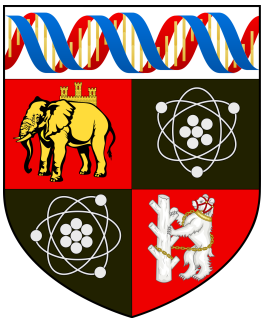
Gameplay Reconstruction using NBA TV footage

Guney Coban

Supervised by Rossella Suma

2023-24



Abstract

The global sports analytics market has grown rapidly over the past two decades. Professional teams across the largest sports leagues in the world have been adopting new technologies in hopes of gaining strategic edges on their opponents. In the National Basketball Association (NBA), data that was once readily accessible to fans has become exclusive to teams and major sports networks, limiting public engagement with the analytical aspects of basketball. This project aimed to bridge this gap by demonstrating the feasibility of using publicly available footage of basketball games to provide fans with advanced insights into the game. While the final system operates with some inconsistencies, it successfully plots player movements for a significant number of individual plays. This achievement highlights the viability of our initial objective and offers a promising foundation for further development on works which enhance fan interaction and analysis in basketball.

Keywords: *Machine Learning, Computer Vision, Deep Learning, Neural Networks, Object Detection, Object Tracking*

Acknowledgements

I would like to thank my dissertation supervisor Professor Rossella Suma for her continued support and advice throughout this project, and Professor Victor Sanchez for his constructive feedback after my presentation.

Contents

1 Motivation

In the world of sports, the use of technology has revolutionised how games are played, analysed, and experienced. The use of advanced analytical tools has become indispensable for teams trying to gain an advantage, with the global sports analytics market projected to grow from $3.78 billion in 2023 to $22.13 billion by 2030[1]. However, particularly in the sport of basketball, this technological advancement has been exclusive to professional teams who benefit from access to sophisticated and proprietary camera systems and sensor-based tracking technologies [2].

This presents a significant gap in the market for the vast majority of basketball enthusiasts. They are left with comparably limited means to engage with the game's analytical aspects. While detailed metrics exist for consumers, they often rely on data sources that are expensive and have complex technological requirements.

1.1 Existing Works

This section delves into some examples of complete analytics solutions across the world's largest sports leagues. By exploring the impact their data has on teams and fans alike, we were able to justify our motivation and evaluate the success of different strategies. This insight helped us avoid past pitfalls and outline key objectives for our project, particularly regarding accessibility for fans.

One case study we explored was the rise of analytics used in American football. During a game of American football, every possession is a result of both teams calling 'plays' - a sequence of actions designed to advance the ball or stop the opponent. The play-by-play structure of the sport means that there must be constant analysis of game situations to make correct calls. This makes American football an ideal example of a sport which can benefit from the adoption of new analytical technologies.

The National Football League (NFL) is the largest professional American football league in the United States and stands as the highest revenue-generating sports league in the world, generating $18.1B in 2023.[3]. Given its prominence and wealth, the NFL has been positioned to leverage cutting-edge technologies to improve the game. A key example is their partnership with Amazon Web Services (AWS), bringing new insights to coaching and player fitness.

Launched in 2017, AWS Next Gen Stats (NGS) employs advanced Radio-Frequency Identification (RFID) technology embedded in players' shoulder pads and the football itself, to track players and the ball in real-time. The system captures player data such as location, speed, distance travelled and acceleration ten times per second, and charts their individual movements. More than two hundred new data points are created on every play of every game [4].

AWS systems use machine learning models on this raw data to derive further metrics such as pass completion likelihood and expected rushing yards. Analysis of such a comprehensive dataset allows coaches to make ad hoc adjustments based on the unfolding dynamics of the game. Beyond the immediate applications during live games, the new data also has profound impacts on player rehabilitation and injury prevention strategies for teams [5], with the NFL commissioner Roger Goodell stating "When we apply next-generation technology to advance player health and safety, everyone wins [6]."

In regards to our project, it was important to analyse the impact this new wave of data has had on the fan experience. The NFL has taken a transparent approach with its data, integrating these advanced analytics into live broadcasts (Figure 1), which has allowed fans watching the games to be exposed to the same level of analysis as coaches on the sideline. This accessibility has deepened casual fans' understanding of the game dynamics and team strategies, which has led to increased fan engagement [7].

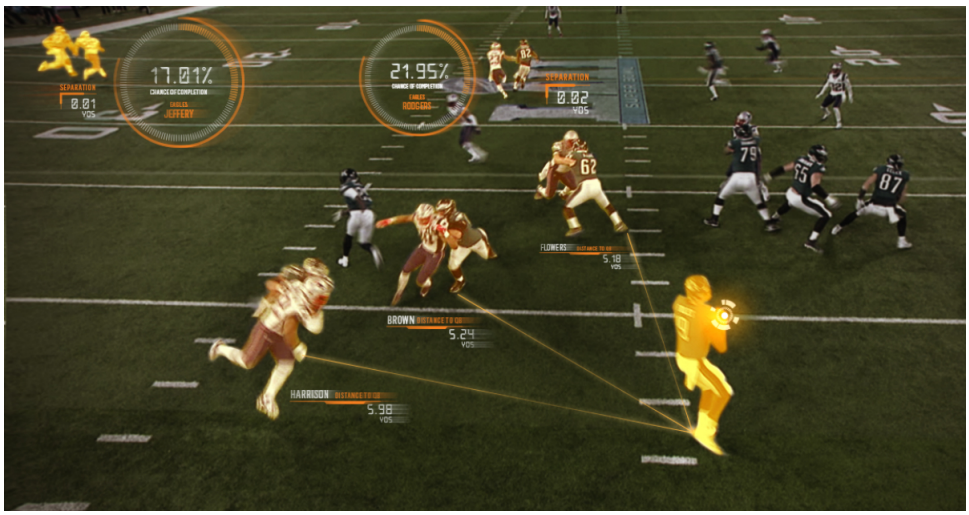


Figure 1: Example of advanced predictions made by AWS Next Gen Stats in real-time

Another large aspect of fan engagement in the NFL is fantasy football, which has become an integral part of the NFL experience, with over 40 million players in the US in 2022 [8]. Studies have shown a positive relationship between fantasy participation numbers and NFL viewership numbers [9] as well as increased team loyalty. [10] [11], making it an important consideration for analysing fan engagement with the sport.

Historically, fantasy football relied on basic statistics like touchdowns, rushing yards, and receptions. However, as it has grown in popularity over the years, fans and analysts have been responsible for the creation of numerous American football efficiency metrics that better explain past football performances and attempt to predict future player production, using them to enhance the fantasy experience. The openness of Next Gen Stats data has been revolutionary for this purpose. With NGS, the NFL provides public access to a vast array of advanced metrics, including player speed, acceleration, route precision, and separation from defenders - data points that were once only available to team insiders. This level of detail allows fantasy players and analysts to evaluate player performance and connect with the game in a way previously unavailable.

Both directly and indirectly, the league's promotion of advanced data to the average fan has encouraged a growth in fan engagement with the sport and enriched the watching experience [12].

A contrasting case study which solidified our motivation for this project is the National Basketball Association's (NBA) handling of big data. The NBA is the third largest league in the world by revenue [13]. It is also a league where the dynamics of the sport have evolved rapidly since its peak in popularity in the 1990s, largely in part due to the role of emerging technologies.

First introduced to the NBA in the 2010-11 season by STATS LLC., SportVU is a system comprised of six cameras each collecting data 25 times per second, following the ball and all players on the court [14]. The data collected by these cameras is used with advanced computer vision techniques to provide real-time player and ball positioning information. STATS uses this data to derive performance metrics for players and teams and also provides the raw data to allow teams to carry out their own analysis. Initially adopted by only ten NBA teams, by the 2013-14 season, every arena in the league had the system in place [15].

While explaining the use of predictive analysis in the NBA, commissioner Adam Silver claimed that the information derived from the technology can be incredibly detailed, going beyond what the human eye can easily notice during a game [16]. For example, a report revealed that a specific player tends to go forward with their left foot 42% of the time and that they exhibit certain patterns before executing particular moves. More generally, teams are now equipped with a way of quantifying aspects of the game that were previously only subject to theoretical analysis, such as how effective a certain player is at setting up their teammates for open shots or how much of an impact another player has closing out on 3 point shooting [17].

One impressive visualisation of the data it generates is that in real-time, the system can display 'phantom' players representing what it believes would be the ideal defensive positioning and movement for each of the players to create the lowest possible percentage shot for their opponent (Figure 2). Today, teams can make such estimations with 10 entire season's worth of data to rely on, and as the system is employed throughout further seasons, this dataset becomes increasingly valuable.

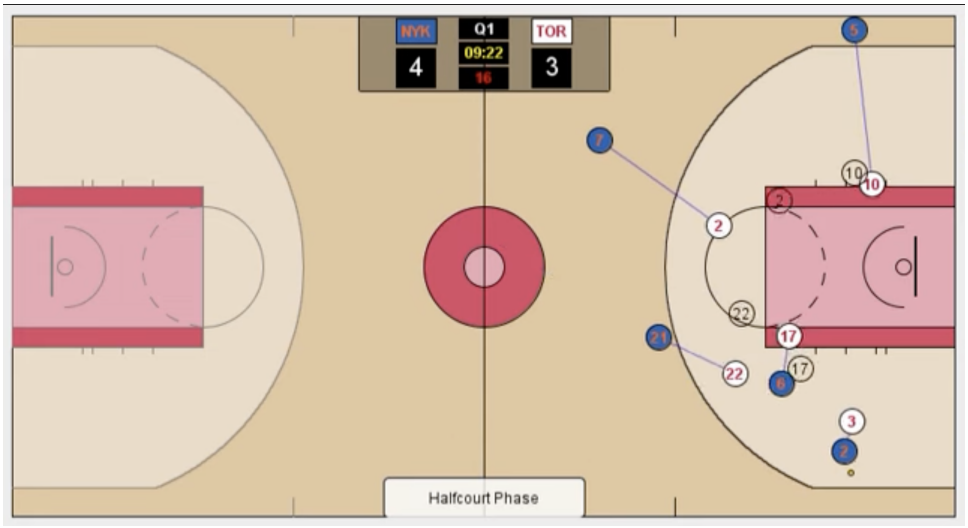


Figure 2: Demonstration of 'phantom' players, shown as blank circles

Utilising this data to understand tendencies and patterns has become invaluable for teams to make defensive strategies and develop game plans against certain teams and players. A useful case study is the Golden State Warriors' adoption of the data for their game strategies. They were one of the first teams to install the technology in their arena and to fully utilise the data provided [18]. Their data-driven approach led to one of the most successful periods by a single team in NBA

history, culminating in three NBA championships in four years (2015, 2017, 2018). Whilst talent is a key factor for any championship-winning team, in the case of the Warriors, it is hard to ignore the impact of strategic innovations derived from analytics. Most notably, their play style involving a significantly increased volume of three-point shot attempts came from an analytical understanding of shot positions and percentages. The success of the Warriors catalysed a league-wide shift in strategy. Other teams began to adopt similar approaches, particularly in regards to their three-point shooting strategies and this is visible through the increase in the ratio of three-pointer shots taken in a game, which has increased by over 13% since the Warriors' first championship in 2014 [19]. This transformation demonstrates the profound impact that data can have not only on a single team but on the sport as a whole.

In 2016, STATS and the NBA reached an agreement to extend SportVU tracking data to media outlets including ESPN, NBA on TNT, and Bleacher Report [20], but also quietly removed public APIs which allowed users to access historical and current data [21]. Since this turning point, fans have been unable to draw their own conclusions or analyse the comprehensive data that is captured at each game, only being exposed to analysis provided by professionals from a small selection of media outlets. This is in stark contrast to the NFL's Next Gen Stats, where data transparency has encouraged fan participation and fostered a richer engagement with the sport. The NBA's approach has inadvertently restricted fan interaction to passive consumption rather than active analysis.

2 Methodology

2.1 Objectives

Learning from the successes of the NFL's approach to data availability for fans, this project exists as a proof of concept that the high level of analysis provided to NBA teams can be replicated at a suitable level for fans by utilising publicly available TV footage.

The primary objective of this project was to develop a system capable of processing this footage to track and map the movement of players over the course of

a play into a mathematical representation. This would form the foundation for automating the process and ultimately creating a comprehensive, open-source dataset of player movements in plays across the course of a season.

By filling this gap in the market for non-professionals, we sought to give fans a platform to understand and compare team strategies, seeing firsthand which offensive plays succeed against specific defensive setups and removing the need for manual film study. It was also aimed at encouraging the development of interactive tools for fans to explore plays, strategies, and player performance in an accessible format.

2.2 Research Methodology

This was a project that heavily incorporated computer vision concepts; a field which has experienced exponential growth in interest and research since 2015 (Figure 3).

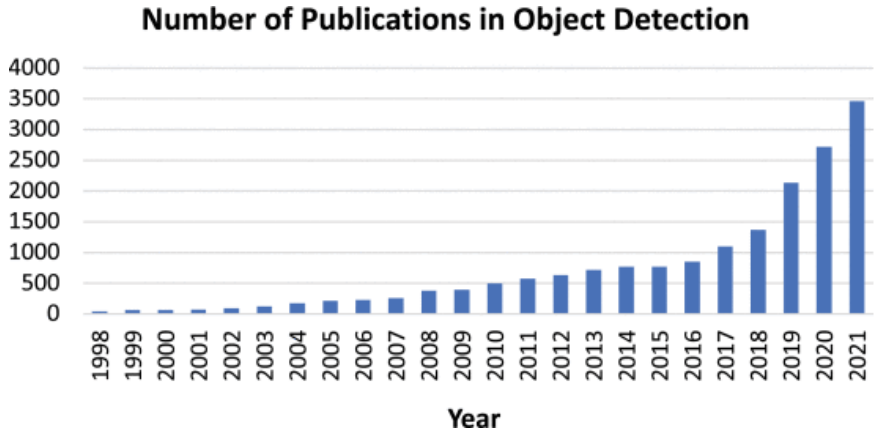


Figure 3: Number of publications in object detection from 1998 to 2021. (Data from Google Scholar advanced search: allintitle: "object detection" or "detecting objects.")

Due to the popularity of the field in recent years, there exists a rich catalogue of up-to-date comparative research papers regarding the performance of object detection and tracking algorithms for varying scenarios. We utilised an experimental literature review as our research methodology to critically analyse existing research, shortlisting ideas and providing insights into the approaches we should be taking. This involved conducting a systematic search to collect relevant studies, establishing criteria for their evaluation, and using these to evaluate a given algorithm's performance and applicability [22]. We assessed each study's methodology and findings to

understand their reliability and context. To confirm our hypotheses on performance for our specific use cases and choose from created shortlists, we then conducted our own personal testing on the selected algorithms and methods. This involved implementing the algorithms in controlled test environments that replicated smaller instances of problems our project faced. Through these practical tests, we were able to directly observe the algorithms' performance, providing a secondary layer of validation to the theoretical research findings.

This approach ensured that our selected algorithms were not only up-to-date and theoretically sound according to current literature but also proven for our specific use cases.

2.3 Development Methodology

Much of the development process involved exploring techniques and areas that were previously unfamiliar to us. This uncertainty in the development process was not just about the feasibility of certain technical approaches but also about the time and resources that would have been required to achieve our goals. As such, the project demanded a management approach that accommodated learning and improvement in the project timelines.

A framework that fit these purposes and allowed for the iterative development of individual components was Personal Extreme Programming (PXP) with MoSCoW Prioritisation. PXP is an adaptation of the popular Extreme Programming (XP) methodology tailored for individual developers. PXP utilises sprints to enhance responsiveness to changing requirements through frequent iterations. Its core principles involve test-driven development, where tests are written before the actual code, ensuring each new feature is properly verified; continuous integration and frequent releases which help with identifying and addressing integration issues early; and implementing the simplest possible working solution to speed up delivery while ensuring functionality. [23].

For our project, we utilised this sprint-based development strategy to work on and complete requirements systematically. At the beginning of each sprint, we reiterated and verified the requirements of the current component, followed by a detailed planning phase where we used the MoSCoW ranking of the requirements

to outline the sprint's development goals. The MoSCoW method is a prioritisation technique used to categorise requirements into four groups: Must have, Should have, Could have, and Won't have, referring to their relative importance and urgency in the project timeline. This aims to ensure that essential features are completed first, while less critical enhancements are appropriately scheduled or deferred [24].

Over the course of a sprint, we planned to iteratively improve the component, completing the sprint's chosen requirements step by step and only once the previous implementation was working as expected. Each iteration had its own planning phases, followed by a development phase and a testing phase as shown in Figure 4.

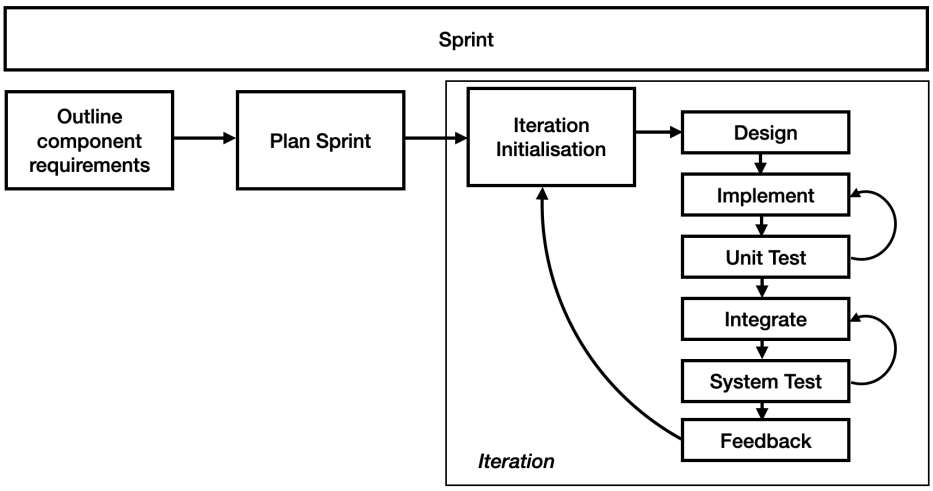


Figure 4: High-level overview of a sprint.

The methodology was laid out such that one requirement could take multiple iterations until it passed testing, at which point we integrated the new code into the existing system and participated in system testing to ensure the integration worked. This method ensured that the goals of the project would be met and also prevented the overdevelopment of non-crucial features. Upon completion of system testing, we conducted a retrospective to review our processes, discuss what worked well and identify areas for improvement. This reflection allowed us to learn from our experiences and apply those lessons to the subsequent sprints, enhancing our methodology as the project advanced.

Throughout this development process, a backlog of remaining requirements was maintained, using their initial MoSCoW ratings to keep an order of prioritisation. Our goal was to initially solidify the 'Must Have' requirements to establish a foun-

dation for the project, before partaking in subsequent sprints to focus on addressing the 'Should Have' and 'Could Have' categories. This process ensured that we would be able to complete the essential features necessary to fulfil our project objective while adhering to our deadlines.

2.4 Legal, Social, Ethical and Professional Issues

Given the stringent copyright restrictions on NBA footage [25], this project strictly utilised publicly available footage for analysis purposes. No copyright infringement is intended or committed, as the footage is never reproduced or redistributed.

There were no social, ethical or professional issues to consider as no personal data was collected.

3 Project Planning

3.1 System Requirements

We outlined the following requirements to guide the evaluation of our objective, shown in Table 1.

	Requirements	MoSCoW
R1	**Detection and Tracking**	
1.1	Accurately detect players in any given frame.	Must Have
1.2	Continuously track individual players throughout the game footage.	Must Have
1.3	Extract players' movement data from the footage.	Must Have
1.4	Accurately detect and track basketball throughout the game footage.	Should Have
R2	**Court Mapping**	

2.1	Calculate mathematical mapping which allows perspective transform of player positions onto 2D court representation for a given frame.	Must Have
2.2	Maintain accurate mapping throughout the course of footage.	Must Have
R3	**Data Visualisation**	
3.1	Display positions of players on a 2D court from a bird's eye view for a given frame.	Must Have
3.2	Display movement paths of individual players throughout the game footage.	Must Have
3.3	Display ideal positioning of defensive players.	Won't Have
R4	**Data Analysis**	
4.1	Derive statistics regarding player movement and positioning during plays.	Should Have
4.2	Detect key game events like shot attempts, passes, rebounds, blocks, and steals.	Could Have

Table 1: System Requirements

3.2 System Design

Given the complexity of the project, it was essential to abstract the problem and systematically organise the work to ensure smooth development. We defined four main components; player detection, court mapping, player tracking and data analysis. Whilst the visualisation of the data was also an important requirement, our approach was to implement plots of the currently available data within each component, to aid our testing and validation process during their respective iterations.

It was also important to define dependencies between these components to help

guide and explain our development process. Both the player detection and court mapping components were able to be developed independently. In contrast, to provide an effective tracking model, we had to build on the work of the previous components. It was apparent that accurate player detection was a prerequisite as we required the initial identification of players before being able to track their movement. Precise court mapping was also important as it could allow for more accurate tracking predictions based on the real movement of players on the court. Finally, the data analysis module required the tracking to be working at a high level to be able to draw meaningful insights. A visualisation of this dependency structure is shown in 5 and was used to guide the creation of our project schedule.

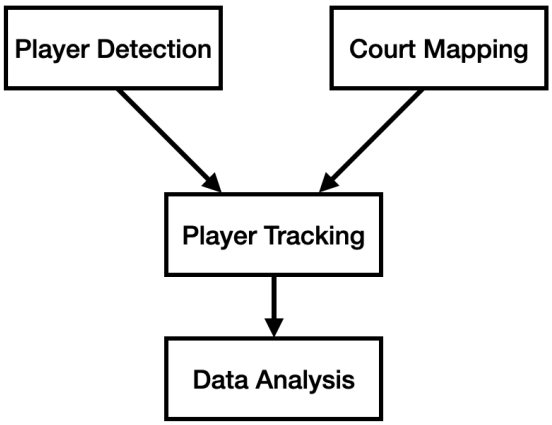


Figure 5: Dependency graph of project components.

3.3 Project Schedule

We decided to organise the work into four primary sprints, initially focusing on the 'Must Have' requirements of our project which involve the player detection, court mapping, and player tracking components, as well as a final sprint to ensure their successful integration within the system. This choice was to allow us to focus our time on research and providing the best possible working solution without rushing to add extra features. We employed a Gantt chart to visualise the project timeline and provide a structure to our development process which considers the dependencies between components (Appendix A).

Having defined our project requirements and timeline, we were able to relate the concepts introduced in Section 2.3 to our actual components. For each sprint,

the objective was determined by selecting a requirement to complete based on its priority and dependencies as outlined in our Gantt chart. In the planning phase, we broke down these broader requirements into smaller components, where each component was to be iteratively improved as defined in our development methodology. At this stage, we implemented a Kanban board for the sprint to track the progress of these components and manage our workflow throughout the sprint. An example of this usage is shown in Figure 6.

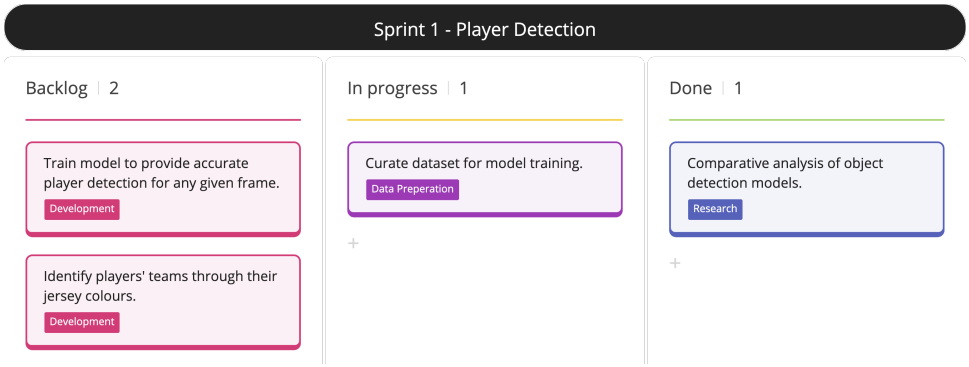


Figure 6: State of the Kanban board designed during the planning phase, captured midway through the first sprint.

4 Development

4.1 Technology Stack

At the start of the development phase, we had to decide on a technology stack which provided the required tools for image and video manipulation as well as support for advanced computer vision methods. Another consideration was the principles of Personal Extreme Programming (PXP) which focus on using our strengths to create a setting that supports quick development cycles [26].

Python stood out as a leading choice for several reasons. Due to our existing expertise in Python development, we could adhere more closely to the PXP methodology. More importantly, Python provides industry-leading frameworks and libraries for computer vision and image manipulation. The usage of the two leading libraries, PyTorch and TensorFlow, accounts for around 75% of deep learning projects on Github [27]. Given the consistent increase in PyTorch's adoption and the gradual decline in TensorFlow's usage, we chose PyTorch for its growing rele-

vance and broader model support. This decision was guided by our project's usage of cutting-edge technologies and ensured adaptability for future improvement.

For its general utility in the handling of images and videos, we integrated OpenCV during the development of our components. It allowed us to process videos frame by frame, and we used its ability to facilitate direct annotation on images, enabling us to label and visualise our results. OpenCV also provided functions for manually implementing computer vision techniques such as initialising Kalman filters. Additionally, we employed NumPy to manage and perform computations on our dataset of player positions and related data points. NumPy's efficiency in handling large arrays and matrices made it the preferred choice for implementing mathematical concepts that emerged from our research.

Furthermore, Python's active machine-learning community proved invaluable for addressing and resolving the challenges we encountered during development. As Python is widely used in machine learning, many of the issues we faced had already been experienced and solved by others. This meant we could quickly find solutions and advice through community forums, GitHub discussions, and extensive documentation.

Besides programming, the project required the creation of a custom dataset tailored to our needs. To obtain footage from basketball games we utilised the-highlow.io, a resource that enables mass downloads of individual play footage with customisable filters such as only made shots, specific player match-ups, or actions involving particular teams. For the process of using this footage to create a dataset on which to train our model, we chose Roboflow due to its comprehensive suite of tools which allowed for the annotation of large image sets. Roboflow also provided seamless integration with several object detection models, which was a consideration in improving our workflow efficiency [28].

Together, these technologies provided a comprehensive toolkit which allowed us to effectively complete all aspects of our project.

4.2 Player Detection

4.2.1 Research

The field of computer vision, particularly object detection, has undergone a rapid transformation in recent years. Since 2014, traditional detection algorithms have been replaced by deep-learning based detectors, with eleven new milestone algorithms being discovered in the six years following this change [29]. Deep learning models utilise multiple layers of neural networks which allow them to learn representations of data with multiple levels of abstraction [30]. These approaches can be categorised by the number of passes through the neural network an image requires before a prediction.

One-stage detectors are designed for speed and efficiency. They directly predict object bounding boxes and class probabilities in a single pass through the network, making them well-suited for real-time applications. Comparatively, two-stage detectors make initial region proposals – areas of the image that potentially contain objects, then, on a second pass through the neural network, classify the objects within these regions and refine their bounding boxes. Theoretically, this approach leads to more accurate detections, with a loss of speed and efficiency.

Given the requirements of the project, it was crucial to stay updated with the latest developments and carefully select an appropriate model. As explored in the related works, having real-time data for broadcasts can help deepen fans' understanding of the game, making a fast model valuable. However, speed without accuracy is not useful. Reliable player detection is critical to providing the meaningful insights needed to engage fans. Therefore, we needed a model that is both quick and precise.

There are two main factors used to differentiate chosen models on their performance. Their accuracy on a given dataset is measured with mean Average Precision (mAP), where high precision relates to a low false positive rate regarding how accurate the predicted classes for each bounding box were. The other key measure relates to a model's speed and is measured in Frames Per Second (FPS). Before we could begin our comparative analysis, we researched developments in each category of algorithm to compile a shortlist of the most suitable methods.

For two-stage detectors, the Region-Based Convolutional Neural Network (R-

CNN) was a groundbreaking method that applied Convolutional Neural Networks (CNNs) to bottom-up region proposals to localise and segment objects. It involved initially generating region proposals which estimate object locations then warping each proposed region to a fixed size and feeding them into a CNN to extract features. Finally, each feature vector is fed into a set of class-specific linear Support Vector Machines (SVMs) to classify the object within the region [31]. While effective, this method was computationally expensive primarily because it processes each region proposal independently through the CNN, leading to significant redundancy and high computational cost. Since its inception, it has been repeatedly improved. Firstly, Fast R-CNN improved upon R-CNN by processing the entire image only once and extracting features using a shared convolutional network, rather than handling each region proposal separately, achieving 213 times faster performance than R-CNN in testing on a commonly used object detection dataset (PASCAL VOC 2012) [32]. Faster R-CNN further improved on this by introducing a Region Proposal Network (RPN) that uses the same shared features to simultaneously predict object boundaries and objectness scores, providing a more efficient model while maintaining the same accuracy rates [33]. Considering its built-in support in PyTorch, which ensured compatibility with our technology stack, Faster R-CNN presented a compelling option that was considered for our project.

One-stage detectors have seen more diverse development compared to the two-stage approach, where most improvements have been on iterations of the R-CNN family. In contrast, popular one-stage detectors such as You Only Look Once (YOLO), Single Shot Detector (SSD) and RetinaNet have each introduced unique architectures and methods for achieving efficient and effective object detection. YOLO directly predicts bounding boxes and class probabilities in one evaluation. It divides the image into a grid and predicts bounding boxes and probabilities for each grid cell simultaneously [34]. SSD uses a fixed set of bounding boxes across different scales and aspect ratios on the image. It then simultaneously predicts the presence of objects in these boxes and adjusts them to better match the object's shape [35]. RetinaNet processes images at various scales using a technique called a feature pyramid network, and uses a method called focal loss which helps the model pay more attention to harder-to-classify examples and less to those it already classifies well, enhancing its overall performance on challenging cases. [36]. Each of these

models offer distinctive benefits, making them suitable for various applications that require fast and accurate object detection without the computational overhead of generating region proposals.

Hence, a comparative analysis of Faster R-CNN, YOLO, SSD and Retina-Net was necessary. This comparison aimed to find a suitable model which provided acceptable levels of accuracy whilst being sufficiently fast for handling high-frame-rate videos. To guide this comparison, several articles that analyse these models in different contexts and scenarios were considered. [37]

In their 2020 study, Kim et al. compare Faster-RCNN, YOLOv4, and SSD for Real-Time Vehicle Type Recognition[38]. As they conclude, YOLOv4 outperforms other methods significantly with a 93% accuracy in recognising the vehicle model. SSD has the the least accurate results but achieves the highest FPS, but we had to consider that the study used a lighter version of the SSD model, hence these results cannot be treated as representative. Surprisingly, YOLO outperforms the 2-stage detector in accuracy.

In a similar study in 2021, Tan et al. compared the one-stage detectors Retina-Net, SSD, and YOLOv3 to identify medical pills in real time. As they conclude, YOLOv3 performs at a significantly quicker detection speed, 3x higher FPS than Retina-Net and 2x higher than SSD, whilst having a slightly lower mAP. This is in contrast to the previous set of results, however, when we observed these specific values in this research, we found that the precision values were very similar. It was noted that this version of YOLO was weaker compared to its fourth iteration which the previous study utilised. Furthermore, as they explain, the YOLOv3 model outperformed the others when tasked with 'harder sample detection', and concluded that the model was the most suitable for deployment in hospital equipment [39].

A final study analysed the detection algorithms discussed, applying various constraints to assist users in choosing an algorithm best suited to their needs. This study compared each algorithm individually, concluding that the R-CNN family of detectors are too slow for real-time applications, whereas SSD and YOLOv3 offer comparable levels of speed and precision. It also highlighted that YOLOv2 and YOLOv1 produced significantly slower and less accurate detections, highlighting the continuous improvements that occur within the YOLO series.[40]

Thus, when evaluating the various models under consideration, it was important

to note this continuous evolution regarding the YOLO models, with YOLOv8 as the latest iteration [41]. The results discussed utilise previous versions of YOLO, which have since been significantly improved in both aspects of performance. Whilst there is a lack of research papers studying the performance of the latest model, the developers provide evidence that it is both faster and more accurate than previous iterations [42]. This ongoing development added another compelling reason for its selection. YOLO's adaptability and consistent improvements simplify future upgrades to the project. For instance, if a more efficient version of YOLO is released, it is considerably easier to retrain the new model using the same parameters, as opposed to starting anew with a completely different algorithm. This ensured that the project could remain cutting-edge and integrate advancements to maintain its effectiveness over time.

Consequently, we decided to proceed with YOLOv8 as the model to train for player detection.

4.2.2 Development

Before training our YOLOv8 model, we had to obtain a high-quality dataset which covers a range of situations that may arise in frames from a basketball game. For individual players, these include large levels of occlusion and motion blur, whilst other factors include the diversity in court designs and team jerseys used in different games. This was an important step as studies have demonstrated that using limited datasets to train advanced computer vision models can significantly impair their accuracy [43].

Our strategy for obtaining a representative dataset involved individually annotating frames from various NBA games across the previous three seasons, dating back to the 2020-21 season. By not relying on existing datasets, we ensured that the bounding box annotations were consistent with the requirements of the project.

For this phase of our project, we used thehighlow.io to download a random sample of play footage from the last three years. From the downloaded footage, we took image snapshots from these plays to create a representative image set. For the annotation of this image set, we utilised the features provided by Roboflow.

Within each frame, the players, the referees and the hoop were annotated. Referees are consistently present on the court and interact with players but should

not be confused with them, thus their inclusion helps to minimise misclassification. The inclusion of the hoop is for future works, which may include identifying key game events such as scoring. The annotated dataset consisted of 250 images from 89 different games and included footage containing all 30 teams. It contained 33 frames of player interactions with the referees, substitute players and fans near the court to allow the model to contextually learn their differences.

The images were then divided into training and validation sets. We allocated 87% of the images to the training set and 8% to the validation set, which was used to evaluate the model's performance during the training process and tune its hyper-parameters. This split allowed the model to learn from a broad range of data and have its performance fine-tuned without overfitting. A separate test set with the remaining 5% of images was used to evaluate the model after training, ensuring it performs well on unseen data and across different scenarios.

To further enhance the training dataset's size and robustness, we applied image augmentation techniques. These were applied to allow the model to encounter a wider array of variations of the same images during training, to help it learn more general features rather than memorising specific image details. By making the model less likely to overfit the training data, these techniques have proven to increase mAP scores more effectively than simply expanding the model size, which also adds to training and inference costs [44]. A 2021 study by Poojary et al. demonstrated that using an augmented dataset can improve model performance by 9% over a baseline model without augmentation [45].

To capitalise on this advantage, through the tools offered in Roboflow, we applied several augmentations to the training set, leaving us with 639 images to train the model on. The dataset was then exported into a YOLOv8 PyTorch-friendly format, which created a folder containing the image and annotation files in the required format, as well as configuration files that specify class names and other settings needed for training.

Training deep learning models, particularly those with complex architectures, is heavily resource-intensive. As a result, one of the most common issues during such training jobs is running out of GPU memory, leading to failures in the training process [46]. To address this, we utilised Google Colab, which offers access to Tesla T4 GPUs, which come with 16GB of memory. This allowed us to execute our

resource-intensive Python code through the browser, with no strain on our personal hardware. Another of YOLO's notable advantages is its simplicity and ease of implementation within coding environments, allowing us to train the model on our dataset using only the dataset as an input.

The YOLOv8 series comprises five distinct models, each designed to balance the trade-offs between detection speed and accuracy as shown in Figure 7. The larger models contain more parameters and provide higher mAP results for the same datasets but are significantly slower.

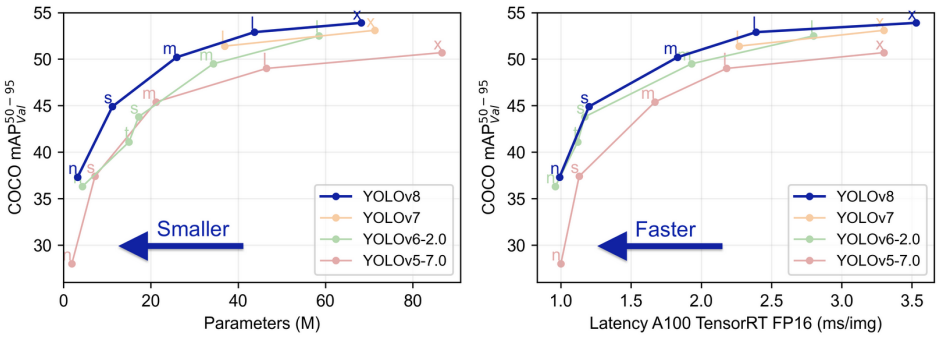


Figure 7: YOLOv8 model comparisons [42].

To justify the selection of one of these model variants for our use case, we trained various configurations of the YOLOv8 model. Then, by comparing their training results on the validation set, we observed whether each model was suitable and analysed their performance trends to determine the optimal selection to balance accuracy and processing speed.

For our testing, we chose to compare the YOLOv8s and YOLOv8l versions as they represent the two ends of the performance spectrum. To ensure they were both trained to similar performance levels, we trained the smaller model for four times as many epochs as the larger model.

To compare their speed, we ran the trained detection model on every frame of the five separate plays and timed how long it took each model to provide detections for the entirety of the footage. We did this five times each time to ensure representative values for both models. We discovered that YOLOv8s operates at around 2.5 times the speed of the larger model, shown by our results in Table 2. This is in line with the values provided by the developers of the algorithm.

	Average time taken to complete detection for entire play (s)				
	Play 1	Play 2	Play 3	Play 4	Play 5
YOLOv8s	59.4	9.4	14.9	112.1	33.4
YOLOv8l	149.5	23.2	34.9	266.4	83.3
Smaller model speed advantage	2.5x	2.5x	2.3x	2.4x	2.5x

Table 2: Average time taken by each model to complete detection for an entire play.

To assess the accuracy of each model, we compared their accuracy results on the validation set using the mAP50-95 and mAP50 results. mAP50 is a metric which assesses whether the predicted bounding box overlaps at least 50% with the ground truth box. mAP50-95 offers a more detailed assessment by carrying this process out for overlap matches from 50 to 95% and finding the average precision values across these. The results for the respective models are shown in Figures 8 and 9.

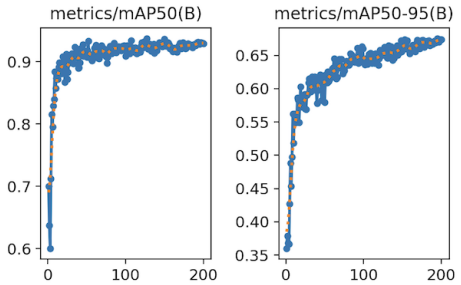


Figure 8: YOLOv8s model accuracy on the validation set.

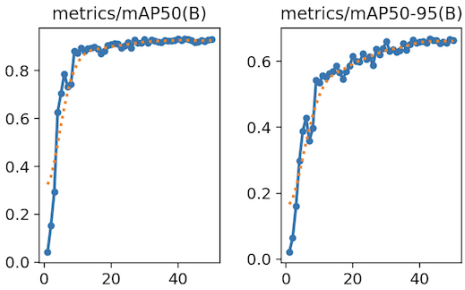


Figure 9: YOLOv8l model accuracy on the validation set.

Surprisingly, we discovered that towards the end of their training processes, both models provided highly accurate results, with both their mAP50 results converging at around the 0.95 mark. Similarly, the mAP50-95 value for the YOLOv8l model appears to converge around the 0.65 mark, whereas the same value in the smaller model appears to continue increasing past 0.65. These results were not consistent

with the expected relative accuracy based on the strengths of the models. We concluded that as our dataset does not require capturing extremely complex patterns, the performance benefit of using a larger model with more parameters must be very marginal.

Therefore, this made our model selection a simple choice. YOLOv8s proved to be a significantly faster model which also provided sufficiently accurate detections. Our next stage of validation involved fine-tuning the number of epochs the model was trained for. While this is a critical step, our decision was particularly influenced by the loss metrics observed during model training (Figure 10). These metrics are used to quantify the difference between the predicted values by the model and the ground truth for both the training set and the validation. They showed that while the model kept improving on the test data, the improvement on the validation dataset was stagnant past one hundred epochs of training, which led us to consider the possibility that the model was overfitting on our training data.

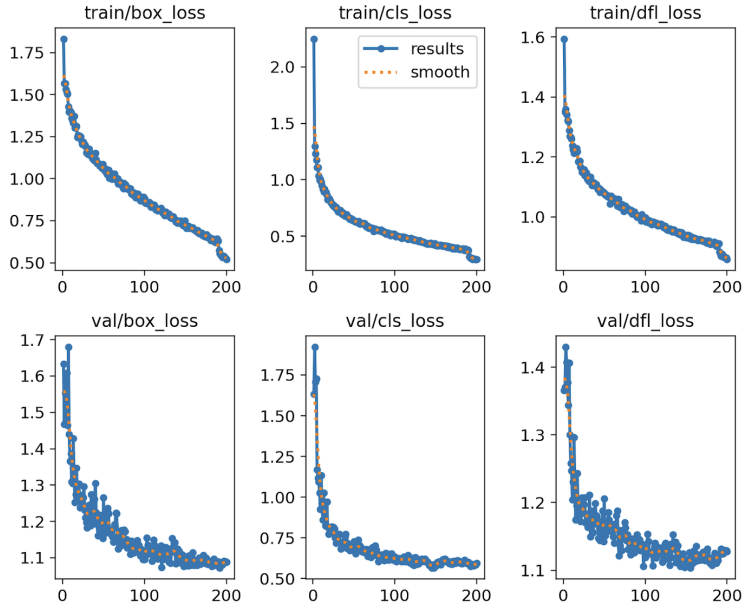


Figure 10: Loss metrics for trained YOLOv8s model across training and validation set.

A metric which assisted our verification of these claims was the normalised confusion matrix produced when testing on the validation set. It provides a visual representation of the proportion of true and false predictions for each class relative to the total instances of that class. This matrix is particularly helpful because it

not only displays the number of correct and incorrect predictions but also breaks down the types of errors made, such as how often a player was mistakenly identified as a referee. A key sign of overfitting for our case would be an increased number of prediction errors confusing the background for a player. This is because in our training set, we use many examples of occluded players, where there is only slight visibility of one player behind another. Therefore, if the model were to overfit on this data, it could be led to detect more than one player in situations where there are none. The respective confusion matrices for the model trained for one hundred and two hundred epochs are shown in Figures 11 and 12.

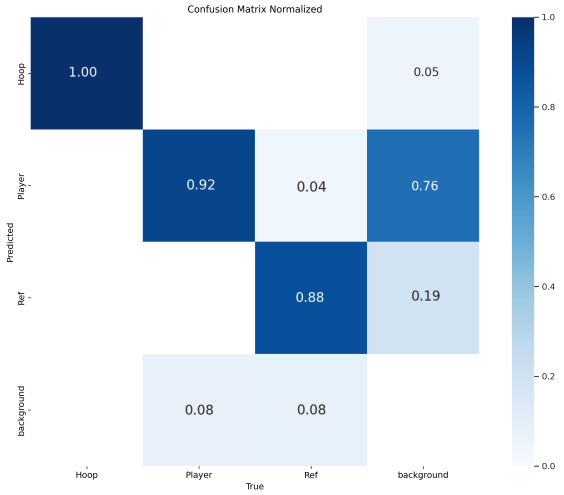


Figure 11: Normalised Confusion Matrix for model trained for one hundred epochs.

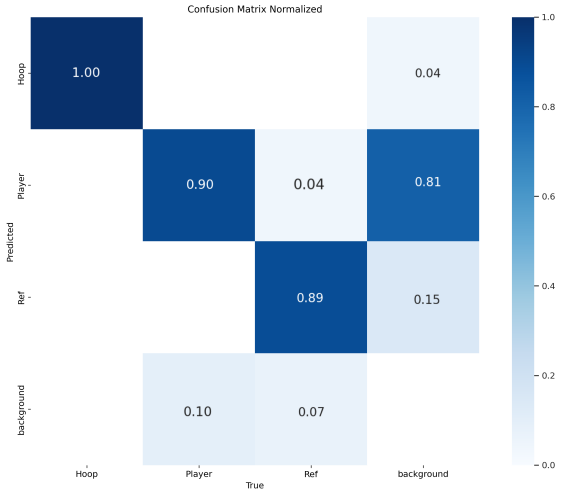


Figure 12: Normalised Confusion Matrix for model trained for two hundred epochs.

As seen, the model trained for longer had a higher rate of false detections, where it mistakenly identified background elements as players, aligning with our prediction that this model may be overfitting on the training data. Additionally, this model showed a decreased effectiveness in correctly identifying actual players. Based on these observations, we decided to choose the YOLOv8s model trained for one hundred epochs as our primary player detection model, as it provides more reliable detections.

The tuning for the model involved deciding the confidence threshold for accepting detection results. We analysed the Precision-Recall (P-R) graphs of the model to help us come to a conclusion. Precision measures the accuracy of the positive predictions made by a model and recall is the proportion of true positives detected from the total actual positives. The P-R graphs demonstrate how precision and recall vary with changes in the confidence threshold (Figure 13).

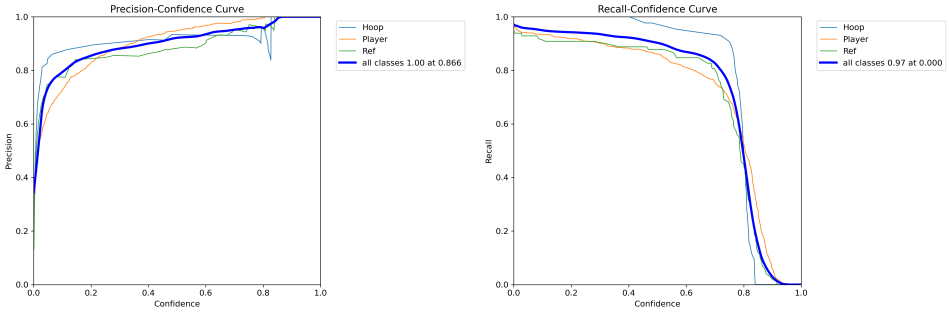


Figure 13: Precision and Recall curves for chosen model.

The graphs indicated that setting a confidence threshold of 0.5 would balance the precision and recall effectively, ensuring reliable detection while maintaining a manageable number of false positives. Therefore, we opted to accept detections only with a confidence level greater than 0.5.

The final step in completing the player detection component was a team identification system that determines each player's team from their bounding box. Once again, this was affected by the issue of occlusion and the presence of repeated background colours where taking the average colour of the bounding box would not yield precise results. To simplify these issues we categorised them into the reasons why they occurred. A bounding box contains both player data but also background data due to its rectangular nature. Hence it was important to ignore irrelevant colours.

The chosen process involved defining the two jersey colours for a given game and adjusting for brightness by including a variance of 20% to accommodate different lighting conditions and shadow effects. For each player's bounding box detected by the system, we applied separate masks for the colour ranges of each team's jersey. This isolated the colours that are indicative of each team's uniform within the bounding box. We then calculated the ratio of pixels that fall within each team's jersey colour range to the total number of pixels in the bounding box, equivalent to the number of non-zero pixels in the mask. The team whose colour range constitutes a higher ratio of the bounding box's pixels is identified as the player's team.

This method ensured team classification was resilient to common issues such as varying game-day lighting and reflections from shiny court surfaces. For the issue of occlusion within a bounding box, we handled this through the feature tracking methods explored in Section 4.4.2.

4.3 Court Mapping

4.3.1 Research

The process of court mapping in our project involved translating the position of players observed in 3D footage into a fixed 2D court representation. The key theoretical foundation of our approach was derived from the work outlined in "Multiple View Geometry in Computer Vision" by Hartley and Zisserman. This paper discusses the mathematical foundations necessary for understanding how multiple camera views can be used to reconstruct a 3D scene onto a 2D surface.

Any two images of the same planar surface in space are related by a homography, which is a transformation that maps points from one representation of the plane to the other. Mathematically, a homography is a 3x3 matrix transformation that operates on homogeneous coordinates. In homogeneous coordinates, a point in two-dimensional space (x,y) is represented by three coordinates, usually (x,y,1). This representation simplifies matrix operations for translations, rotations, and other transformations. The process to calculate the homography matrix H involves solving a set of equations derived from pairs of corresponding points in the two images. Specifically, if we have a point x_1 in the first image and a corresponding point x_2 in the second image, the homography H must satisfy equation (1), where $\approx$ denotes

equality up to a scalar multiplication [47].

$$x_2 \approx H \cdot x_1 \tag{1}$$

To solve for H, we typically need at least four point correspondences between two views, under the assumption that no three points are collinear. These points provide the constraints necessary to solve the linear system that defines H.

In our project, the basketball court serves as the planar surface that we aim to map from 3D broadcast footage to a 2D representation. Calculating H for any given frame allows the position of any detected players to be mapped onto the diagram. For an implementation of this in Python, OpenCV provides a convenient function called *findHomography()*, which automates the process of calculating the matrix using the Direct Linear Transformation (DLT) method given a set of point correspondences [48].

Given the static nature of the 2D court diagram, we can easily select four constant points that define key features of the court. However, accurately matching these points with their counterparts in the frames of the 3D game footage presents several challenges. The diversity of lighting conditions across different games makes it difficult to consistently detect and match features across different games, the occlusion of key court markers by players prevents reliably identifying the necessary points in the footage, and moving camera angles repeatedly change the sections of the court which are visible at any time.

Thus, it was crucial to implement an accurate feature-matching technique that could adapt to these conditions and accurately identify corresponding points between the two representations.

We conducted research into feature-matching methods to find reliable algorithms that could consistently and accurately identify these points under the variable conditions typical of broadcast sports footage. There are typically three commonly used techniques for this task: Scale-Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF) and Oriented FAST and Rotated BRIEF (ORB), all of which have implementations for Python through the OpenCV library. As concluded in several comparative papers, ORB is consistently the fastest of the three, making it suitable for real-time applications while SIFT performs the best in most

scenarios and is the least prone to distortion [49] [50].

Alongside these feature-based methods, we also explored the Hough Line Transform for its direct applicability to detecting straight lines, for another approach to feature extraction: identifying court lines to derive the coordinates of its corners. The Hough Line Transform is a widely used technique in computer vision for detecting straight lines. This method transforms points in the image space into polar coordinates, represented by the line parameters angle (θ) and distance (ρ) from the origin. In an image, each point casts votes for every line passing through it, and a high concentration of these votes in the parameter space signals the presence of a line in the image.

Research in the area of the Hough Transform has led to adaptations that improve its efficiency and applicability to different scenarios. The most notable is the Probabilistic Hough Transform, which improves computation time by randomly sampling points from the image space and estimating the existence of a line segment, rather than the entire line [51]. Our approach utilised intersections of court lines, therefore, detecting line segments was as valuable as identifying the full length of lines, as we could still calculate the equation of these lines, which is what we used to find intersections.

The Hough Line Transform only works effectively when applied to the output of an edge-detected image [52]. Hence we carried out comparative research on existing edge detection algorithms. The two commonly used methods for this are the Canny and Sobel algorithms. Canny is designed to provide precise, thin, and well-defined edges and involves several steps to filter out noise and accurately trace the edge paths, making it more complex than Sobel. Sobel is simpler and quicker, focusing on highlighting general edge directions and strengths but without the advanced processing of edge connectivity and thresholding. Multiple comparative articles of these methods exist for a variety of use cases, all of which conclude that the Canny edge detection algorithm provides significantly more accurate results at the cost of more computational power [53] [54] [55] [56]. Due to the requirements of our project, it was imperative to have accurate feature detection, hence we opted to only consider Canny for our use case.

4.3.2 Development

We initially worked to verify the conclusions reached in our research by testing each advanced feature-matching technique. We then analysed their effectiveness in matching features from the footage to a 2D image of the same court.

This yielded surprisingly poor results for all three suggested methods. We found that there were several reasons for this. Basketball courts have repetitive patterns and textures including the wooden floorboards and court lines, which confuse the feature detectors and descriptors. These algorithms mistakenly match features that occur repeatedly across the court, leading to inaccuracies in establishing correspondences between the footage and the 2D court diagram (Figure 14).

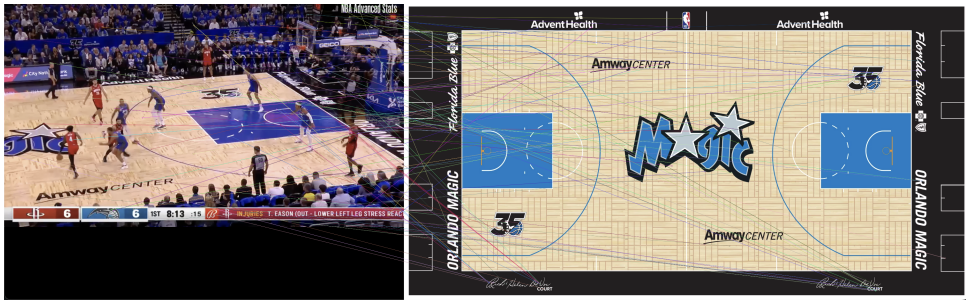


Figure 14: Example of SIFT matching footage to 2D image of court.

Furthermore, variations in lighting conditions between the indoor arena settings and how they are captured in different broadcasts affect the appearance of the court. Shadows cast by players and changes in arena lighting altered the visual texture and contrast of the court surface, complicating the feature detection and matching. Therefore we concluded that this was the wrong process for our case. We were tasked with finding a new approach considering the pitfalls of the existing feature-matching algorithms. The most crucial aspect of this process was determining common points in all available TV footage that are representative of the court's perspective without being affected by environmental conditions. We discovered that a visible marker which was consistent between all plays was some of the court lines, namely the baseline, the far sideline and the perimeter of the key. As demonstrated in Figures 15 and 16, the intersections of these lines can provide us with the same four non-colinear points on the court for each play in the same direction, even if all four points are not visible in the frame.

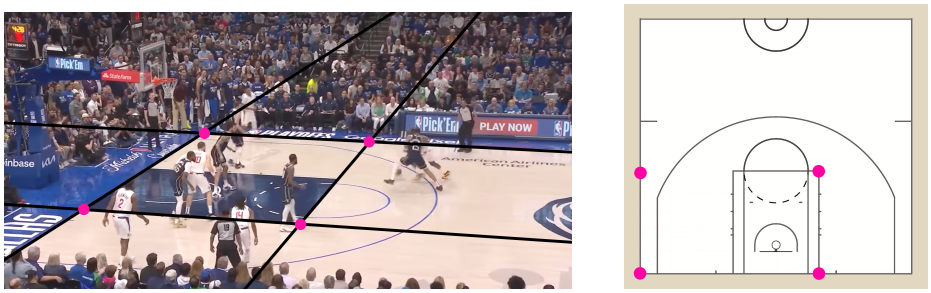


Figure 15: Left side of the court represented as intersections of lines.

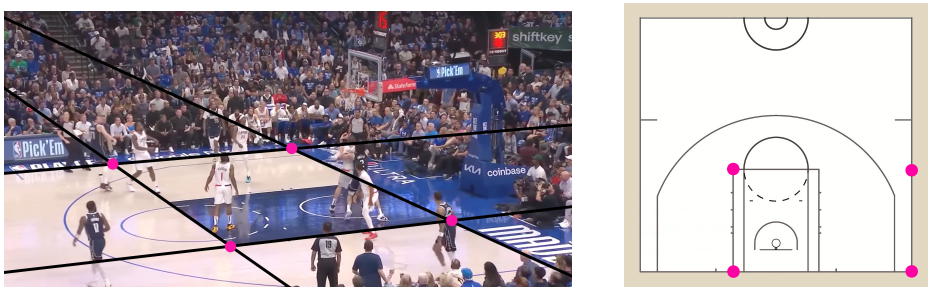


Figure 16: Right side of the court represented as intersections of lines.

Our method for obtaining these points for an individual frame consisted of the following steps. Firstly, we had to pre-process the image for accurate edge detection [57]. We initially converted the image to Grayscale to reduce its complexity, then applied a Gaussian Blur to reduce noise. Thresholding is then applied to simplify the image to a binary format, which helps to eliminate irrelevant details. Finally, we used a close morph to connect nearby objects or edges that are slightly apart, ensuring lines do not go undetected due to slight occlusion. Next, we were able to perform edge detection on the processed image using the Canny algorithm. An example of this process is shown in Figure 17.

Implementing these filters and edge detection was streamlined by the comprehensive image manipulation features available in OpenCV. However, the challenge arose in ensuring that our edge detection was robust across various scenarios encountered in different game footage. This required extensive parameter tuning for each preprocessing step, including the Gaussian Blur's kernel size and the thresholds in the Canny algorithm, to optimise the accuracy and reliability of our edge detection under varying conditions.

Once we were able to obtain consistent results through the Canny edge detection,

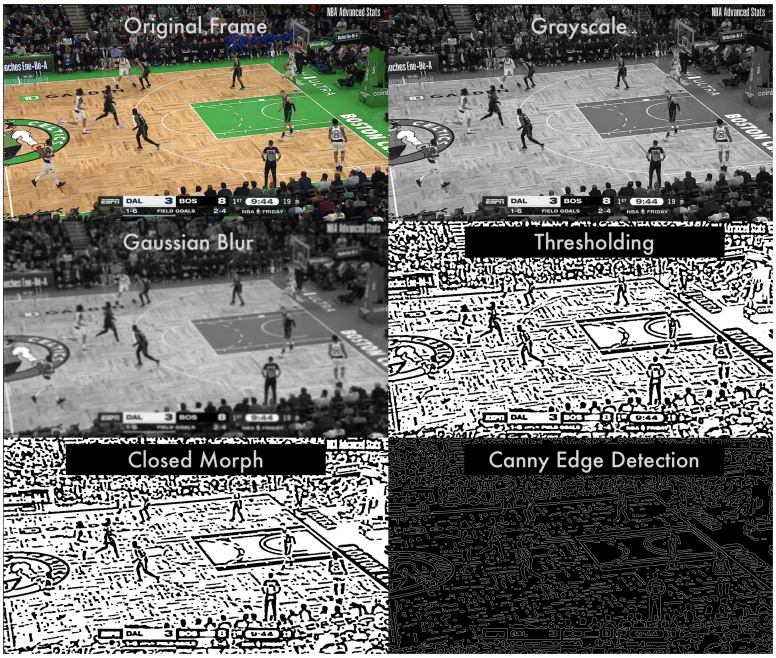


Figure 17: Process of edge detection on a frame.

we implemented a Probabilistic Hough Line Transform on the image, to identify the court lines. OpenCV provides the *HoughLinesP()* function, which we utilised. Due to the inherent noise present in the scene, we had to ensure the parameters were tuned to only consider long enough lines and also lines without significant gaps. Once we obtained a consistent level of accuracy, we used the x_1, x_2, y_1, y_2 values of each line provided by the function to derive each of their gradients and y-intercepts.

We discovered that due to the design of many courts and the thickness of court lines, the process of edge detection followed by line detection led to individual court lines being identified as multiple. To overcome this issue, we grouped similar lines by matching gradients and y-intercepts within a small threshold. The second issue that arose was outlier lines, which may occur for many reasons, including noise such as the fans and external objects with long straight edges. To detect and filter lines in this situation, we implemented a consensus method, grouping lines in the same direction and eliminating any that were further than the threshold away from the median gradient of the group. After all previous steps were applied, if there were two lines in one direction and three in the other, we classified that frame as having valid features.

Finally, we reached a situation where court lines could be successfully extracted

from a large number of still frames. The next step was to derive useful information from the line positions and gradients. Using the equations of lines found, we found their intersections and considered the four corner intersections to use as the frame's representative points. As mentioned previously, these intersections represent different points on the 2D representation of the court, depending on their orientation. Hence, we implemented a method to identify the direction of play and choose the points in the 2D representation correspondingly. This was a simple method, consisting of counting the number of lines with a positive gradient versus lines with a negative, which, as visualised in Figure 18, can determine which side of the court is in the frame.

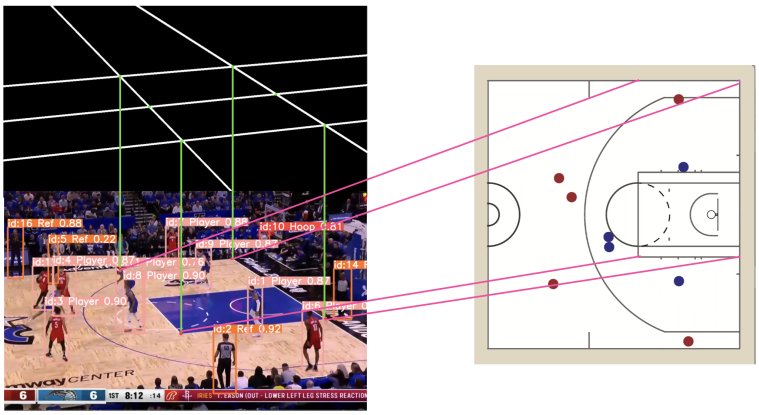


Figure 18: Right side of the court inferred by Hough Line Transform results.

With this information, we could calculate the homography matrix for the specific frame and represent points in the footage in their respective positions on the 2D court by multiplying their feet coordinates (x, y) by this matrix, H.

The next stage of this process was to consider issues pertaining to the nature of the footage. This included the constant moving and panning of the camera requiring frequent updates of selected points. Ideally, we could find the points in each frame and maintain a completely accurate mapping throughout the play. However, this proved difficult due to the changing conditions throughout a play including players on court lines and lighting scenarios. Furthermore, this process on every frame would significantly slow down the program's running.

Therefore, we needed to consider alternatives for when a frame was not valid or being taken through the process of feature detection. For this implementation, we utilised Optical Flow Tracking, which uses the apparent motion of objects between

consecutive frames to track our points [58]. Secondly, to improve the consistency of this method, we employed Kalman Filters for situations where the points were not picked up by the optical flow tracking. These estimate the trajectories of occluded points by predicting their positions based on previous movements. Due to the movement of the camera, the points move in mostly linear trajectories, so this method proved successful in keeping the points in the right place across frames where the line detection process was not being used.

We also had to account for the cases where incorrect lines were detected. This is a specific case and we discovered that it mainly occurs on certain courts at the end of plays when the camera has zoomed very close in to show the player that has just scored. The program should not detect any new lines for these camera angles, thus we calculated the Euclidian distance between the previous mapping and the newly found points every time new lines are discovered. We experimented with finding a threshold for this distance such that, for distances above this value, the program accurately identifies that these new lines do not represent the court's lines, whilst still ensuring that valid lines are not discarded. If a mapping exceeds this threshold, the current play is considered complete, and the loop for the current video footage is ended.

4.4 Player Tracking

4.4.1 Research

Basketball is a fast-paced sport with frequent player interactions, hence developing a tracking algorithm for players required careful consideration. Occlusion was again a primary concern, as players often overlap or are obscured by others. This required a system which could manage and account for players temporarily disappearing from view. Additionally, accurate re-identification of players post-occlusion was crucial. Once players reappear after being overlapped or obscured, the algorithm had to correctly reassign their identities to maintain consistent tracking throughout the game. Through our research, we assessed the suitability of various tracking methods for our use case.

A popular mathematical method used by tracking algorithms for dealing with occlusion are Kalman Filters. The Simple Online and Realtime Tracking (SORT)

algorithm leverages Kalman filters which, in the context of the algorithm, predict the future state of an object based on its current and past observed positions. When an object becomes occluded and is momentarily lost, the Kalman filter continues to predict the object's trajectory based on its previous movement, allowing for the re-identification of the object once it becomes visible again [59].

Since the introduction of SORT, the field of object tracking has seen significant advancements. Various adaptations have aimed to enhance the tracker's accuracy, particularly in terms of identity retention. Two notable developments are Deep Learning for Real-Time Multi-Object Tracking and Segmentation (DeepSORT) and Bounding-box Transformer for Simple Online and Realtime Tracking (BoT-SORT)

DeepSORT built upon the foundational ideas of SORT by integrating deep learning to improve the accuracy of the identity association across frames. DeepSORT uses a convolutional neural network to extract features from detected objects [60]. Unlike SORT, which primarily relies on motion information, DeepSORT also utilises appearance information, making it more suited to situations where motion data alone might lead to identity switches or losses. It maintains a more accurate and longer-term memory of object identities, which is important in complex scenes where objects frequently enter or exit the frame.

BoT-SORT is a more recent adaptation that integrates with YOLO directly in PyTorch. It utilises transformer models to handle the bounding box transformations. These are deep learning models that excel at handling sequences of data, and can therefore analyse the relationships between different objects over time, helping to maintain consistent tracking even when objects interact closely or become occluded. BoT-SORT maintains fast processing speeds by optimising both the detection (via YOLO) and tracking stages, making it suitable for our application [61].

4.4.2 Development

Due to the integration of BoT-SORT within the YOLO framework, we were able to use the provided *model.track(frame)* function, which assigns an ID to all detected players in a given frame, persisting the given IDs across the course of a video.

This simple implementation allowed us to rapidly move on to testing and improving the rate of tracking success. Initial results from using the built-in tracker showed high levels of accuracy in tracking, especially through occlusion. The main

issue that arose was due to limitations in the detection model causing missing players for several frames. The tracking algorithm did not perform as well as expected when re-identifying these lost players, often assigning them new IDs.

This was a problem as it led to more tracks than players on the court, which provided confusing data at the end of the play for someone attempting to analyse a single player's movement. To catch instances of this, we first established the valid set of player IDs as the first ten ID numbers given by the algorithm. This decision was based on the assumption that there are typically ten players on a basketball court. We then defined an identification failure as the scenario when a player was assigned an ID which is not in the valid set.

As a protective measure against lost players in these cases, we needed to implement a system to relate new IDs to the ones initially assigned. For this, we developed a second tracking system with more leeway to be used alongside BoT-SORT for re-identification for when it fails.

Our implementation consisted of tracking each player's position history along with several features within a nested dictionary, using their tracking ID as the key and a dictionary as the value, containing their position information and features. The position history contained their movement on the 2D court plane, which was calculated as the coordinates of the bottom centre of their bounding box multiplied by the current frame's homography matrix. The features included the player's jersey colour, how many frames it had been missing and its velocity based on its movement in previous frames.

Throughout each pass of detection, the system compiled a list of IDs for a given frame. Once every player has been detected, it identified a subset of the original IDs which had not been assigned and a list of new IDs. For the original IDs no longer being assigned, the program made a prediction for what the next position of the player could be, using its velocity. This prediction was likely to hold as it is a natural movement on a court for a player over the course of fractions of a second, which was often how long players go undetected by the detection algorithm.

Once predictions for missing players were made and new detections were processed, a cost matrix was created. This matrix's purpose was to evaluate the potential matches between newly detected players and those whose IDs were currently unassigned due to missing from recent frames. The costs were calculated based

on the Euclidian distances between predicted positions of missing players and new detections. To match new detections with previously tracked IDs efficiently, we employed the Hungarian method, to minimise the total assignment costs in the matrix [62]. This ensured that each new detection was matched with the correct previous player ID. However, we also had to consider the possibility that a missing player was still undetected in the new frame. Therefore, we added an extra check after the optimisation to ensure that the chosen assignment was within a threshold distance of the original.

If these checks were passed, the new ID was added to a lookup dictionary where it pointed to its original ID. From this point onwards, each new ID was first checked in the lookup dictionary, and only if it was not found, followed the previous steps to find a matching ID. This method ensured that for the entirety of a play, the program worked with the same ten IDs, allowing us to make players uniquely identifiable.

Players who continued to be undetected had their missed frame count incremented and their positions continually predicted. For those exceeding the threshold of allowed missed frames, we removed their data from the tracking system to avoid errors in future re-identifications and to maintain system efficiency. By removing these players from the system, we sacrifice the possibility of continuing to track them if they become visible again later in the game. However, this approach was necessary to avoid cluttering the system with outdated or inaccurate data, which could lead to errors in re-identification and negatively impact the overall clarity of the player tracking. It ensured that the system remained focused on players who are consistently visible and can be tracked with higher accuracy.

A further implementation of our new tracking method involved enhancing the team identification we introduced in Section 4.2.2. Instead of attaching a team to each player based on their initial team identification, the feature tracking worked by storing a count of how many times a given player had been identified as being on each team. The current team of a player was then the team they had been assigned to the most. This method made the team identification resistant to the fluctuations in lighting and occlusion that can cause incorrect original team assignments.

4.5 Integration

During the integration phase of the project, we emphasised achieving both modularity and maintainability to allow future works to be built off it with ease. We adopted Object-Oriented Programming (OOP) principles when combining the individual components, which allowed us to encapsulate specific functionalities within distinct classes, each corresponding to a core component of the project as well as a utility module. This encapsulation simplified the maintenance of the code and made it easier to apply our iterative development methodology to individual components.

In the utility module, we provided key functions necessary for the components. This includes the jersey colour identifier, which given a player's bounding box, outputs their team; a function which attempts to find features in a given frame; the homography matrix calculator, which computes the matrix given four coordinates from both representations; and a position conversion function that applies this matrix to translate player positions to their 2D court equivalents.

For our court mapping component, we created a Mapping class which was designed to be instantiated every time feature points are found in a frame. Once initialised, its primary function was to manage the continuous updating of these points as the game progressed. At its core, an initialised object holds the current feature points that are active within the system. Each time a new frame is processed, if new feature points have not been detected, the class then executes its *map()* function, which is responsible for updating the positions of the existing points using the approach discussed in section 4.3.2.

For our player tracking component, we developed a Tracking class, which was made to be instantiated at the beginning of the program and remain active for the duration of a play. This class manages player re-identification and movement tracking throughout the game. Upon initialisation, it sets up a structure to store tracking information and feature data for each identified player, alongside a dictionary that facilitates the ID lookup feature outlined in section 4.4.2. The core functionality of this class is to continuously update and maintain the tracking data for each player. If a player is not detected in a new frame, the class employs predictive algorithms to estimate their next position and attempts to match it to a newly identified player.

Additionally, the Tracking class provides functionalities to smooth the player

trajectories using moving averages, to allow for better visualisation of player movement. The class also handles scenarios where players are lost by temporarily suspending tracking if it has been lost for more than 10 frames and re-initiating it once the player has been detected again.

Due to this modular design choice, our main file remained uncluttered and primarily handled the high-level coordination of the different system components, which aligns with the OOP principle of encapsulation. Most importantly, this ensured that improvements or adjustments in one area of the system could be implemented without disrupting other parts, which was crucial for our iterative design process.

5 Project Architecture

In this section, we provide a comprehensive overview of our system's workflow, tying in the concepts discussed in the development section and detailing the step-by-step process through which the program derives player movement data from basketball game footage.

Before running the program, an array consisting of two RGB values correlating to the jersey colours of each team in the play must be defined to produce accurate results for our team identification system. Once set, the system can be run.

Utilising Tkinter's filedialog module, the program initially prompts the user for file input. Once a file has been selected, the program runs the *visualise(file_path, teams)* function with the chosen file path and team colours as parameters.

Before opening the file, our player detection model is loaded into a variable, an instance of the Tracking class is initialised and then the video is loaded as an OpenCV VideoCapture object. We use a while loop to iterate through the video frame by frame. The *find_features(frame)* function is called with the current frame as a parameter, which returns None if no features are detected, and the coordinates of the points and the direction of play if they are. The program iterates through frames until this function finds valid points in a frame. At this point, a Mapping object is initialised and depending on the direction of the play, the *find_homography_matrix(footage_points, court_points)* method is called with the correlating court points to obtain H.

After this initial pass, the program checks for new features every three frames, using *Mapping.map()* to obtain current points when new features are not discovered. If new features have been found, it creates a new Mapping object with this information, checking our similarity measure to ensure the play is still active.

Next, we run YOLO's *model.track(frame)* function with our model to obtain bounding boxes and ID's for all detected players. We iterate through each detected bounding box, using the *get_jersey_color()* function with the team colours array and the current bounding box to identify the players' teams. Secondly, the *transform(player, H)* function is called to transform the bounding box into the feet position of the player in the 2D court representation. Before tracking a player, the system performs a check on its tracking ID. It verifies whether there are exactly ten IDs being tracked. If there are less than ten, and the player is new, their position and features are added to the Tracking object. If there are ten and the player is already being tracked, its current position and jersey colour are updated in the Tracking object through the *update_tracks()* function. Otherwise, the player's information is temporarily stored until the end of the pass, at which point it is assigned to an existing ID.

At the end of each pass, we use the stored data in the Tracking object to plot a side-by-side visualisation of the current frame and the 2D representation of the player positions in the frame. It displays player tracks from the last 30 frames, showing the unique movements of each player. This step was necessary for the evaluation of the project, as it shows one of the real use cases of the project we outlined in Section 2.1.

The final system diagram showing how the components work together, once initial features have been found, is displayed in Figure 19.

6 Testing and Results

When introducing the testing section of the project, it is important to note that deriving quantifiable metrics for our components' performance often required data that represented the ground truth against which the system's predictions can be measured. However, in our case, obtaining this data was not feasible, as the very aim of the project was to attempt to replicate this data that is only available to

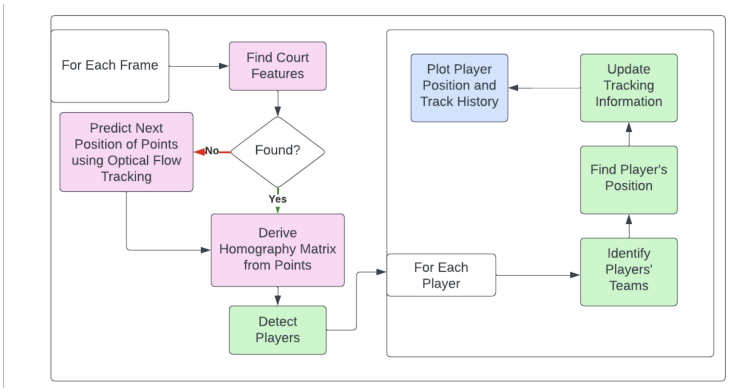


Figure 19: Project system architecture.

professionals. Consequently, where it was deemed not possible to automatically evaluate the effectiveness of a component, we relied on our visualisations to carry out a manual analysis of results.

6.1 Team Identification

In evaluating the team identification accuracy of our tracking component, the module was first tested independently across a large dataset of frames. For each frame, we had to define the two active team colours before we could begin testing. Then, the frame's team predictions were manually compared against the actual team assignments. An example frame from this process is provided in Figure 20, showing the predicted team colours for players mapped onto the court.

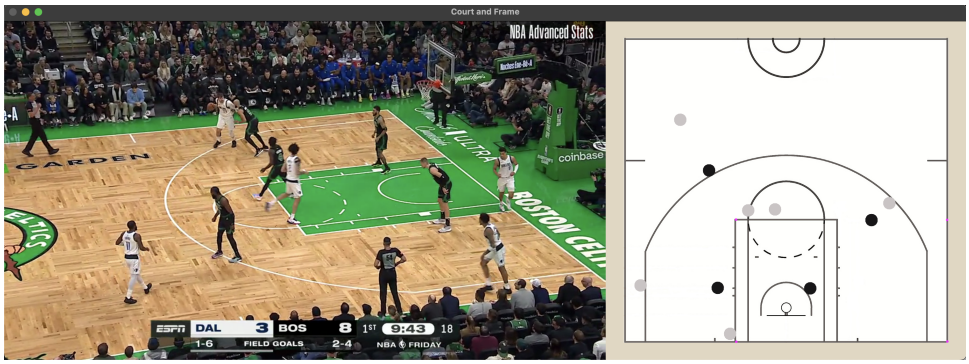


Figure 20: Snapshot of frame and players mapped onto the court with predicted team colours.

From this initial set of tests, which included fifty random frames from various games, the system correctly identified the team of 378 out of 488 detected players, resulting in an accuracy rate of 77.5%. This outcome highlighted an area of con-

cern, as approximately two out of every ten players were misidentified. This was a significant error rate, particularly given the total number of players involved in a typical basketball game.

We then applied the same testing methodology, this time also implementing the tracking algorithm's consensus approach. Unsurprisingly, this produced better results. From fifty frames, this system correctly identified the teams of 443 out of 491 detected players, a success rate of 90.1%.

An illustrative comparison of the performance discrepancy between the standalone team identification and the integrated tracking system on the same frame is shown in Figure 21.

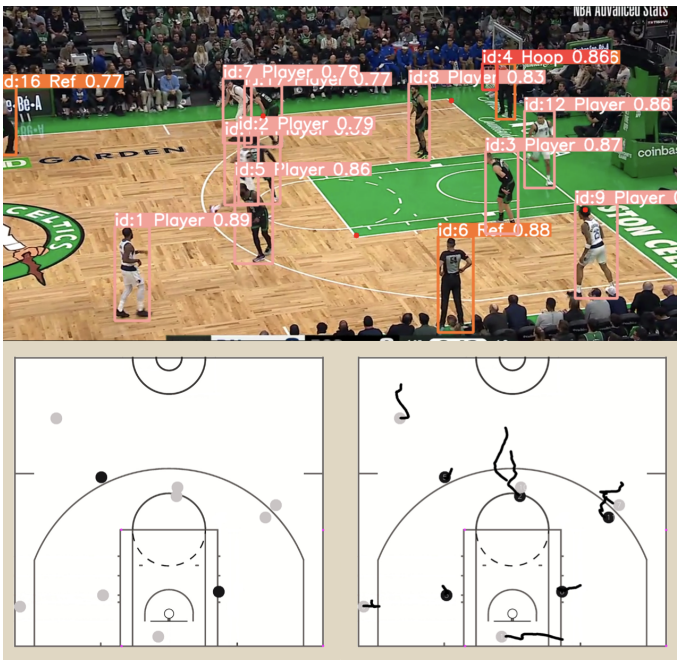


Figure 21: Original versus tracking-enhanced team identification on an identical frame.

This improvement of 12.6 percentage points demonstrated the value of incorporating our feature tracking method for team identification. An accuracy rate of 90.1% is especially impressive given the quality of the visual data involved. We concluded that these results are acceptable for our project objective and that they affirm the viability of extracting data from TV footage.

6.2 Court Mapping

During the testing phase of the court mapping module, our focus was on evaluating both the accuracy of the identified points on the court and the consistency with which the system could maintain an accurate mapping over the duration of a basketball play. Our methodology for assessing accuracy involved comparing the points selected by our system to the actual reference points on the court, as viewed in various video frames.

During our manual testing, we made a crucial observation that when the system successfully identifies and maps court features within the first 10 frames of a play, this early success usually indicates that the entire play will be mapped with high accuracy. In contrast, when feature detection and mapping only occurred later in the play, the resulting mappings tended to be less accurate and less useful. In such cases, the delay in feature recognition not only rendered the mapping process less effective but also meant that the data from the beginning of the play was missing. This finding was significant as it provided a measure which could be automated.

To quantify this measure, we ran the program on fifty random plays. This testing revealed that our mapping system successfully identified and mapped court features within the first 10 frames for only 22% of the plays. This result was particularly surprising, possibly because during development, we often tested on plays from the same game where conditions are more consistent, leading to a higher success rate in those specific instances.

Despite achieving high accuracy in these isolated instances, the method's inability to reliably detect and map court features early in the play compromised the effectiveness and applicability of our system in real-world scenarios.

6.3 Player Tracking

Common metrics used to evaluate the accuracy of tracking algorithms are Final Displacement Error (FDE) and Average Displacement Error (ADE). These both involve the comparison of our results to the ground truth of the players' movement. As mentioned, we do not have access to the true movement of players. The only possibility of deriving these metrics would come from manual drawing of player paths for given plays, and comparing the results from the system against these.

This method was not feasible. If the dataset we tested against was too small, the results would not be representative of the algorithm's performance. However, we also did not have the resources to generate a large dataset of manual path annotations. Therefore, we elected to utilise the re-identification rate of players as our metric.

This decision was also based on our observation from the court mapping tests. Plays that were accurately mapped typically also exhibited high accuracy in player movements. This suggested that tracking accuracy was closely tied to the successful detection and mapping of plays, which our system has shown to handle effectively when conditions are favourable.

Our testing process involved calculating the average number of frames a player's ID remains unmatched for a given play. To generate a baseline value, we used the built-in BoT-SORT tracker and noted the frequency with which new IDs were introduced, signalling lost players. On testing this for fifty plays across randomly selected games, we found that, on average, initial player tracks were lost for about three players per game without any tracking adjustments.

To produce comparable results, we tested our enhanced implementation on the same set of plays. We discovered that in our system, the average number of frames with unmatched IDs was reduced to approximately eight frames per play. In regards to lost players, this was an event which occurred in only ten of the plays. This significant improvement underscores the effectiveness of our implementation in maintaining consistent player identification throughout a play.

7 Evaluation

7.1 Project Objective

Referring back to the requirements outlined in Section 3.1, the system produced can be considered a success, to an extent. It would feel unfair to claim that the 'Must Have' components have all been fulfilled. Whilst this is a true claim for a given number of specific plays, our results demonstrated that the largest issue regarding our final product was the consistency of these results. Whilst we yielded accurate results when all components worked in harmony, this was uncommon. Our results

demonstrated significant strength in regard to accuracy, involving player detection, team identification and player tracking, however, the system only produced accurate mappings for a smaller number of plays and situations. Secondly, our program is still heavily reliant on user input, particularly regarding the selection of team colours. While this is the case, it will not be satisfactory for the advanced purposes we discussed in our motivations.

It is still important to realise that our main objective for this project was to prove that we could use publicly available TV footage to derive insights into player positioning and movement. This project successfully shows that such analysis is indeed possible.

7.2 Project Management

As stated in our project planning phase, this was a project that required significant amounts of research and programming involving new methods and advanced concepts. Our development strategy worked as anticipated, and the time dedicated to researching proved valuable. More relevantly, the iterative model was the most successful part of our approach. Whilst our Gantt chart provided a good structure to follow, we underestimated the difficulty of the court mapping component and how long it would take to implement a successful tracking strategy. As planned, our management strategy was flexible enough to account for this scenario. We ended up splitting the initially planned work segments into more sprints and developing smaller parts of the components in each one. At this point, instead of being required to re-plan our timelines and objectives, we were able to use the prioritisation-based requirements backlog to systematically work through the development of the remaining parts. Whilst we overran our initial planned deadline, we were able to complete our 'Must Have' requirements. The actual timeline of our project is shown in Appendix B.

8 Future Work

As reflected, the project does not currently fulfil its objective to the fullest extent due to its lack of consistency across all game situations and its incomplete automation. Therefore, regarding future efforts, immediate improvements to the project

should involve working on improving both the court mapping and the team identification methods to work with higher success rates for all games. Secondly, for automation purposes, the program should have a method to automatically detect the two team colours without user inputs. The most important addition to the current stage of the project would be the accurate detection of the basketball and the tracking of its movement throughout each play. As useful as it is to know how the players move, a lot of insight is lost to the issue of not knowing where the ball is.

It is also useful to discuss possible extensions of the project assuming the existing features are improved. If the program can reach a stage of acceptable accuracy for over 90% of the games tested and provide an automated system to input footage, we can create a large dataset of plays categorised by game metadata. This would include information such as game importance, whether the play ended in a make or a miss, time left, team records and player availability. Through this dataset of player positions and movements, users could derive insights regarding what plays have higher success rates in which situations, providing a high level of analysis on the comparison of different teams' offensive and defensive strategies.

This experience can further be enhanced through the integration of machine learning models for the classification of plays into types. By learning from the relationship between player movements and positions, they could be trained to distinguish between zone and man-to-man defence or identify isolation offence [63]. This brings us to the main goal of the project. If we can utilise these models to draw new conclusions about team play styles and match-up strengths, we can display them for fans in a graphical format with backed-up evidence, providing a new way of interacting with the sport. Once this level of insight has been successfully achieved, the final improvement would be the real-time integration of this model into live NBA games, providing users with predicted plays and percentages of success as they watch the game, not dissimilar to the fan engagement-improving methods employed by the NFL.

The implementation of these future works is reliant on many assumptions regarding the potential limits of our current project. Nonetheless, pursuing these developments would allow us to fully realise our vision of revolutionising how fans, researchers, and aspiring analysts can engage with the game of basketball.

9 Conclusions

Our project has proven largely successful. From the project initialisation and through the initial research phase, there were uncertainties about the feasibility of our methods, especially within the project's deadlines. These cast doubts on our ability to produce a set of results which could demonstrate the achievement of our objective. Although our final results do not consistently provide a level of accuracy that is good enough to draw insights and plot player movement for every single play, they are sufficient for a large number of plays. This confirms our belief that TV footage is a viable method of extracting game information. As stated in our original goals, this project was undertaken as a proof of concept and has now shown the potential for further development to improve its functionality and fulfil our goal of making sophisticated game analysis accessible to the general public. This project should hence be viewed as an innovative approach to our defined problem, and one that provides an outlet of opportunities to enhance fans' interactions with sport.

References

[1] Fortune Business Insights. Sports Analytics Market Size, Share & COVID-19 Impact Analysis, By Deployment, By Type, By Solution, By Technology, By End-user, and Regional Forecast, 2023-2030;. Available from: https://www.fortunebusinessinsights.com/sports-analytics-market-102217.

[2] Buchheit M SB. Player-Tracking Technology: Half-Full or Half-Empty Glass? IJSPP. 2016. Available from: https://doi.org/10.1123/ijspp.2016-0499.

[3] Forbes. NFL Team Valuations;. Available from: https://www.forbes.com/lists/nfl-valuations.

[4] NFL. Next Gen Stats;. Available from: https://operations.nfl.com/gameday/technology/nfl-next-gen-stats/.

[5] Ghasem W, Valenzuela J, Saxon LA. Player tracking technology and data for injury prevention in the National Football League. Current Sports Medicine Reports. 2021 Sep;20(9):436–439.

[6] NFL. Using Artificial Intelligence to Advance Player Health and Safety;. Available from: https://www.nfl.com/playerhealthandsafety/equipment-and-innovation/aws-partnership/using-artificial-intelligence-to-advance-player-health-and-safety.

[7] El-Maghrabi Y, Sharif M. Game Changers or Game Predictors? Big Data Analytics in Sports for Performance Enhancement and Fan Engagement. Journal of Contemporary Healthcare Analytics. 2022 Jun;6(6):19–39. Available from: https://publications.dlpress.org/index.php/jcha/article/view/47.

[8] Sport F, Gaming Association (FSGA). Industry Demogaphics;. Available from: https://thefsga.org/industry-demographics/.

[9] Fortunato JA. The relationship of fantasy football participation with NFL television ratings. Journal of Applied Sport Management. 2011;3(1):16.

[10] Lee J. The effects of fantasy football participation on team identification and NFL fandom;. Available from: https://repository.lsu.edu/cgi/viewcontent.cgi?article=2610&context=gradschool_theses.

[11] Yuksel M, McDonald MA, Milne GR, Darmody A. The paradoxical relationship between fantasy football and NFL consumption: Conflict development and consumer coping mechanisms. Sport Management Review. 2017;20(2):198-210. Available from: https://www.sciencedirect.com/science/article/pii/S1441352316300316.

[12] Wade McElwain. AWS: The Promise of Next Gen Stats;. Available from: https://www.sportsboom.com/nfl/super-bowl/aws-the-promise-of-next-gen-stats/.

[13] Forbes. NBA Team Valuations;. Available from: https://www.forbes.com/lists/nba-valuations.

[14] Stats Perform. Optical Player Tracking;. Available from: https://www.statsperform.com/team-performance/basketball/optical-tracking/.

[15] McCann Z. Player tracking transforming NBA analytics;. Available from: https://www.espn.com/blog/playbook/tech/post/_/id/492/492.

[16] Sathish S. How NBA is Leveraging the Power of Data Analytics in the Game;. Available from: https://logandata.com/how-nba-is-leveraging-the-power-of-data-analytics-in-the-game/.

[17] McCagh W. How Spatial Analytics Killed The Mid-Range Jump Shot;. Available from: https://www.vice.com/en/article/pgj338/numbers-game-how-spatial-analytics-killed-the-mid-range-jump-shot.

[18] Emmons M. Understanding Data: The Golden State Warriors And The Role of Analytics;. Available from: https://the-cauldron.com/understanding-the-data-the-golden-state-warriors-and-the-role-of-analytics-37c1b387c7b

[19] Ramírez LC. Analyzing the Evolution of Three-Point Shooting in the NBA;. Available from: https://longomatch.com/en/blog/post/Analyzing-the-Evolution-of-Three-Point-Shooting-in-the-NBA/.

[20] NBA Communications. Stats LLC and NBA to make STATS SportVU Player Tracking data available to more fans than ever before;. Available from: https://pr.nba.com/stats-llc-nba-sportvu-player-tracking-data/.

[21] Vatterott D. Creating Videos of NBA Action With Sportsvu Data;. Available from: https://www.pybloggers.com/2016/06/creating-videos-of-nba-action-with-sportsvu-data/.

[22] Snyder H. Literature review as a research methodology: An overview and guidelines. Journal of Business Research. 2019;104:333-9. Available from: https://www.sciencedirect.com/science/article/pii/S0148296319304564.

[23] Yani Dzhurov IIK Sylvia. Personal Extreme Programming – An Agile Process for Autonomous Developers. 2009. Available from: https://core.ac.uk/works/7870760.

[24] Frendy Ardiansyah Ardiansyah WS Gita Marthasari. Personal Extreme Programming with MoSCoW Prioritization for Developing Library Information System. Institute of Advanced Engineering and Science. 2019. Available from: https://core.ac.uk/works/10564853.

[25] NBA. Terms of Use;. Available from: https://www.nba.com/termsofuse.

[26] Beck K. Extreme programming explained: embrace change. addison-wesley professional; 2000.

[27] O'Connor R. PyTorch vs TensorFlow in 2023;. Available from: https://www.assemblyai.com/blog/pytorch-vs-tensorflow-in-2023/.

[28] Roboflow. Top Basketball Datasets and Models;. Available from: https://universe.roboflow.com/search?q=class:basketball.

[29] Zou Z, Chen K, Shi Z, Guo Y, Ye J. Object Detection in 20 Years: A Survey. Proceedings of the IEEE. 2023;111(3):257-76.

[30] LeCun Y, Bengio Y, Hinton G. Deep learning. nature. 2015;521(7553):436-44.

[31] Girshick R, Donahue J, Darrell T, Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2014. p. 580-7.

[32] Girshick R. Fast R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV); 2015. .

[33] Ren S, He K, Girshick R, Sun J. Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems. 2015;28.

[34] Redmon J, Divvala S, Girshick R, Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. 2016 June.

[35] Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, et al.; Springer. Ssd: Single shot multibox detector. 2016:21-37.

[36] Frendy Ardiansyah Ardiansyah WS Gita Marthasari. Personal Extreme Programming with MoSCoW Prioritization for Developing Library Information System. Institute of Advanced Engineering and Science. 2019. Available from: https://core.ac.uk/works/10564853.

[37] Tan L, Huangfu T, Wu L, Chen W. Comparison of RetinaNet, SSD, and Yolo V3 for real-time pill identification. BMC Medical Informatics and Decision Making. 2021;21(1).

[38] Kim Ja, Sung JY, Park Sh. Comparison of Faster-RCNN, YOLO, and SSD for Real-Time Vehicle Type Recognition. 2020:1-4.

[39] Redmon J, Farhadi A. YOLOv3: An Incremental Improvement; 2018.

[40] Bhavya Sree B, Yashwanth Bharadwaj V, Neelima N. An Inter-Comparative Survey on State-of-the-Art Detectors—R-CNN, YOLO, and SSD. In: Intelligent Manufacturing and Energy Sustainability: Proceedings of ICIMES 2020. Springer; 2021. p. 475-83.

[41] Terven J, Cordova-Esparza D. A comprehensive review of YOLO: From YOLOv1 to YOLOv8 and beyond. arXiv preprint arXiv:230400501. 2023.

[42] Ultralytics. ultralytics;. Available from: https://github.com/ultralytics/ultralytics.

[43] Barbedo JGA. Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification. Computers and electronics in agriculture. 2018;153:46-53.

[44] Zoph B, Cubuk ED, Ghiasi G, Lin TY, Shlens J, Le QV. Learning data augmentation strategies for object detection. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16. Springer; 2020. p. 566-83.

[45] Poojary R, Raina R, Mondal AK. Effect of data-augmentation on fine-tuned CNN model performance. IAES International Journal of Artificial Intelligence. 2021;10(1):84.

[46] Gao Y, Liu Y, Zhang H, Li Z, Zhu Y, Lin H, et al. Estimating gpu memory consumption of deep learning models. In: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering; 2020. p. 1342-52.

[47] Hartley R, Zisserman A. Multiple view geometry in computer vision. Cambridge university press; 2003.

[48] OpenCV. Feature Matching + Homography to find Objects;. Available from: `https://docs.opencv.org/4.x/d1/de0/tutorial_py_feature_homography.html`.

[49] Tareen SAK, Saleem Z. A comparative analysis of sift, surf, kaze, akaze, orb, and brisk. In: 2018 International conference on computing, mathematics and engineering technologies (iCoMET). IEEE; 2018. p. 1-10.

[50] Karami E, Prasad S, Shehata M. Image matching using SIFT, SURF, BRIEF and ORB: performance comparison for distorted images. arXiv preprint arXiv:171002726. 2017.

[51] Kiryati N, Eldar Y, Bruckstein AM. A probabilistic Hough transform. Pattern recognition. 1991;24(4):303-16.

[52] OpenCV. Hough Line Transform;. Available from: `https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html`.

[53] Maini R, Aggarwal H. Study and comparison of various image edge detection techniques. International journal of image processing (IJIP). 2009;3(1):1-11.

[54] Ahmed AS. Comparative study among Sobel, Prewitt and Canny edge detection operators used in image processing. J Theor Appl Inf Technol. 2018;96(19):6517-25.

[55] Vit P. Comparison of various edge detection technique. Int J Signal Process Image Process Pattern Recognit. 2016;9:143-58.

[56] Vijayarani S, Vinupriya M. Performance analysis of canny and sobel edge detection algorithms in image mining. International Journal of Innovative Research in Computer and Communication Engineering. 2013;1(8):1760-7.

[57] Abo-Zahhad M, Gharieb RR, Ahmed SM, Donkol AA. Edge detection with a preprocessing approach. Journal of Signal and Information Processing. 2014;05(04):123–134.

[58] OpenCV. Optical Flow;. Available from: `https://docs.opencv.org/4.x/db/d7f/tutorial_js_lucas_kanade.html`.

[59] Bewley A, Ge Z, Ott L, Ramos F, Upcroft B. Simple online and realtime tracking. 2016:3464-8.

[60] Wojke N, Bewley A, Paulus D. Simple online and realtime tracking with a deep association metric. In: 2017 IEEE international conference on image processing (ICIP). IEEE; 2017. p. 3645-9.

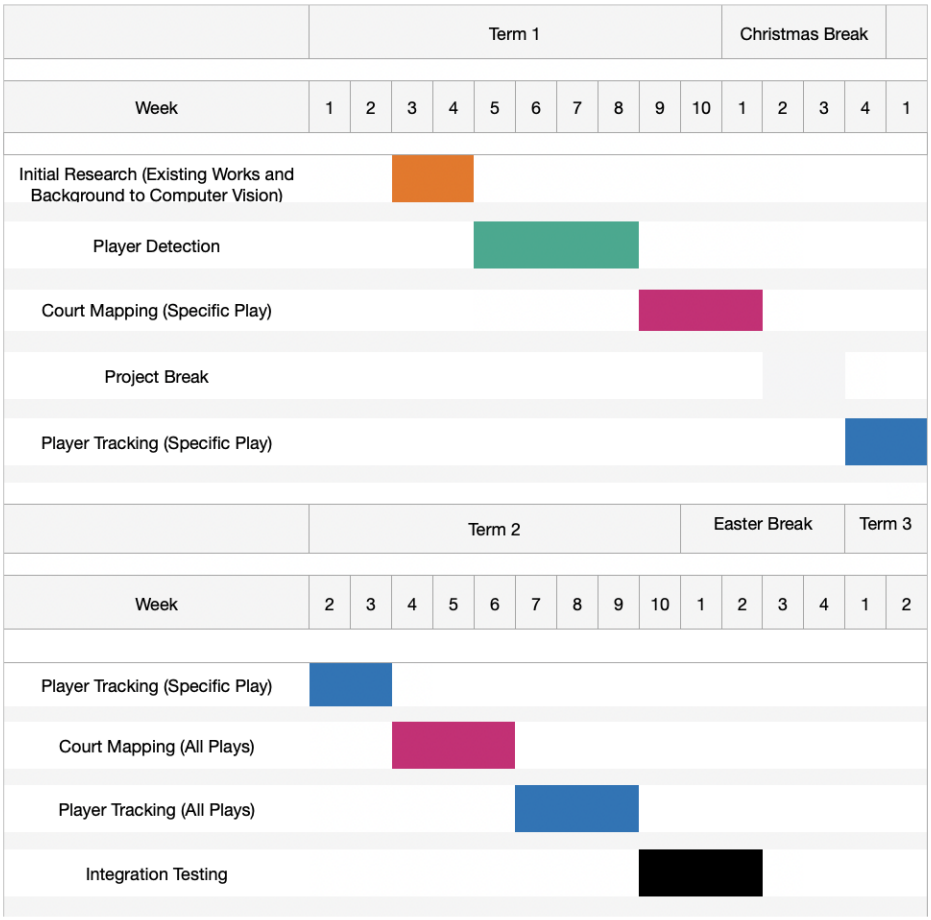
[61] Aharon N, Orfaig R, Bobrovsky BZ. BoT-SORT: Robust associations multi-pedestrian tracking. arXiv preprint arXiv:220614651. 2022.

[62] Kuhn HW. The Hungarian method for the assignment problem. Naval research logistics quarterly. 1955;2(1-2):83-97.

[63] Owayo. The Xs and Os: Basketball Tactics and Strategy;. Available from: `https://www.owayo.co.uk/magazine/basketball-strategy-tactics-en.htm`.

10 Appendicies

A Initial Gantt Chart



B Actual Project Timetable