, the coordinates of reference points are detected and then perspective transform is applied to get cartesian coordinates on a virtual 2D plane. Simultaneously, the jersey number can be detected as well. The statistical data and the positional estimation of agents using time series forecasting can also be done following these stages.

3.4 PROPOSED ARCHITECTURE AND ALGORITHMS

For each of the modules outlined in the section "proposed workflow", the proposed algorithms and architectures are outlined in this section as follows.

3.4.1. DETECTION OF AGENTS

There are several ways to detect players and the ball in a video frame by leveraging various computer vision and deep learning techniques. Some of these methods that could be implemented are as follows.

(i) **Background Image subtraction:** This technique involves comparing each frame of the video with a reference background image to identify moving objects. By subtracting the background, the remaining foreground elements, such as players and the ball, can be isolated and detected. While this method is simple to implement, the robustness and the scalability of this model will be highly questionable as there are many noisy-humans, and lack of clear and distinct agents within a frame.

(ii) **Object detection algorithms:** Pre-trained object detection algorithms, such as YOLO (You Only Look Once) or SSD (Single Shot MultiBox Detector), can be fine-tuned to detect specific objects like players and the ball. These deep learning models are capable of identifying and localising multiple objects within a single frame, providing accurate and efficient detection. This strategy would be ideal in my use case as it helps not only detect players quickly but also localise them within the environment.

(iii) **Machine learning classifiers:** Supervised machine learning algorithms, such as Support Vector Machines (SVM) or K-Nearest Neighbours, can be trained on labeled datasets to classify different objects within a frame, including players and the ball. These classifiers can be combined with other computer vision techniques, such as feature extraction or object segmentation, to improve

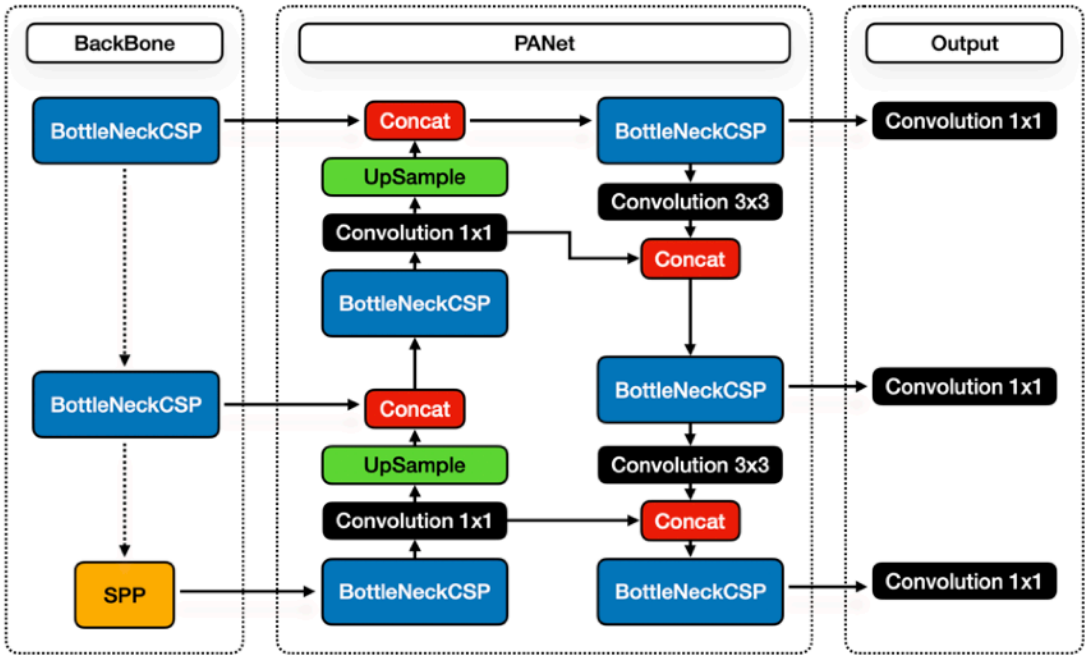


Figure 3.2: YOLOv5 architecture

detection accuracy; however, they are not very useful when it comes to localisation of objects within an image.

3.4.2. TRACKING OF AGENTS

Once the players and the ball have been detected in a video frame, several methods can be employed to track these objects throughout the video sequence. Some of these tracking techniques include:

(i) **Kalman Filter:** This is a recursive mathematical algorithm that estimates the state of a dynamic system based on a series of noisy measurements. The Kalman Filter can be used to predict the future position of the detected objects and correct the estimations based on new measurements, providing a smooth and accurate tracking of the players and the ball. While this strategy is immensely useful, in the subsequent points, I have outlined a better algorithm that combines Kalman filter with YOLO and DeepSORT algorithm to do much more powerful processing of data.

(ii) **Optical Flow:** Optical Flow methods estimate the motion of objects between consecutive frames in a video sequence. By calculating the apparent motion of pixels, we can track the movement of players and the ball across frames. This

technique can be combined with other tracking methods to improve accuracy and robustness. Moreover, this would allow me to keep track of fixed points (absolute points or landmarks) that can be used in perspective transformation module.

(iii) **Multiple Object Tracking (MOT):** MOT algorithms are designed to track multiple objects simultaneously in a video sequence. Two of the proposed algorithms to perform multiple object tracking are:

(i) *YOLO + DeepSORT:*

In the YOLO+DeepSORT strategy, Yolo is first employed to detect objects (e.g., players and the ball) in each frame of the video. The detected objects are then passed to the DeepSORT algorithm, which tracks the objects across frames by associating detections and maintaining unique object identities. DeepSORT uses a combination of the Kalman Filter for motion prediction, the Hungarian algorithm for data association, and a deep learning-based appearance model for matching objects across frames. Hence, using YOLO and DeepSORT in combination results in a powerful tracking model that offers high detection accuracy and robust tracking performance.

(ii) *BYTE Tracking:*

BYTE (Back-propagated Temporal YOLO Ensemble) tracking is a more recent tracking method that improves upon the YOLO+DeepSORT strategy by leveraging an ensemble of YOLO detectors trained on consecutive video frames. In BYTE tracking, multiple YOLO detectors are trained on the temporal context of video sequences, enabling the model to learn the appearance variations and motion patterns of objects across frames.

The BYTE tracking approach offers several advantages over the traditional YOLO+DeepSORT strategy. By incorporating temporal information from video sequences during the training process, BYTE tracking can better handle appearance changes and motion variations, resulting in more accurate and robust object tracking of players, ball and referees. Additionally, the ensemble-based approach helps to mitigate the limitations of single-frame YOLO detectors, which further improves detection and tracking performance.

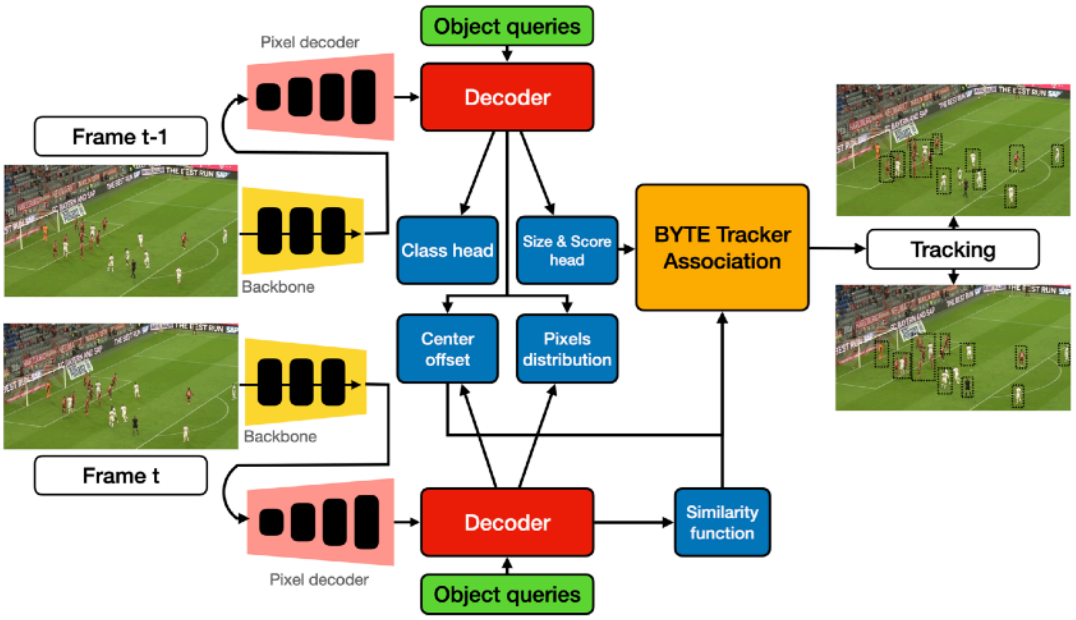


Figure 3.3: BYTE Tracking architecture

(iv) **Deep learning-based tracking:** Techniques such as Recurrent Neural Networks (RNN) can be employed to learn the temporal dependencies and appearance variations of objects across video frames. These deep learning models can provide robust and accurate tracking of players and the ball, even under challenging conditions such as occlusions or rapid motion. While this strategy might yield best performance in terms of accuracy and robustness, it is also crucial to consider the latency in detection as this system is meant to be a real-time model (as mentioned in Chapter 1: Research Objectives).

3.4.3. DETECTION OF JERSEY NUMBER

There are several image recognition techniques that can be employed for jersey number detection in football videos. Some of these methods include:

(i) **Optical Character Recognition (OCR):** OCR algorithms, such as Tesseract, are helpful in detecting and recognising text and numbers within jersey images. Pre-processing techniques, such as image binarisation or other morphological operations can help enhance the readability of the numbers before feeding the image to the OCR engine. However, in my use case this technique may be too slow and carry a heavy latency which would slow down its processing thereby causing delay in real-time tracking of players.

(ii) **Feature-based detection:** Feature-based methods, such as Scale-Invariant Feature Transform (SIFT) or Speeded-Up Robust Features (SURF) or Oriented FAST and rotated BRIEF (ORB), extract unique features from images and use them to identify and match objects across different frames or images. These features can be used to detect and recognise jersey numbers, especially when combined with other strategies mentioned here such as machine learning classifiers or template matching techniques. However, the clarity of images must be quite clear and legible for this algorithm to do an effective job of matching. Since the images of players are going to be a few pixels in width and length, this technique may not be the most suitable approach.

(iii) **Deep learning-based methods:** Convolutional Neural Networks (CNNs) and other deep learning architectures can be trained on labeled datasets to identify and recognise jersey numbers. These models can learn to capture complex patterns and variations in the appearance of numbers, providing accurate and robust detection. While this strategy is simpler to implement, more robust architectures such as EfficientNet.

(i) *EfficientNet:*

EfficientNet is a deep learning model primarily used for image classification tasks involving large scale images and data processing, as it efficiently balances model accuracy and computational complexity. It employs compound scaling to optimise the model's depth, width, and resolution, resulting in improved performance with fewer parameters and reduced computational overhead. EfficientNet can be fine-tuned to perform jersey number recognition, by training the architecture on a labeled dataset of jersey numbers, as it can learn to identify and recognise the digits accurately even with overlaying and occlusions, or distorted perspectives, which are common in sporting images and videos.

(iv) **Machine learning classifiers:** Supervised machine learning algorithms such as Support Vector Machines (SVM) or Random Forest can be trained on labeled datasets to perform classification tasks on different digits within an image. These classifiers can be combined with other computer vision techniques, such as feature extraction or object segmentation using YOLO networks for instance, to improve jersey number detection accuracy as well as perform localisation.

3.4.4. LANDMARK IDENTIFICATION

In order to accurately estimate the boundary lines of the football field, I decided to use canny edge detection and Hough Line transformation as they provide the simplest, most robust, and quickest way to extract the football field from the frames to be fast enough to implement on a real-time system.

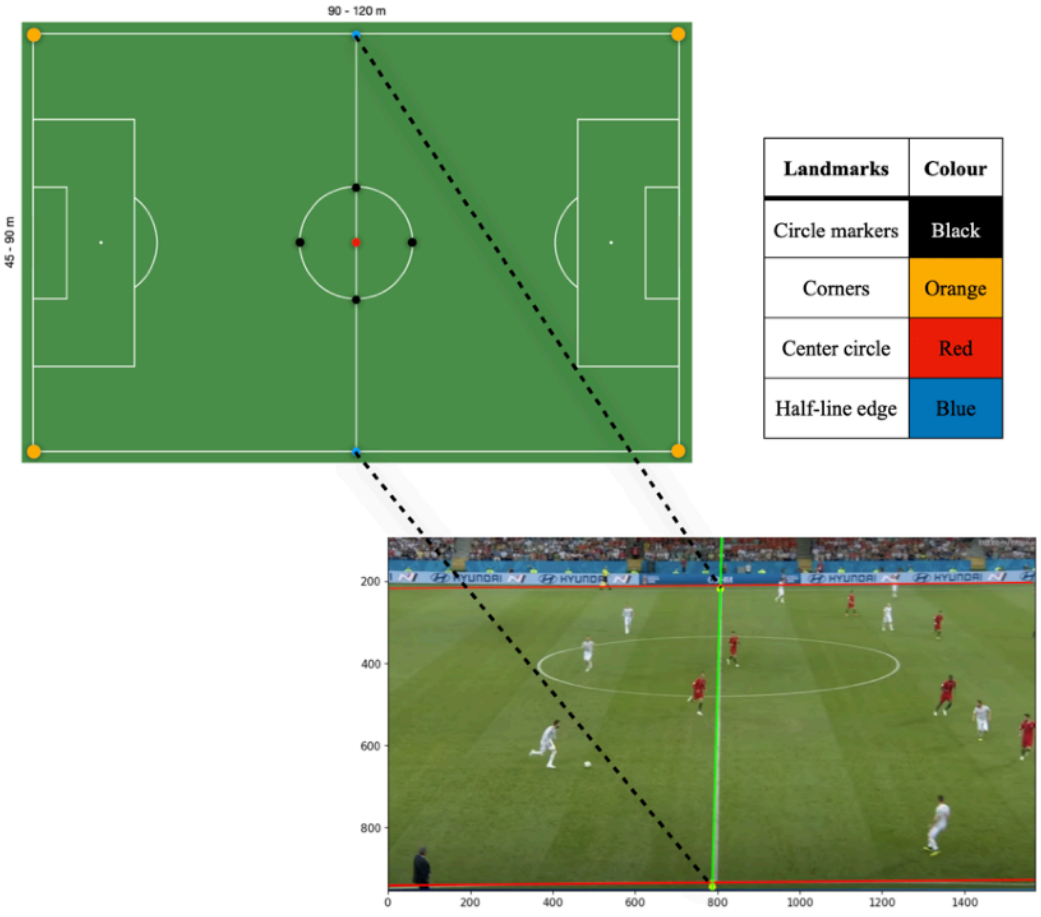


Figure 3.4: Landmark matching

Canny edge detection is an algorithm designed to identify edges within an image by detecting areas with rapid changes in intensity. It works by first applying a Gaussian filter to smooth the image, then calculating the intensity gradients and applying non-maximum suppression to thin the edges. Finally, the algorithm uses double thresholding and edge tracking to generate a binary image containing the detected edges. When applied to an image of a football field, the Canny edge detection

algorithm can effectively highlight the field lines and markings by identifying the sharp transitions between the grass and the painted lines.

Once the edges have been detected, the Hough lines transformation can be used to identify the straight lines that correspond to the football field lines. The Hough transformation is a feature extraction technique that works by mapping each edge point in the image to a set of lines in a parameter space, known as the Hough space. The intersections of these lines in the Hough space represent the parameters of the straight lines in the original image. By accumulating votes for each intersection point and applying a threshold, the Hough lines transformation can identify the most prominent straight lines within the image, corresponding to the football field lines.

In combination, the Canny edge detection and Hough lines transformation provide a powerful tool for determining the football field lines within an frame. By accurately identifying and locating these lines, the absolute coordinates of the landmarks can be determined and the perspective transform operation can be performed to calculate the relative positions of the players.

Additionally, I also implemented Optical Flow, specifically, I used Lucas Kanade algorithm to track some of the points that were deemed interesting by the algorithm - points containing high contrasts and sudden changes in colours, such as corners and dots. While this was capable of tracking all landmarks continually as the camera pans away, the chances of losing each of the identified points were very high as the frames move too fast. Also, the speed of the detection and tracking was far too high to be implemented in a real-time model. Hence, the above strategy using Canny and Hough Lines was used.

Furthermore, ORB, SIFT and SURF models were built to use a template image to map a subject image from the video feed. However, due to high noise within the subject images or frames, the feature matching algorithms were not as robust and clean. Among the three, ORB performed well in most cases, closely followed by SURF and then SIFT. However, it was still not doing as good as the first method.

3.4.5. PERSPECTIVE TRANSFORMATION TO 2D CARTESIAN PLANE

Perspective transformation is a crucial step in converting the 3D representation of a football field captured by a camera into a 2D top-down view, which allows for more effective visualisation and analysis of the game. This transformation process involves

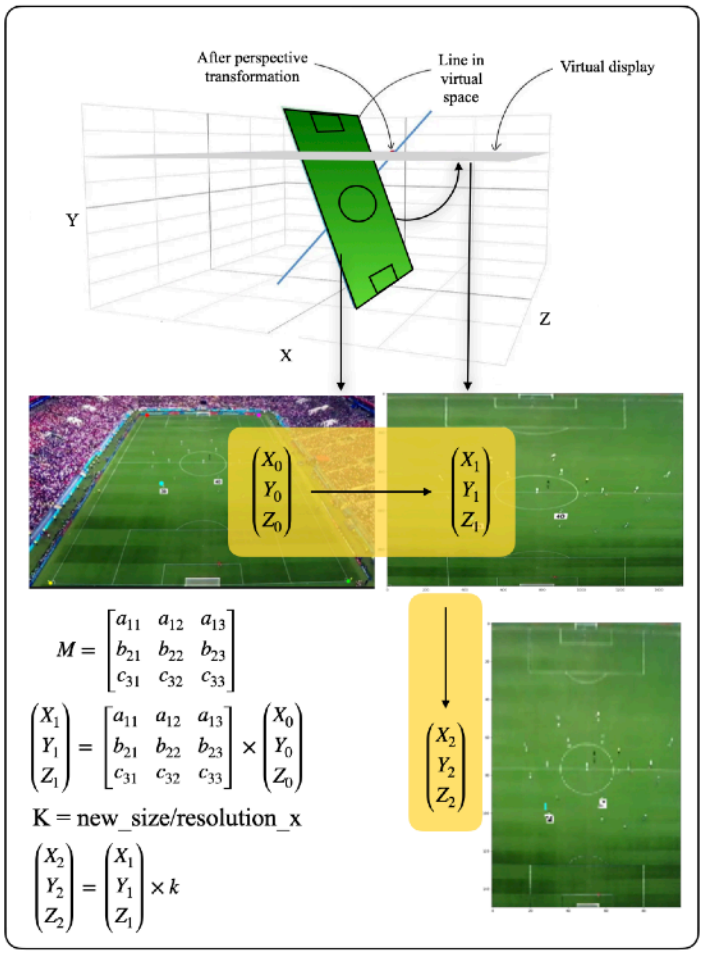


Figure 3.5: Perspective transformation

mapping the original image coordinates to the desired 2D plane using four reference points, which correspond to the corners of the playing field.

The first step in perspective transformation is to identify the four coordinates in the original image that represent the corners of the football field. These coordinates can be determined manually by selecting the corner points, or automatically using computer vision techniques such as edge detection and line extraction. Once the four coordinates are identified, a relationship between the original image points and the corresponding points in the desired 2D plane can be established.

To perform the actual transformation, a homography matrix is computed, which defines the geometric transformation between the original image points and the corresponding 2D points. The homography matrix can be calculated using various methods, such as the Direct Linear Transformation (DLT) algorithm. With the

homography matrix in hand, we can apply the transformation to the entire image, effectively mapping the 3D view of the football field to a 2D top-down representation.

3.4.6. POSITIONAL ESTIMATION USING TIME SERIES FORECASTING

Time series forecasting is a powerful technique used to predict future values based on historical data, which is the requirement in the context of predicting a player's position in a football game using a 2D cartesian coordinate dataset from multiple frames, several methods can be employed to perform this forecasting. Some of these approaches include:

(i) **Autoregressive Integrated Moving Average (ARIMA):** ARIMA is a statistical model that combines autoregressive (AR), differencing (I), and moving average (MA) techniques to perform time series forecasting. By fitting an ARIMA model to the x and y coordinate datasets, we can predict the future positions of a player based on their past positions.

(ii) **Exponential Smoothing:** Exponential Smoothing is also a statistical forecasting method that uses exponential smoothing techniques to capture various components of a time series, such as trend and seasonality. By applying Exponential Smoothing to the player's x and y coordinate datasets, we can generate accurate predictions of their future positions. However, it is generally more useful in situations where there is a sense of periodicity which players' motion will not likely have.

(iii) **LSTM Networks:** LSTM, as outlined several times int he previous sections, is a type of RNN specifically designed to handle time series data and capture long-range dependencies. By training an LSTM network on the sequence of x and y coordinates, we can model the complex dynamics of a player's movement and generate predictions for their next positions.

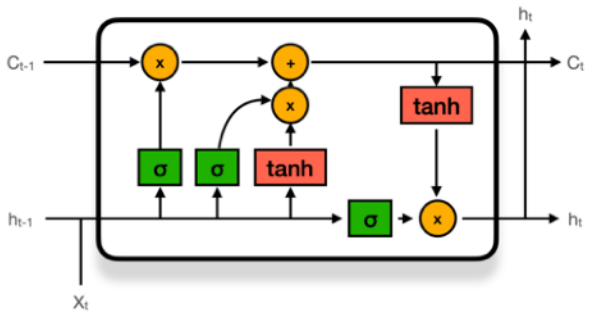


Figure 3.6: LSTM cell

(iv) **XGBoost regression:** XGBoost regression is an ensemble learning algorithm that utilises decision trees to make predictions. XGBoost is particularly useful for time series forecasting because it can handle missing data and is capable of capturing complex patterns in the data. This, like the others mentioned above, can be used in the same way to make positional predictions.

(v) **VAR (Vector Autoregression):** VAR is used in time series forecasting and it involves modelling the relationship between multiple variables over time. In time series forecasting, VAR can be used to predict future values of multiple variables based on their historical values.

3.5 EXPERIMENTAL SET-UP & ARCHITECTURE FORMULATION

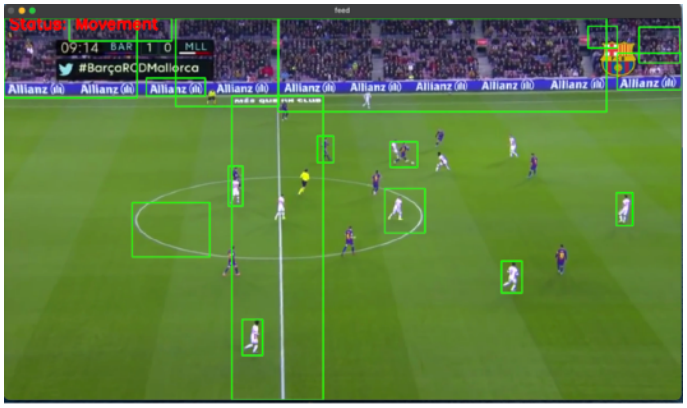
This section has been removed as the research is ongoing. For more information, kindly get in touch using the contact information.

Chapter 4
Results

In this chapter, the results of all the experiments are presented in a module-wise order. The experiments conducted may not be in the order in which they are displayed but the general sequence of modules is outlined in the same way as described in Chapter 3: Methodology. Moreover, it is to be noted that not all tests and experiments conducted are described in this chapter, only the most relevant ones are shown.

4.1. DETECTION OF AGENTS

4.1.1 BACKGROUND IMAGE SUBTRACTION (EXPERIMENT 3.5.1.1)

Background subtraction
 Figure 4.1: Background subtraction
Figure 4.1 shows the background image subtraction method employed to detect agents (non-noisy humans) within the image.

4.1.2 YOLOV5: OBJECT DETECTION ALGORITHM (EXPERIMENT 3.5.1.2)

Experiment 3.5.1.2. (A) COCO standard weights

COCO standard weights

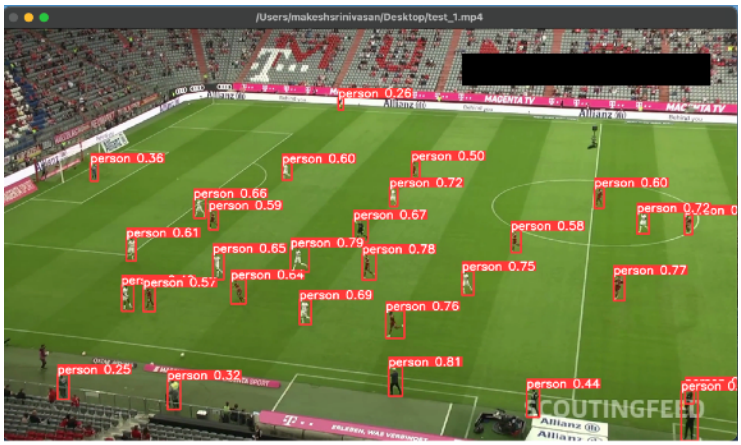



Figure 4.2: COCO standard weights
Figure 4.2 shows the detection of ALL humans in the image or frame using COCO standard weights. The confidence in predictions are mentioned above the bounding boxes, and the values range drastically between 0.4 and 0.7 (range: 0-1).

Experiment 3.5.1.2. (B) COCO custom trained weights

Custom COCO model

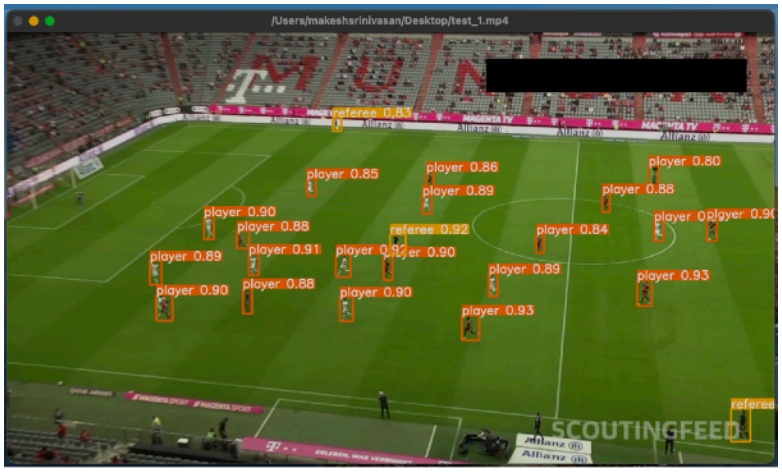



Figure 4.3: COCO custom weights
Figure 4.3 shows the detection of players, referees and line-referees in the image or frame using COCO custom weights trained to detect only non-noisy humans. The confidence values of the predictions range between 0.80 and 0.99 (range: 0-1).

4.2. TRACKING OF AGENTS

4.2.1 MULTI-OBJECT TRACKING - BYTE TRACK (EXPERIMENT 3.5.2.1)

BYTE Tracking

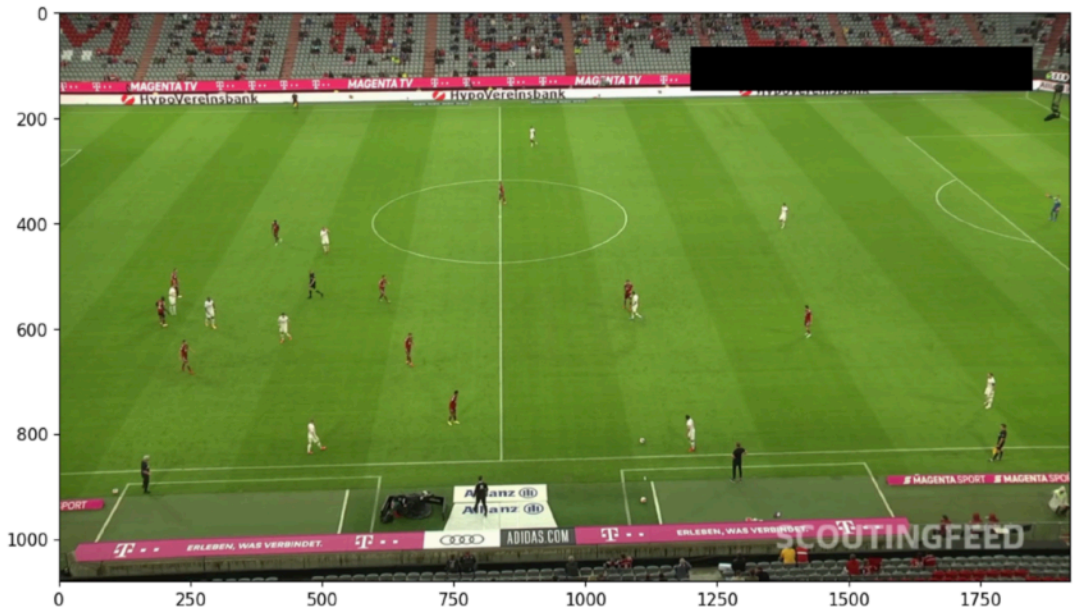


Figure 4.4: Raw frame

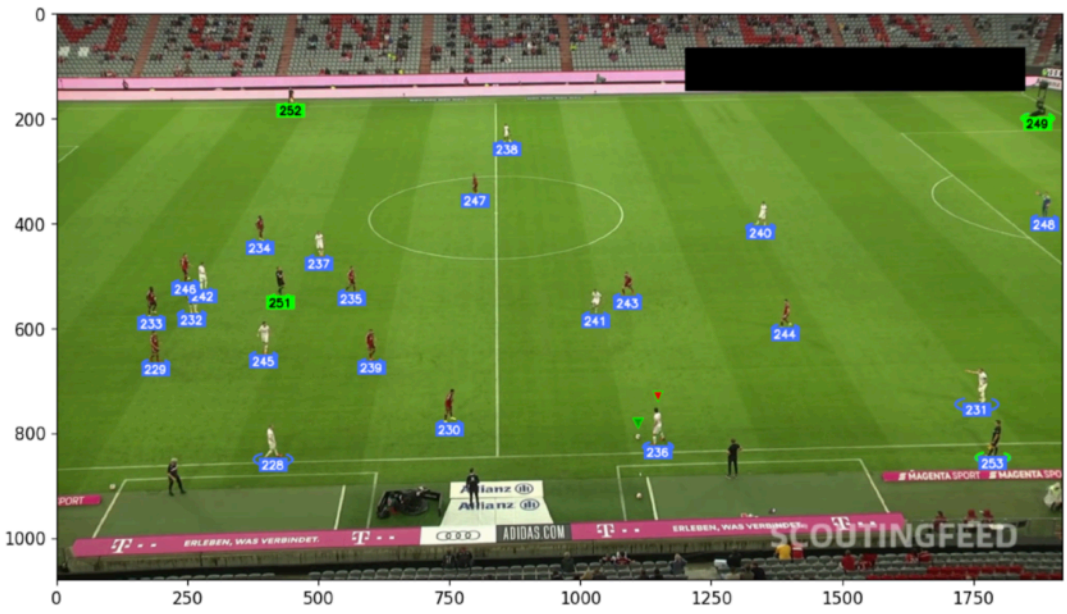
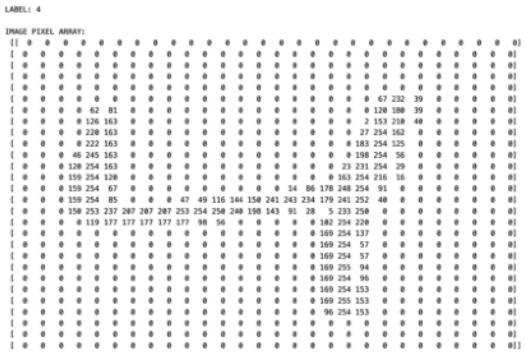
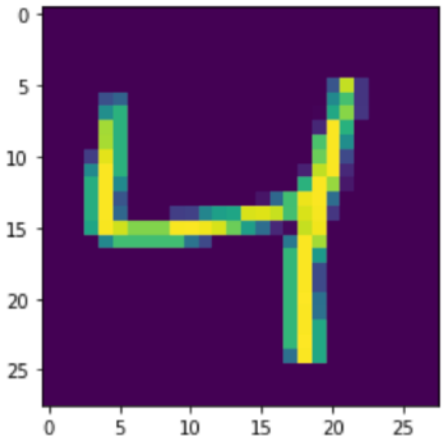


Figure 4.5: BYTE Track of agents

BYTE Tracking
Figure 4.4 shows the raw frame from a video whereas the Figure 4.5 shows the same frame with tracker_ID on all detected players, goalkeeper, referee. Line-referee, and the ball. Additionally, the ball is marked by a small tree triangle at the bottom of the image and the player in possession is marker with a red triangle.

4.3. DETECTION OF JERSEY NUMBER

4.3.1 SEQUENTIAL ARCHITECTURE (EXPERIMENT 3.5.3.1)

A Sequential model- CNN Neural Network
 Prediction: 4 (A): Pixel values  (B): Raw image of a number Figure 4.6: MNIST Sequential CNN model
A sequential model trained using CNNs on the MNIST dataset is able to recognise handwritten digits as shown in Figure 4.6 (A) and (B).

4.3.2 EFFICIENTNET ARCHITECTURE (EXPERIMENT 3.5.3.2)

EfficientNet-B0

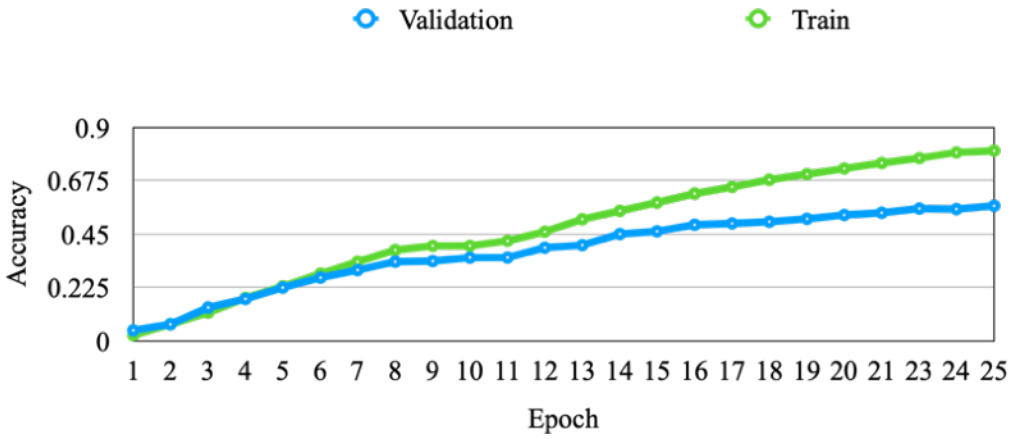


Figure 4.7: Epoch vs Accuracy of EfficientNet-B0

Figure 4.8 (A): Prediction

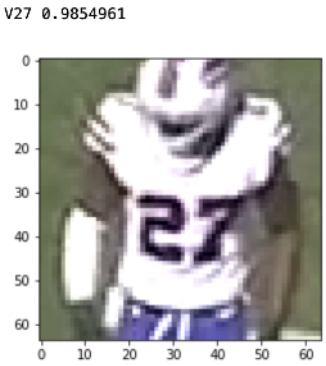


Figure 4.8 (B): Prediction

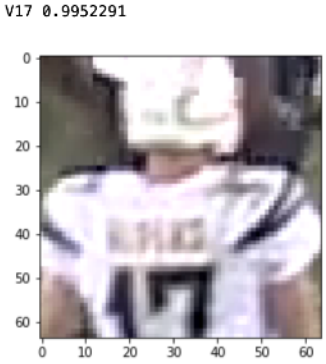


Figure 4.8 (C): Prediction

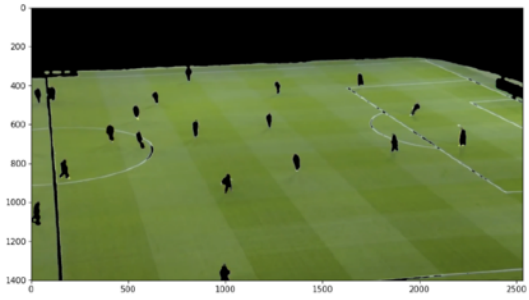
The EfficientNet-B0 trained on the images of NFL players and Football players shows prediction of the jersey number by taking in images of the players. The Figures 4.8 (A), (B) and (C) show the appropriate predictions of the jersey number as shown in the image alone with the confidence value. The Figure 4.7 shows the Accuracy vs Epoch for the train and validation set is also shown to increase over 25 epochs.

4.4. LANDMARK IDENTIFICATION

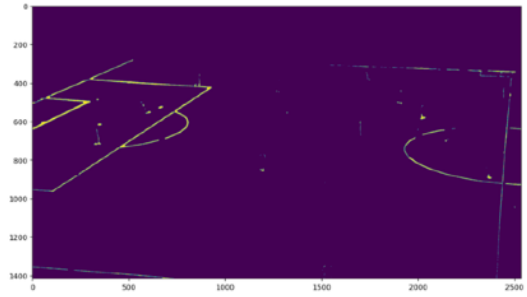
4.4.1 CANNY DETECTION & HOUGH TRANSFORM (EXPERIMENT 3.5.4.1)

Background subtraction and noise removal

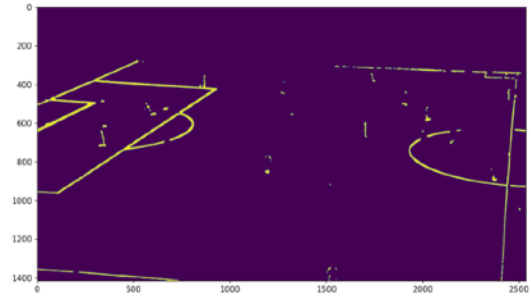
(A) Raw image



(B) Noise removed image



(C) Erosion

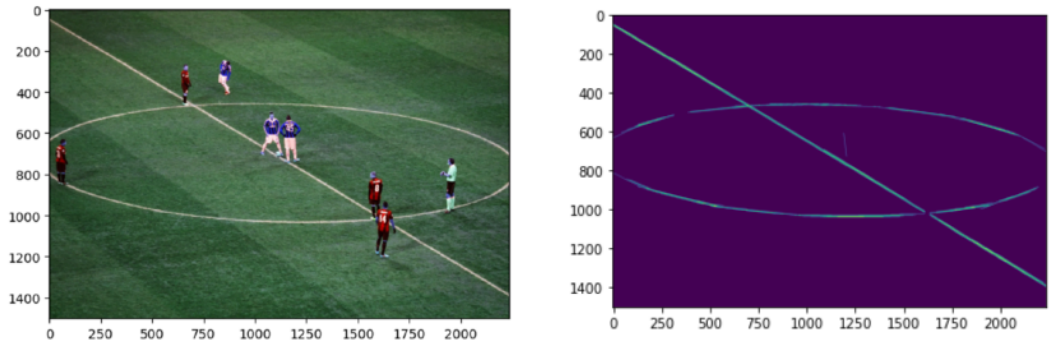


(D) Dilation

Figure 4.9: Noisy-humans removal

In Figure 4.9 (A), the players and audience are present. In (B), the audience and players are removed using background subtraction and several morphological operations. In (C), the white lines are extracted. Finally, in (D), the lines are thickened to make the relevant areas more prominent which can make the edge detection results better.

Morphological operations for Canny edge detection & Hough Lines transform



(A) Raw image (B) Boundary extracted

Figure 4.10: Boundary detection

The Figure 4.10 (A) shows the original image, and after performing a series of morphological operation and canny edge detection, the image (B) shows the boundaries on the field detected.

Final output

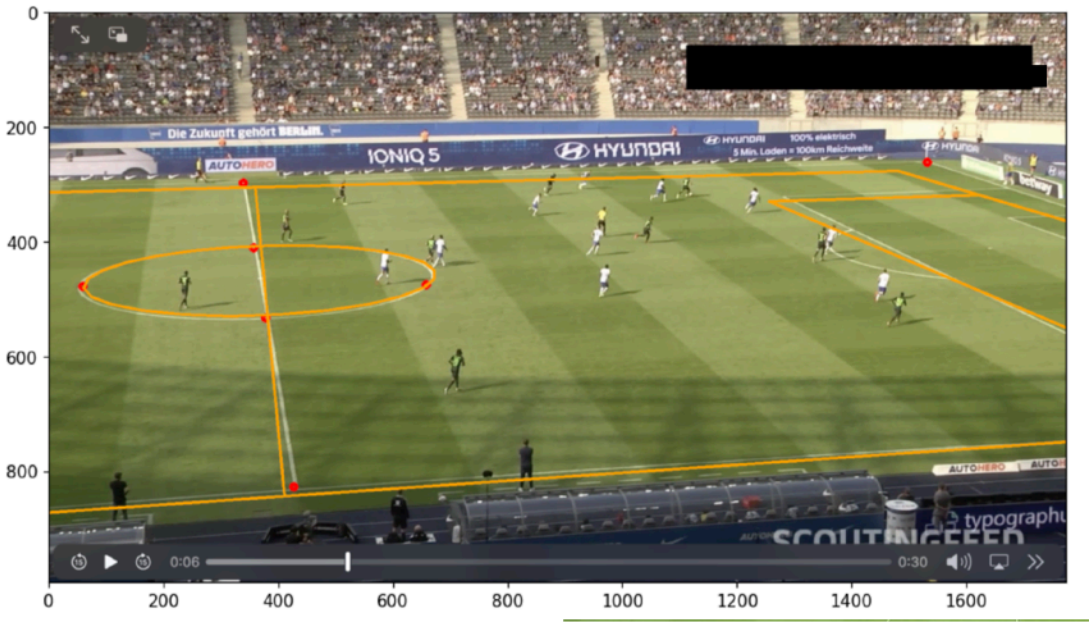
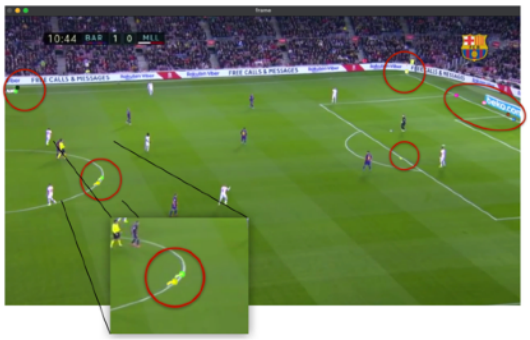


Figure 4.11: Landmark detection overlay on the football field

Figure 4.11 shows the frame with the field lines drawn over the original frame.

4.4.2 OPTICAL FLOW - LUCAS KANADE (EXPERIMENT 3.5.4.2)

Optical flow using Lucas Kanade



(A) Frame 1 at T = 10:44

(B) Frame 2 at T = 10:55

Figure 4.12: Optical Flow using Lucas Kanade algorithm

The Figure 4.12 (A), taken at T = 10:44, has several points that were automatically detected using GoodFeatures. After a second has elapsed (T = 10:45), the points have moved a several pixels as the camera has also moved.

4.4.3 FEATURE MATCHING (EXPERIMENT 3.5.4.3)

Experiment 3.5.4.3. (A) SIFT

SIFT

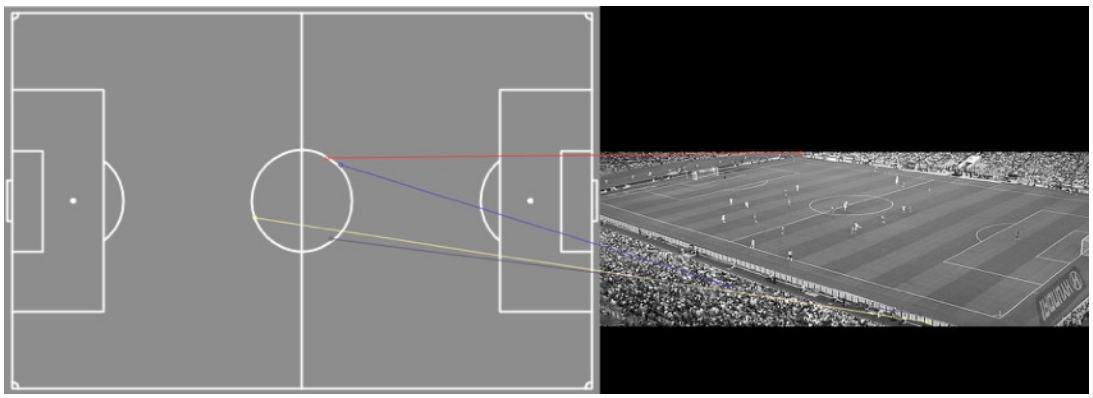
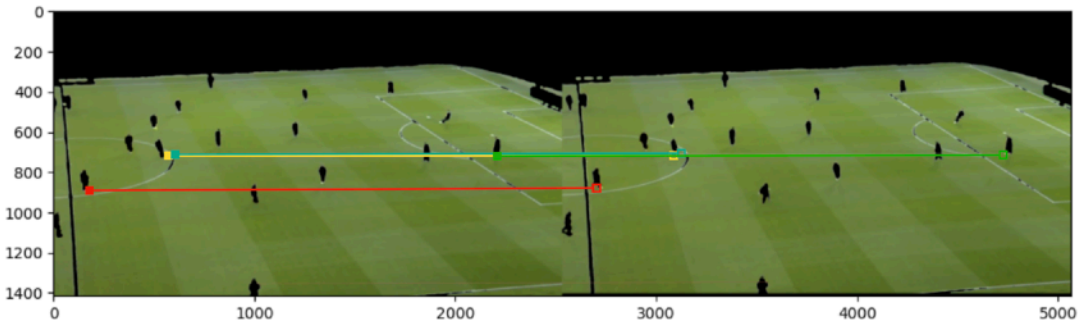


Figure 4.13: Scale Invariant Feature Transform matched 4 points

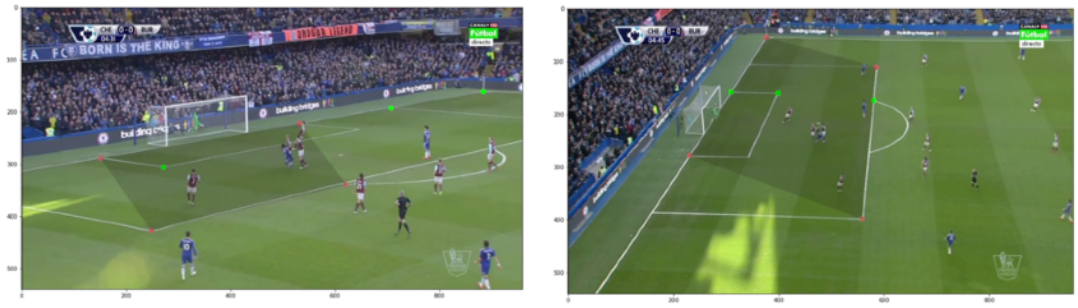
Figure 4.13 shows the subject image matched with the template image on the left.

Experiment 3.5.4.2. (B) ORB

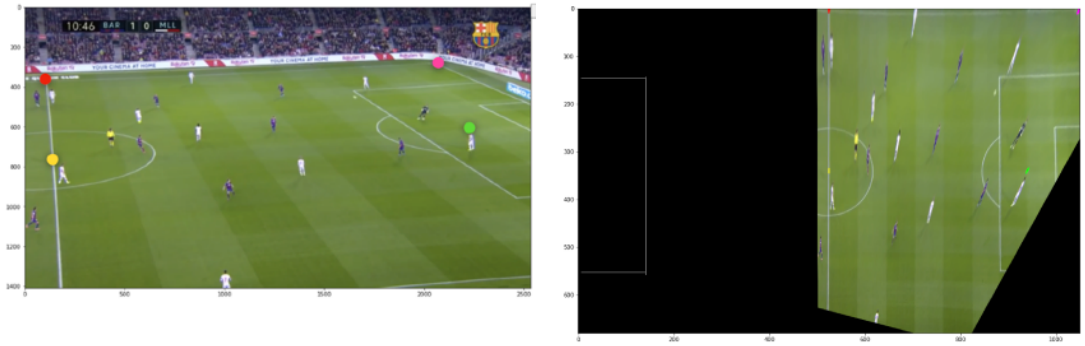
ORB
 Figure 4.14: Oriented FAST and rotated BRIEF matched 4 points NOTE: The lines were not think enough to be visible on print, hence, the image was edited to make the lines appear thicker
Figure 4.14 shows the two images with slight shift in time being matched after removing noisy humans.

4.5. PERSPECTIVE TRANSFORMATION

4.5.1 HOMOGRAPHY MATRIX USING DLT (EXPERIMENT 3.5.5.1)

Selecting appropriate landmarks
 (A) Sample 1 (B) Sample 2 Figure 4.15: Selection of four points with maximum area
The Figure 4.15 (A) and (B) show multiple points on the frame that were acquired through annotated images. Four points were selected in a way that maximised the area enclosed (indicated by red points). The grey region shows the enclosed area.

Perspective transformation from 3D to 2D



(A) Raw image with 4 points

(B) Perspective transformed image

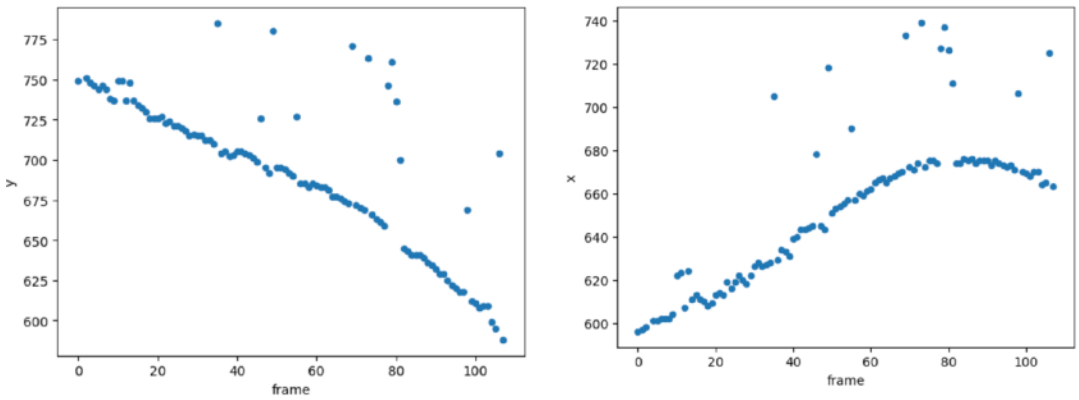
Figure 4.16: Perspective transformation using 4 points

The Figure 4.16 (A) shows the four points manually placed on the image. After performing perspective transformation, the image on the right (B) shows the appropriately sheered and transformed image placed in proportion on a virtual 2D board.

4.6. POSITIONAL ESTIMATION: TIME SERIES FORECASTING

4.6.1 PRE-REQUISITE: OUTLIER REMOVAL (EXPERIMENT 3.5.6.1)

Outliers removal in positional coordinates



(A) Y direction motion over time

(B) X direction motion over time

Figure 4.17: Raw data of the motion of a player over time/frame

Outliers removal in positional coordinates

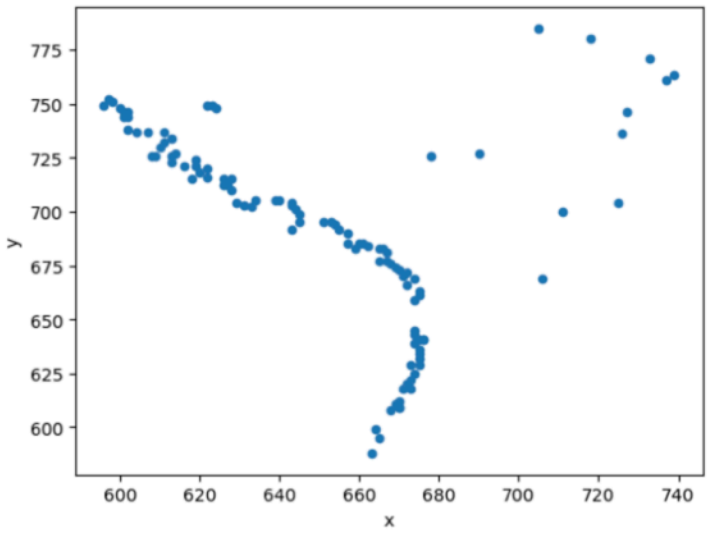
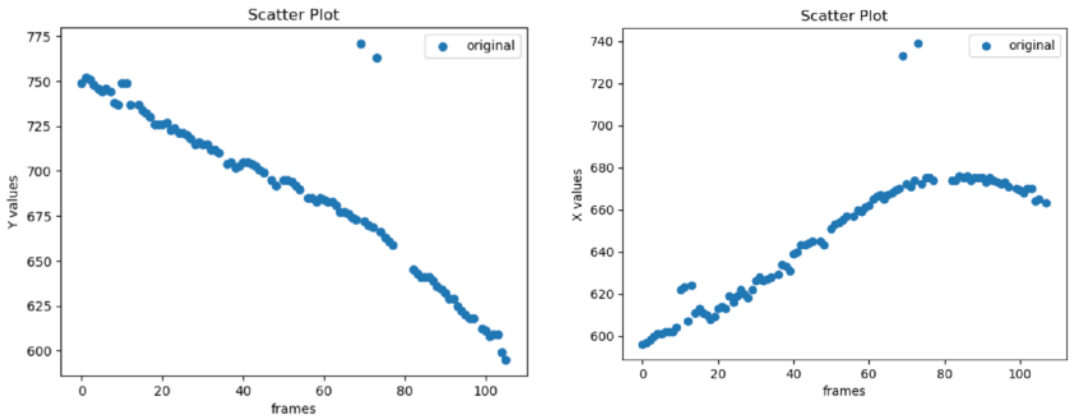


Figure 4.18: Motion of a player in X-Y plane (Raw data)

The Figure 4.17 (A) and (B) shows the raw player coordinates (x and y respectively) of a random player running on the field during a game. Evidently, some points are missing or having erroneous values. The Figure 4.18 shows the exact motion over a x-y plane with outliers and missing values as well.



(A) Y direction motion over time

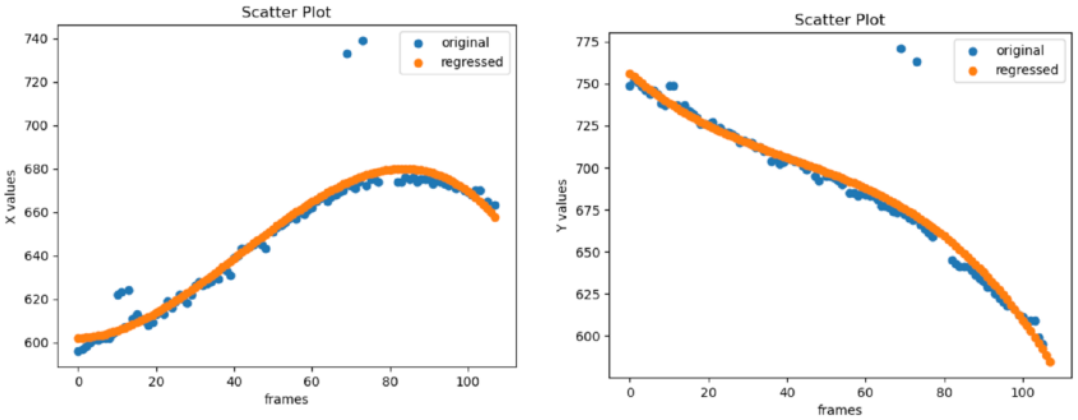
(B) X direction motion over time

Figure 4.19: Motion of a player (Outliers removed)

The Figures 4.19 (A) and (B) show the data of x and y positions of the same player after removing the outliers.

4.6.2 INTERPOLATION: POLYNOMIAL REGRESSION (EXPERIMENT 3.5.6.1)

Interpolation: Polynomial Regression of 4th degree to impute data



(A) X direction motion over time

(B) Y direction motion over time

Figure 4.20: Motion of a player (Interpolated polynomial regression)

The Figures 4.20 (A) and (B) show the regressed plot over frame (4th degree polynomial) in orange points for x and y respectively, whereas the raw outlier-removed data points of x and y positions of the same player are shown in blue.

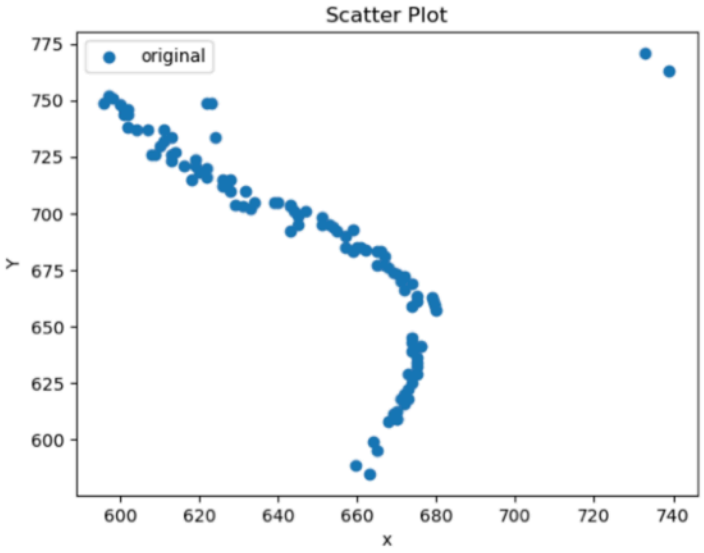
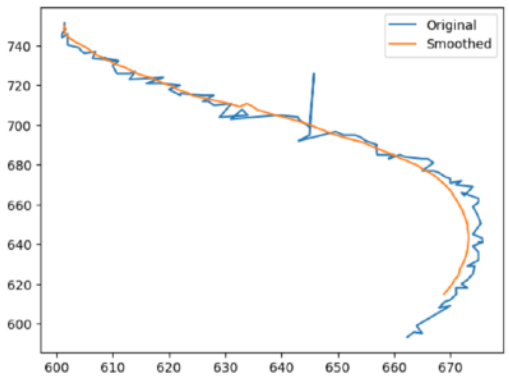


Figure 4.21: Motion of a player in X-Y plane (Interpolated polynomial regression)

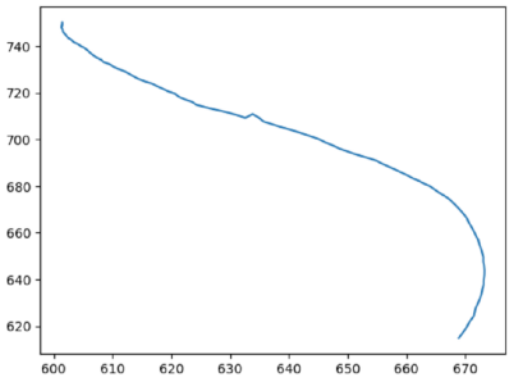
The Figures 4.21 shows the data of x and y positions of the same player after removing the outliers and imputing the gaps using regressed predictions.

4.6.3 KALMAN FILTER (EXPERIMENT 3.5.6.1)

Kalman Smoothing of players motion



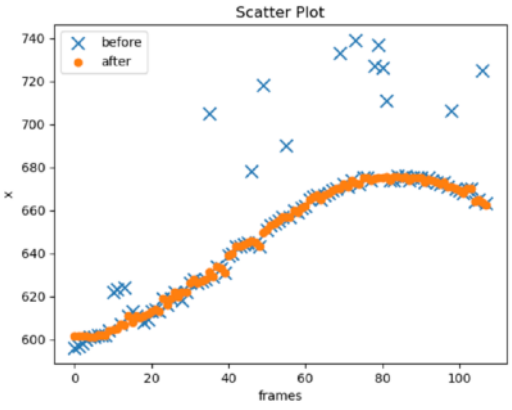
(A) Original vs smoothed



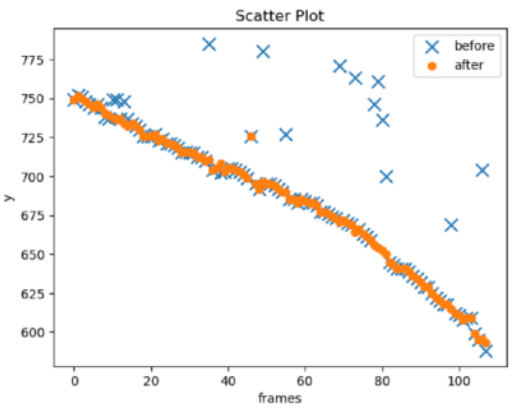
(B) Smoothed

Figure 4.22: Kalman-Smoothened motion of a player in X-Y plane

The Kalman smoothening is done over the plots x-frame and y-frame and shown here in x-y plane in Figure 4.22 (A) as a smoother orange curve over the blue original curve, which was smoothened using only the outlier removed data (not the regressed-imputed data). The Figure (B) shows the former over x-y plane.



(A) X direction motion over time



(B) Y direction motion over time

Figure 4.23: Motion of a player (Raw and regression)

Kalman Smoothing of players motion

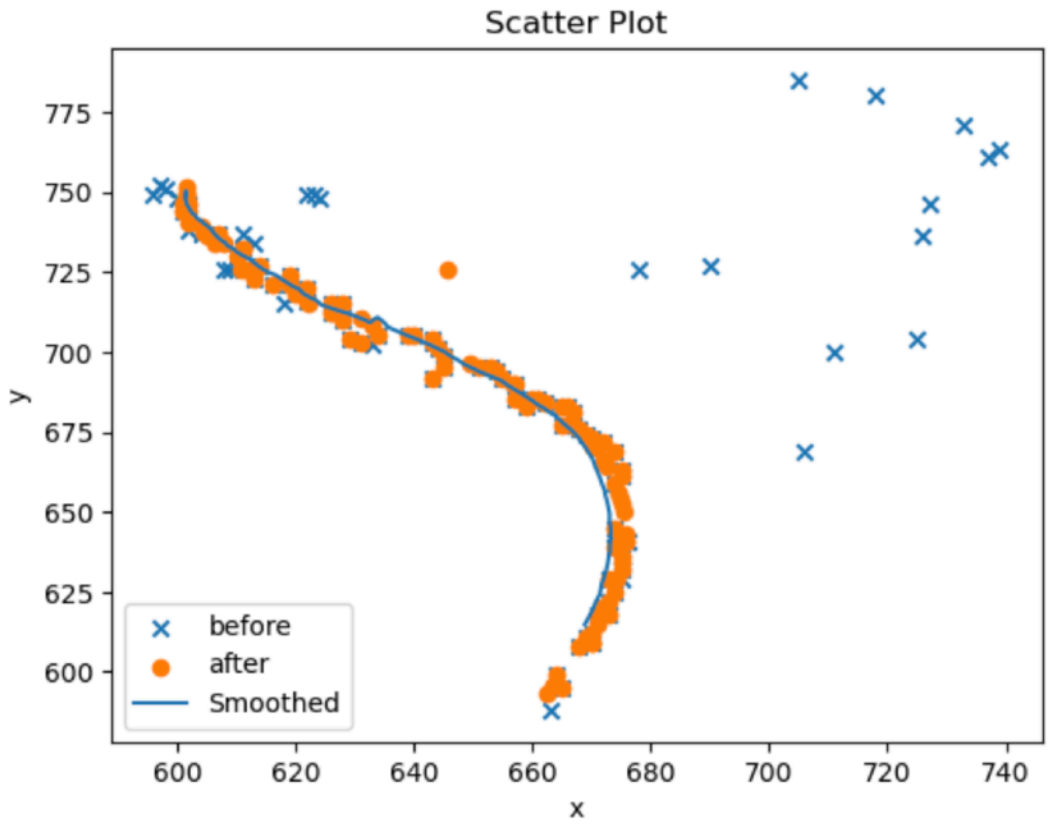


Figure 4.24: Motion of a player in X-Y plane (Kalman smoothened)

The Figures 4.23 (A) and (B) show the motion of the player in x and y directions respectively as a scatter plot. The blue "x" represent the raw data with missing values and outliers. The orange circles represent the filled-in values after interpolation. The Figure 4.24 shows the same motion in a x-y plane with the Kalman smoothening function applied which is indicated by the smooth blue colour curve.

4.6.4 LSTM: LONG SHORT-TERM MEMORY (EXPERIMENT 3.5.6.2)

Experiment 3.5.6.2. (A) LSTM

LSTM - Prediction vs True values

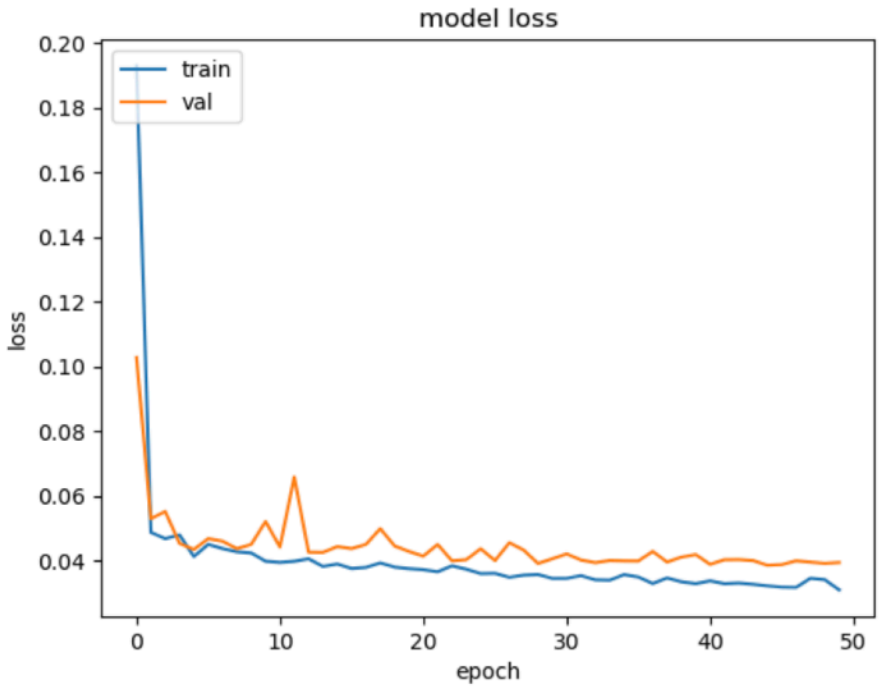
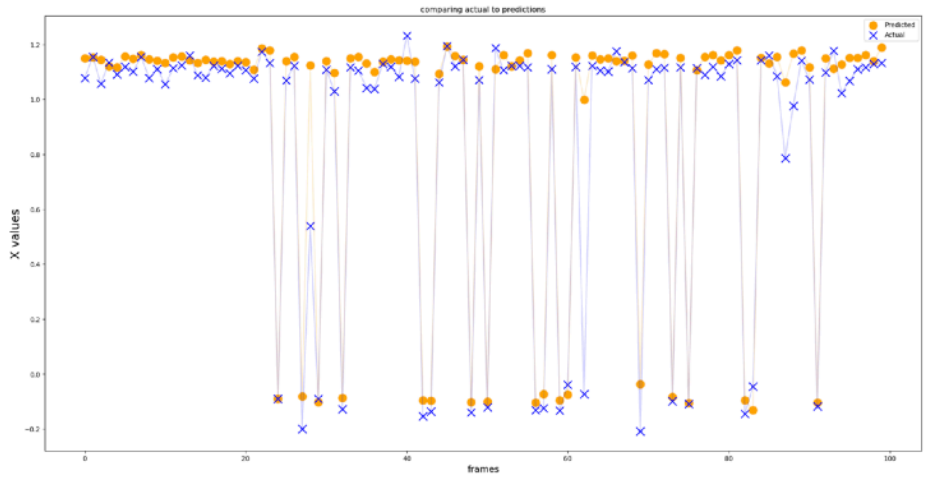
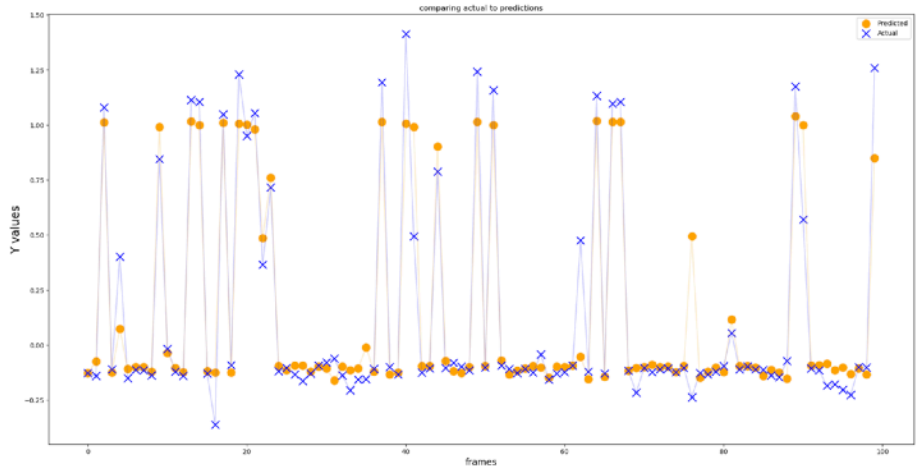


Figure 4.25: Train and Validation epoch vs loss



(A) X values prediction vs true values

LSTM - Prediction vs True values



(B) Y values prediction vs true values

Figure 4.26: Prediction vs True values for LSTM

Figure 4.25 shows the epoch vs loss plot for the LSTM sequential architecture. The Figure 4.26 A and B show the first 100 predictions and how they differ from the original values. Orange dots represent predictions while blue crosses represent actual values.

Experiment 3.5.6.2. (B) Bidirectional LSTM

Bidirectional LSTM

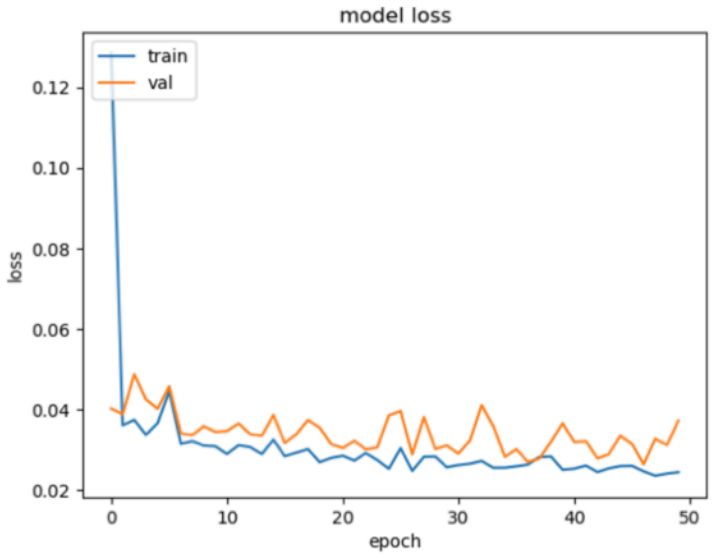
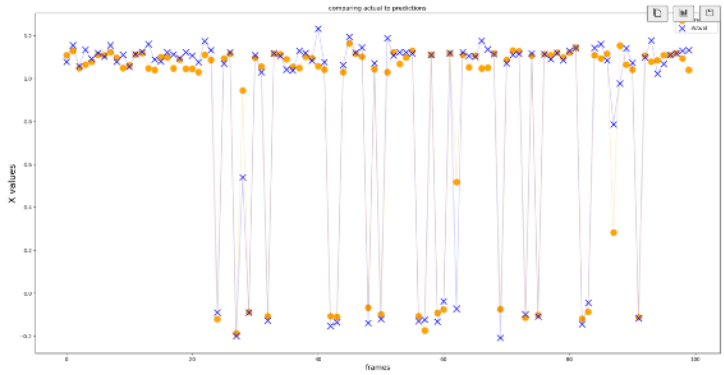
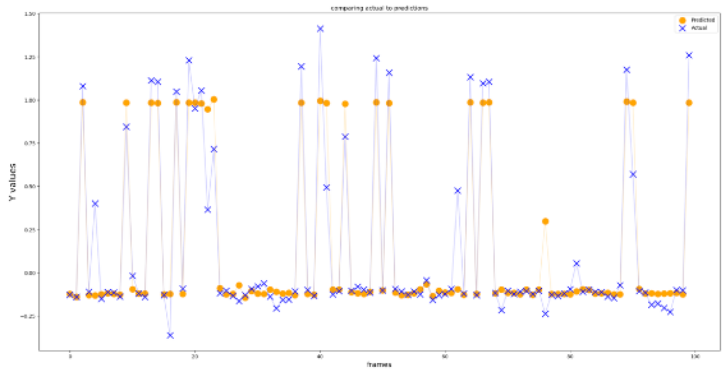
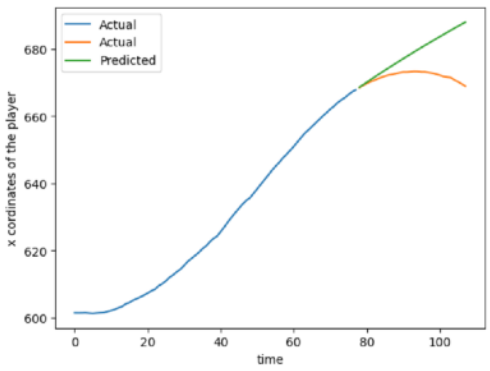


Figure 4.27: Epoch vs Loss

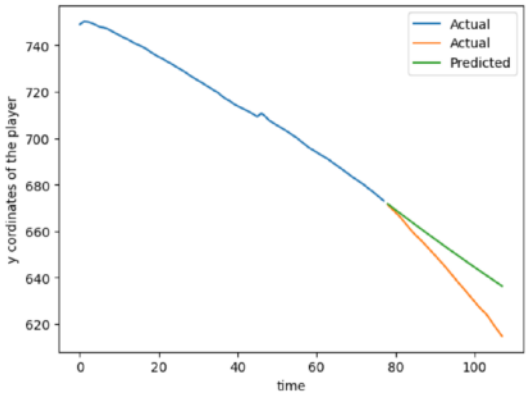
Bidirectional LSTM
   (A) X values prediction vs true values    (B) Y values prediction vs true values Figure 4.28: Prediction vs True values for Bidirectional LSTM
Figure 4.27 shows the epoch vs loss plot for the Bidirectional LSTM sequential architecture. The Figure 4.28 A and B show the first 100 predictions and how they differ from the original values. Orange dots represent predictions while blue crosses represent actual values.

4.6.5 ARIMA (Experiment 3.5.6.3)

Autoregressive Integrated Moving Average model



(A) X direction motion over time



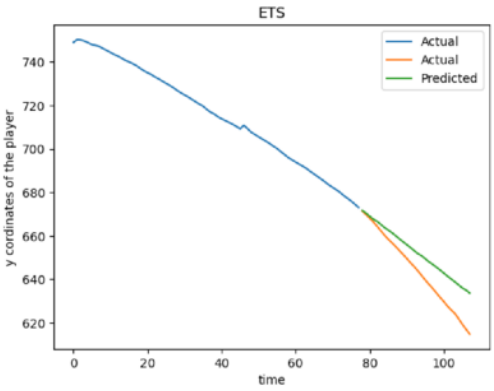
(B) Y direction motion over time

Figure 4.29: Actual and predicted motion using ARIMA

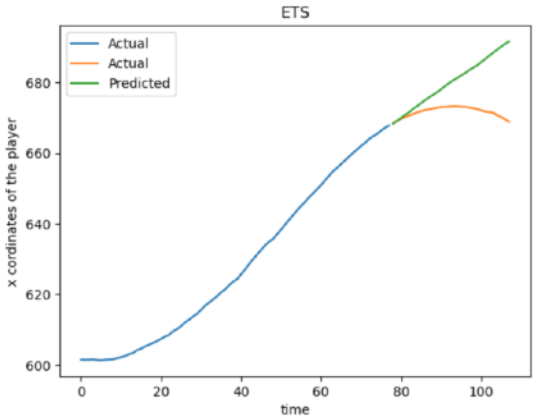
The Figure 4.29 (A) and (B) show the ARIMA model trained on x and y coordinates respectively of player motion to predict the last 30 time steps of positions. The plots show predictions (green), test actual (orange) and train actual (blue).

4.6.6. EXPONENTIAL SMOOTHENING (EXPERIMENT 3.5.6.4)

ETS



(A) Y direction motion over time



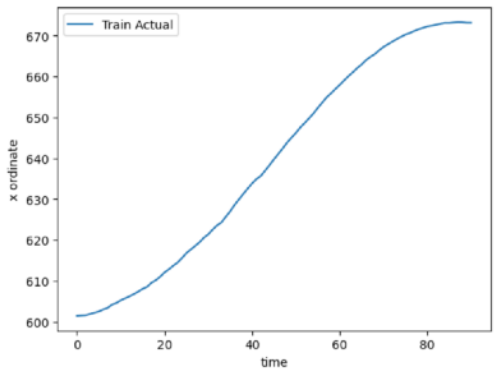
(B) X direction motion over time

Figure 4.30: Actual and predicted motion using ETS

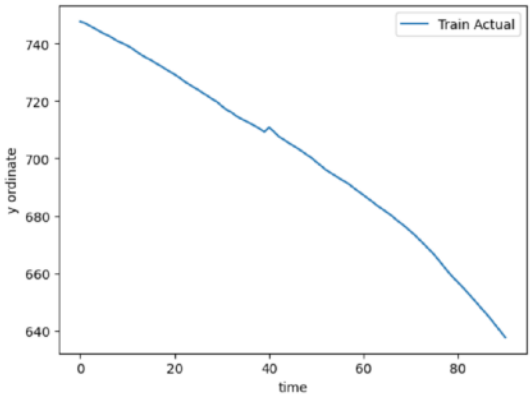
The Figure 4.30 (A) and (B) show the ETS model trained on x and y coordinates respectively of player motion to predict the last 30 time steps of positions. The plots show predictions (green), test actual (orange) and train actual (blue).

4.6.7. XGBOOSTING (EXPERIMENT 3.5.6.5)

XGBoost

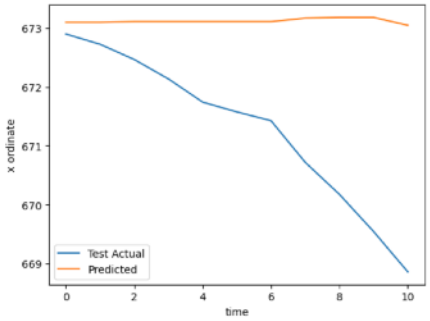


(A) X direction motion over time

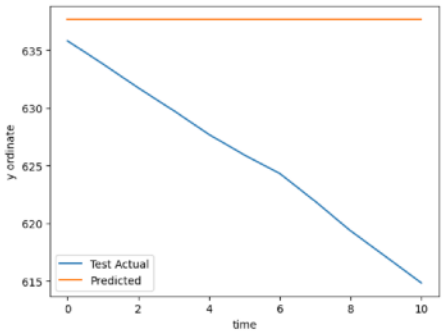


(B) Y direction motion over time

Figure 4.31: Train data of x and y motion



(A) X direction motion over time



(B) Y direction motion over time

Figure 4.32: Prediction vs true value of x and y over time

The Figure 4.32 (A) and (B) show the XGBoost model trained on x and y coordinates respectively of player motion to predict the last 1/10 of the time steps of positions. The plots show predictions (orange), test actual (blue). The Figures 4.31 (A) and (B) show the train set used for x and y positions.

4.6.8. VECTOR AUTO-REGRESSION (EXPERIMENT 3.5.6.6)

Vector Auto-Regression (VAR)

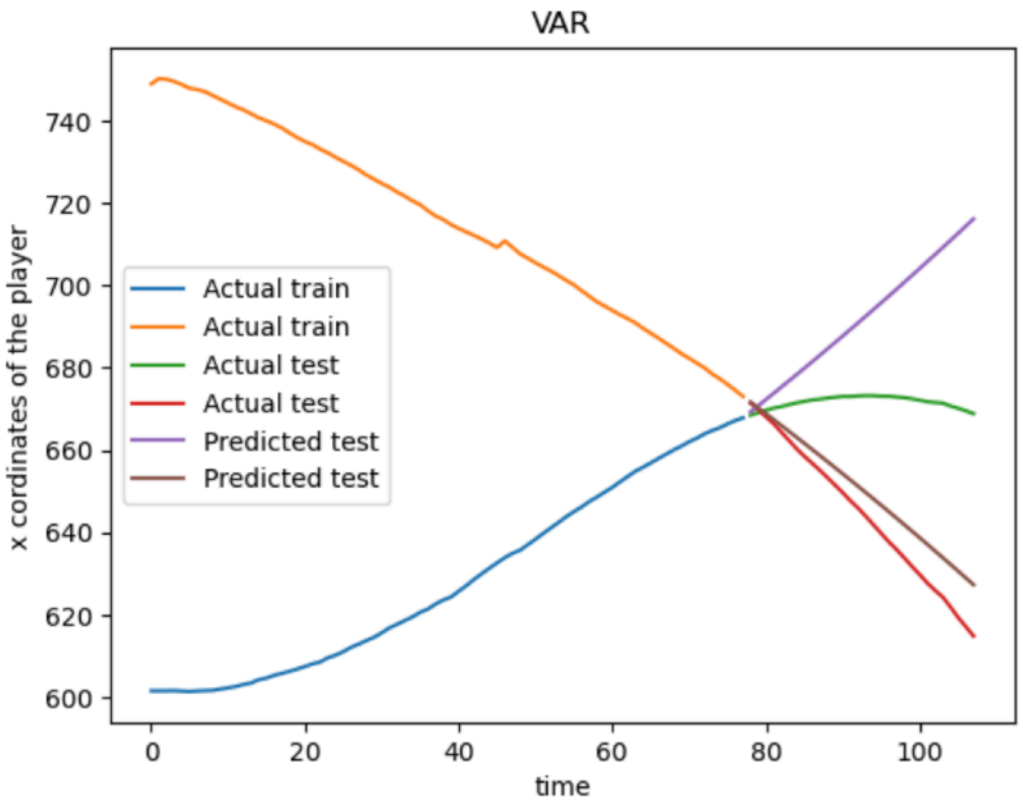


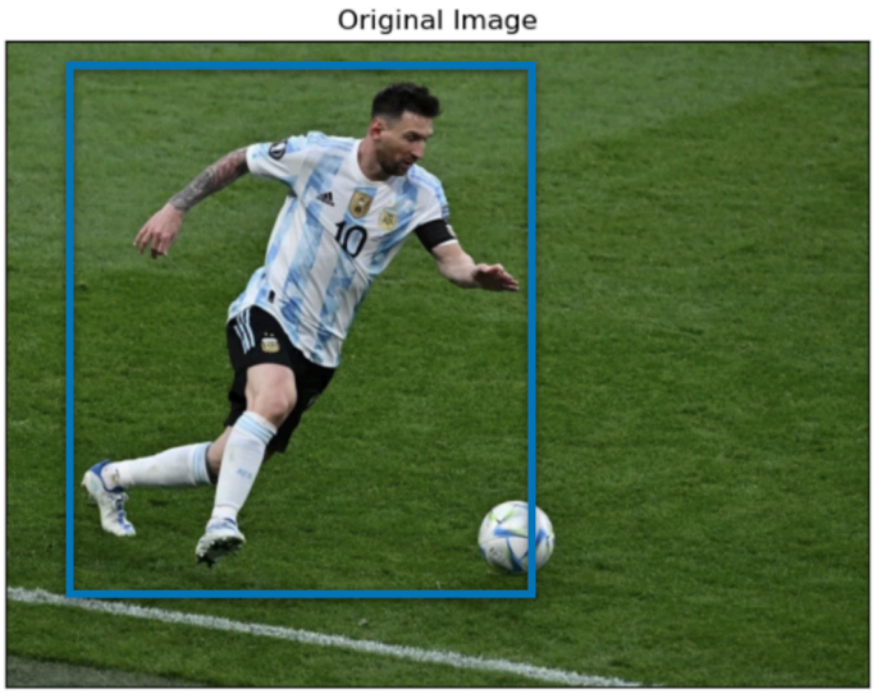
Figure 4.33: Actual train and test & predicted test of x and y motion

The Figure 4.33 shows the VAR model trained on coordinates of player to forecast the last 30 time steps of positions. The plots show predicted test for x (indigo) and y (red), actual test for x (green) and y (brown), and actual train of x (blue) and y (orange).

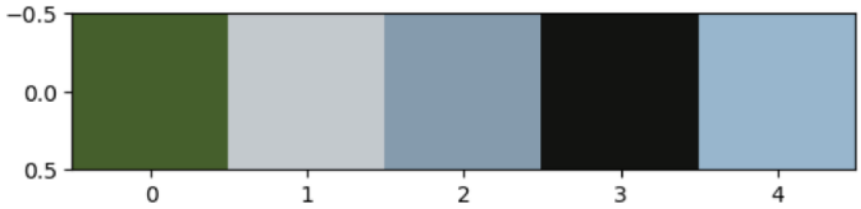
4.7. MISCELLANEOUS EXPERIMENTS

4.7.1 TEAM DETECTION (EXPERIMENT 3.5.7.1)

Detection of dominant colours



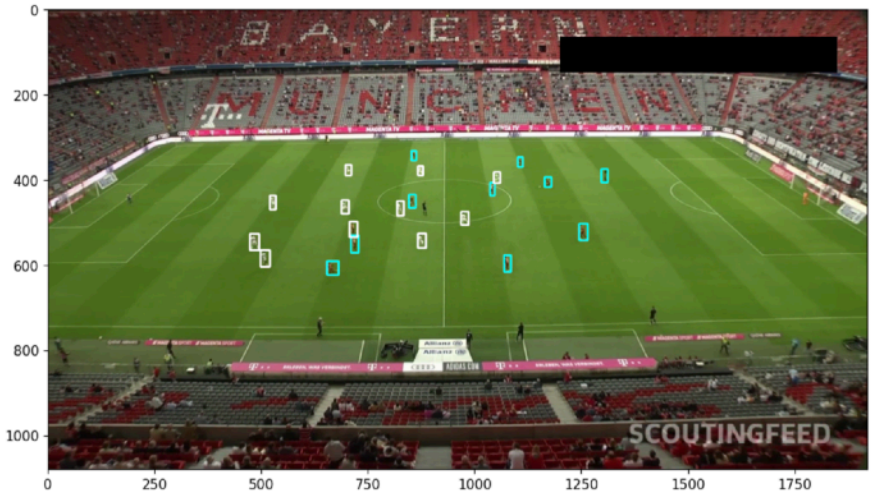
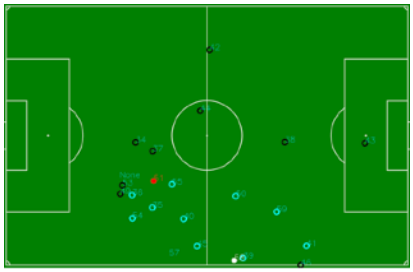
(A) A player with a bounding box



(B) Top 5 most available colour in the bounding box

Figure 4.34: Most available colour using K-Means

In the Figure 4.34 (A), the player has donned a blue-white jersey and therefore the colour of his team will also be blue-white. Using K means clustering, the highest availability of colours in the bounding box surrounding this player is present in Figure 4.34 (B). As shown, the colour green is evidently dominant and therefore, it is shown as the highest available colour, followed by white-blue (and black due to the bottom part of his outfit). As already mentioned, the algorithm returns colours from index 1 till the end to avoid grass colour dominance within the domain.

Detection of dominant colours
 Figure 4.35: Teamed-Up colours of players
The bounding boxes of all players are shown in the colour of their team in Figure 4.35. Blue is for Team 1 while White is for Team 2. The bounding boxes for referees are removed by filtering out their class_ID.
  (A) Detected agents (B) Teamed-up players display Figure 4.36: Display of teamed-up players on 2D virtual display
The detected players (Figure 4.36 A) are now clustered into two teams and displayed on the virtual representation as shown in Figure 4.36 (B). The blue represents players of one team while the black represents another.

Chapter 5
Evaluation and Discussion

This chapter delineates the evaluation of the modules as well as discusses the limitations and prospects of expansion along with the limitations.

5.1. INTRODUCTION

In section 5.2, I have illustrated the details of the modules within the research. Additionally, the performance metrics and qualitative analysis are provided with module-wise inferences as follows. The future works are explained in the subsequent chapter. The quantitative performance metrics used are as follows.

(i) **Mean Squared Error (MSE):** MSE measures the average squared difference between the predicted and actual values of the label. The lower the MSE, the better the model's predictions. However, since MSE is calculated by squaring the differences between predicted and actual values, it can be sensitive to outliers in the data.

(ii) **Mean Absolute Error (MAE):** MAE is another measure of how well a regression model is able to predict continuous target variables. It measures the average absolute difference between the predicted and actual values of the target variable. Unlike MSE, MAE is not sensitive to outliers in the data. However, it may not penalise larger errors as heavily as MSE.

(iii) **R squared (R^2):** R squared is a measure of how well a regression model fits the data. It measures the proportion of variance in the target variable that can be explained by the model. The closer R squared is to 1, the better the model fits the data. An R squared value of 0 means that the model does not explain any of the variance in the target variable. A negative R squared value means that the model is worse than a model that simply predicts the mean of the target variable.

(iv) **Accuracy:** Accuracy is a commonly used metric to evaluate the performance of classification models. It measures the percentage of correctly classified instances out of all instances in the dataset. In other words, it measures how often the model makes correct predictions.

(v) **Confidence:** Confidence is a measure of how certain a model is in its predictions. In the context of machine learning, confidence is often associated with the concept of prediction intervals or confidence intervals, which provide a range of possible values within which the predicted value is likely to fall with a certain level of confidence. In my research, the value ranges between 0 and 1.

5.2. EVALUATION

5.1. DETECTION OF AGENTS

Among the three results (shown in Figures 4.1, 4.2 and 4.3), the custom weights outperform the other two by a significant margin reflected by the range of the confidence values (0.80 to 0.99).

The model is also capable of accurately distinguishing the referees, goalkeepers, line referees and the players. The noisy humans such as spectators and managers are also removed effectively.

Hence, the Custom COCO weights yielded the best results in this module, and thus, will be used in the subsequent modules.

5.2. TRACKING OF AGENTS

The relevant agents are football, players, referees and goalkeepers. They are also called the non-noisy humans while the rest of the humans such as managers and audience are noisy humans. As mentioned in 5.1 detection of agents, the coco custom weights remove the noisy humans and only pipeline the non-noisy ones.

Here, the BYTE tracking algorithm assigns a unique tracker_ID to all agents and monitors the motion of the agents throughout the video. As shown in Figure 4.5, the relevant agents are marked with a unique ID in blue beneath the agents. The line referees as well as the on-field referees are also labelled with a unique ID.

Additionally, the ball is tracked and marked using the upside down triangle on the top showing its location on the field. When the ball comes within a 20 pixels radius of a player, the player is assigned the possession of the ball. This means, until the possession changes or lost for prolonged period of time, he is in possession of the football. This is considered as dribble in the data collection module of the research.

When the possession is transferred to a different player of the same team, the event is now recorded as a pass. If it is done to a different team, then the ball has been lost by the team.

Moreover, the detections and the coordinates are continually tracked and BYTE tracking algorithm does an excellent job at doing so. This will be extremely crucial in the subsequence modules.

5.3. DETECTION OF JERSEY NUMBER

Initially, the MNIST dataset was used to train a standard sequential model that was capable of predicting the handwritten digits. The idea was to perform transfer learning by heavily preprocessing the test images from actual football games. However, this proved to be very arduous, time taking and inefficient.

Therefore, a different approach via EfficientNet was adopted that used NFL dataset and Football images to make predictions on the jersey number directly. This model was trained for over 25 epochs and the training accuracy yielded was 0.88 whereas the validation was 0.67. With increase in epochs, this value could be significantly improved as the graphs have not saturated yet. Due to hardware limitations, the model's training epochs was restricted.

However, it still was able to accurately predict jersey numbers of players even when they are only a few pixels wide. Hence, EfficientNet-B0 yielded the best results in this module.

5.4. LANDMARK IDENTIFICATION

Firstly, the idea of finding feature matches to determine the landmarks and its association with the subject image (test frame) was designed and implemented. This experiment (Experiment 3.5.4.3 - feature matching) was implemented in two parallel stream - ORB and SIFT. SIFT algorithm shown in Figure 4.13 showed that the model was very incapable of making accurate matches without heavy pre-processing involved. Considering the implementation is real-time, it was crucial that the modules performed the job fast and effectively. ORB algorithm was implemented using two pictures taken a few frames apart. The Figure 4.14 shows that the model was able to accurately pinpoint the pixels that were associated with each other; however, this required the prior knowledge of what the pixels meant, i.e, the centre, corner, goal-

post, etc. This was an arduous task of loading all information and therefore, decided to perform automatic feature detection and tracking.

This was done using Lukas Kanade algorithm (optical flow) as shown in Figure 4.12 A and B, which was 1 second apart. The goodFeatureDecetctions module of OpenCV was automatically able to detect the features that were highly contrasted such as the corners and edges - white against the green surroundings, players boundary, ball against the field, etc. The figures show that the points were tracked and the motion was computed in (A) and (B). This could be used to perform perspective transformation directly; however, the points could be lost as the points disappear when the camera pans away. The chances of the same points being detected were also very low. This meant that the model will become less and less capable of learning the orientation and geometry of the field over time as the footages are played. The Lucas Kanade algorithm will lose the points as they decay and therefore, the model will eventually run out of points (less than 4) to do perspective transformation.

Hence, I decided to use morphological operations and canny edge detection to overlay the lines and points of the football field which proved to be quick and efficient as shown in Figure 4.11. The morphological operations such as dilation and erosion are shown in Figure 4.9 whereas the boundary detection is shown in Figure 4.10 A and B.

Thus, the best results were shown by canny edge detection and hough lines transformation experiment and therefore will be used in the subsequent modules.

5.5. PERSPECTIVE TRANSFORMATION

While there is no quantitative means to measure the performance of this module, it is observed very clearly that the shift in perspective is accurately represented on the virtual 2D representation. All the agents and their coordinates are also reflected accurately. However, occasionally, there are several outliers and missing values in getting the perspective transformation of the agents due to the lag in landmark detection and the speed and the direction of the camera’s movement.

5.6. POSITIONAL ESTIMATION USING TIME SERIES FORECASTING

In this module, there were three pre-requisite experiments that were necessary to be used subsequently in the time series forecasting deep learning models. Firstly, the raw data of the players collected using the perspective transformation module, has a lot of

missing values and outliers due to their erratic collection of data which occurs when the camera pans in multiple directions at the same time.

These values must be corrected before the model is trained on it. Hence, the replace_outliers() function runs a moving window of size 10. Once the window is defined, the function calculates the mean and standard deviation of the non-missing values within the window using np.nanmean() and np.nanstd(). If the absolute difference between the current element and the window mean is greater than threshold times the window standard deviation, the current element is replaced with np.nan. This eliminates the outliers by replacing the values with NaN. This is depicted in the Figure 4.19 A and B. The original data shown in Figure 4.17 and 4.18 have multiple outliers which are replaced.

Secondly, the missing values must be interpolated. Here, I used polynomial regression of degree 4 to generate a function that can learn and model the current datapoints x and frame, and y and frame. Once the regression model is generated, the missing values are regressed and imputed using the appropriate functions. This is depicted in the Figure 4.20 A and B where the orange dots repress the regressed points while the blue dots represent the original points. Figure 4.21 shows the final motion of the player after noise removal and interpolation.

Thirdly, the Kalman filter is used to smoothen the motion of the player. As shown in Figure 4.22 A and B, the Kalman filter performs data smoothing which will be used by the deep learning model and forecasting models. The code or the implementation of the above mentioned experiments are provided in the Appendix 1, under 1.1.1 and the following model (LSTM) is provided in Appendix 1, 1.1.2. (NOTE: only this section is shared in the appendix as it is the novelty of my research as there was less extensive work in player position estimation in football).

Once the data is processed and ready, the preparation stages include window normalisation and reshaping into a 3D tensor to be used in time series forecasting. The Figure 3.9 and 3.8, show the general architecture that pipelines the tensor to make positional estimates. Figure 3.7 shows the tensor sequencing.
Once the LSTM model is trained using the mentioned architecture, the results are acquired which are represented in the Figure 4.25 and 4.26 A and B. The former shows the loss vs epoch over time for validation and train, which do not vary significantly. This suggests that the model is able to learn and predict for unforeseen

data as well. This is because, I ensured to shuffle the data while splicing it before training and testing which ensured that the model is exposed to multiple kinds of observations.

The actual vs predicted test data is also shown in the Figure 4.26 which suggest that the difference in value of the prediction is negligible and the player's motion is accurately and effectively calculated. The RMSE was as low as 0.216 and R-squared was 0.817 which means 82% of the dataset was understandable and learnable by the model. A low MAE also suggests that the model is able to learn without much errors.

Similarly, the bidirectional LSTM was also trained using the architecture mentioned in Figure 3.9. The epoch vs loss shown ins Figure 4.27 suggests that the model is performing robustly and covers all data well. Figure 4.28 displays the trend of actual and predicted values of the test dataset. The difference between them is again negligible. The R-squared value was found to be 0.781 and RMSE is 0.237. Hence, based on the provided performance metrics, the LSTM architecture appears to be better than the Bidirectional LSTM architecture.

Firstly, the RMSE (Root Mean Squared Error) of the LSTM model is lower than that of the Bidirectional LSTM model. RMSE measures the average magnitude of the errors made by the model, and a lower value indicates better accuracy.

Secondly, the R-squared value of the LSTM model is higher than that of the Bidirectional LSTM model. R-squared is a statistical measure that represents the proportion of the variance in the dependent variable (target) that is predictable from the independent variables (features). A higher R-squared value indicates that more variance in the target variable is explained by the model's predictions.

Finally, the MAE (Mean Absolute Error) of the LSTM model is only slightly higher than that of the Bidirectional LSTM model, but the difference is relatively small.
Therefore, based on the provided performance metrics, the LSTM architecture is the better model in terms of accuracy and explanatory power.

Other time series forecasting models such as VAR, ARIMA and ETS are shown to be not as effective as displayed in the Table 4.7. The prediction capacity of ARIMA is very low as shown in Figure 4.29 A and B. The green predictions are far from the actual orange values. The performance metrics are also lower in comparison with

LSTM or Bidirectional LSTM. The ETS model was also facing similar issues due to their inability to account for multiple parameters like LSTM or other RNN models. This is shown in Figure 4.30.

XGBoost regressor with an x and y separate prediction models (learning rate of 1 and max depth of 9 and 6 respectively) also showed that the prediction capacity was far less as it could not account for the historical information well enough, as depicted in Figure 4.31 and 4.32. The same issues were observed in VAR shown in Figure 4.33. Hence, it is clear that the models capable of remembering the past data of a few time steps, are more capable of making holistic and well-rounded predictions of positions of the players.

Therefore, LSTM and Bidirectional LSTMs were more suitable for this research.

5.7. MISCELLANEOUS EXPERIMENTS

In order to determine the team of the players, the colours of the jerseys of the players can be used. Firstly, I used the bounding boxes to create images of the players bounded by the bounding box yielded by the BYTE tracking algorithm using custom coco weights. With this sub-image, I used K-means clustering to return 5 most abundant colours within this domain, as shown in Figure 4.34 (A and B).

It is clear that the most abundant colour will the colour of the field -green- most of the times, if not always. Hence, the algorithm returns only the next 4 colours in the order of highest to lowest availability. The index=1 colour is assigned to the player. Similarly, the same is repeated for all players on the detection frame, including the referees. All the colours are stored in an array.

Then, K-Means clustering is applied once again with 3 clusters this time on the colours of the players. The idea is to cluster two teams and referees into 3 classes totally.

The clusters are then assigned team 1 and team 2 and referee clusters are the ones with lowest number of detections and therefore, they are filtered out while annotating the images using their class_ID. Figure 4.35 and 4.36 show the detected teams being annotated and displayed on the 2D virtual board.

5.3. LIMITATIONS

Even though the tracing is done in real-time, the lag in each detection is quite high due to hardware limitation of my computer. If implemented on a powerful GPU, the performance would be more seamless and quick. Considering that this implementation is for broadcasting firms that ca utilise this tool to enhance viewer's experience, hardware will not be as big a concern for them as it can be for consumers.

Secondly, the modules are not all loaded into the main function. The detection, tracking, landmark detection and perspective transformation are integrated as there is only one large architecture involved. The jersey detection module is not loaded into the main flow as it causes many hindrances in the performance of the model. Additionally, the positional estimation model is also not loaded into main flow as it is meant to be a model trained on historical dataset.

The architecture receives historical data from players and converts them into a tensor which is pipelined into the LSTM architecture to output the positional coordinates of the players on the virtual board. This can be made real-time as well; however, that is outside the scope of this research and will be pursued in the following iteration of the publications as there is significant lag in the detection and monitoring if multiple deep learning models are integrated into one main flow.

Chapter 6
Conclusion

The purpose of this research was to devise a novel way to track football players and visualise them in a way that enhanced the viewing experience as well as provided a means to collect data, analyse games and draw useful insights from the footages in real-time. The target users were from regular football enthusiasts and spectators who simply wish to view the games with some augmentation in their viewing experience, and also gaming firms, football clubs and managers, footballers and sporting associations to use this tool to visualise and analyse the games using the tools of AI.

This project was an end-to-end system that used live footages and performed object detection to keep tabs on the players, ball and the referees, and track all their motion throughout the game using BYTE Track. The jersey detection done using EfficientNet-B0 was highly capable of detecting the number on the players' back which could be used to identify the name of the player through a manually entered dictionary of values. Subsequently, the field was accurately and quickly detected and landmarked using a variety of algorithms, morphological operations and edge detection methods to overlay key points that would be used in perspective transformation module to transform the 3D coordinates of the agents to a 2D virtual board. This helps accurately pin-point where the player is positioned even when the camera is moving vertically, horizontally and zooming in or out.

The models are robust and are capable of adjusting to changes in the motion of the cameras and the agents accurately and quickly. Furthermore, the probabilistic estimation of the players allows us to visualise the position of the players even when the player is outside the field of the camera. This will enable the audience to know where the players are even when the camera does not cover their position in the frame. While live audiences can simply look and find the positions of the players, the remote audience watching on television will now be able to experience the same by using this tool. The estimates was done using LSTM which yield less than 0.95 MAE which suggests that the model is very accurate as was observed in the previous chapters.

In the future works, data analytics can be done using the collected data to analyse the games more thoroughly which will give insights into the number of goals, passes and dribbles. This information will subsequently help even make robust and accurate

predictions of which team would win a given match. Moreover, the positional estimate of the players will be made real-time and refined further in the future iterations of the publications by exploring various architecture and reducing computational delays.

Another extension of the project is to perform action recognition to analyse the players' actions such as "pass", "dribble", "taking a shot", "kicking", "heading", etc. Using the positional data from this research and action recognition, we can build a automatic commentary system that can perform natural language processing that could help produce commentary in any languages without the need of a human translator.