Open University of Catalonia (UOC) Master's Degree in Data Science

MASTER'S THESIS

Pattern Recognition and Computer Vision in Applied Sciences

Computer Vision System for Football Performance Analysis

Author: Miquel Arisa Fuente

Tutor: Xavier Paolo Burgos Artizzu

Professor: Ismael Benito Altamirano

Barcelona, January 8, 2025

Credits/Copyright

FINAL PROJECT RECORD

Title of the project:	Computer Vision System for Football Performance Analysis
Author's name:	Miquel Arisa Fuente
Collaborating teacher's name:	Xavier Paolo Burgos Artizzu
PRA's name:	Ismael Benito Altamirano
Delivery date (mm/yyyy):	01/2025
Degree or program:	Master's in Data Science
Final Project area:	Pattern Recognition and Computer Vision in Applied Sciences
Language of the project:	English
Keywords	Computer Vision, Football Analysis, Machine Learning

Dedication/Quote

To my parents, David and Anna, for teaching me the value of hard work; to my sister, Marina, for her unwavering belief in me; and to my partner, Raquel, whose love and constant support have been my shelter throughout this journey.

"The only way to do great work is to love what you do."

– Steve Jobs

Acknowledgements

I would like to thank my tutor, Xavier P. Burgos-Artizzu, for his invaluable guidance and continuous encouragement throughout the development of this thesis. His insights and expertise have been essential in shaping this work.

Abstract

This project aims to develop a football analysis system based on computer vision and machine learning, designed specifically for football academies and modest football clubs. Currently, professional football teams use advanced technologies to track players and analyze their performance in real time, but these tools are often too expensive and complex for smaller clubs. The goal of this project is to democratize access to these technologies by creating an affordable and easy-to-use system that can detect and track players during matches. Using computer vision techniques, the system will capture key performance metrics such as ball possession, player influence zones, speed, and distance covered. The project will involve the development of machine learning algorithms to ensure accurate tracking and analysis of player movements. The system will be tested in real-world football environments to validate its functionality and reliability. By making such technology accessible to lower-budget teams, this project aims to enhance player development and decision-making processes in clubs that otherwise wouldn't have access to professional-level analytics tools. The expected outcome is a user-friendly tool that can help coaches and clubs better understand their team's performance through data-driven insights.

Keywords: Computer Vision, Deep Learning, Football Performance Analysis, Machine Learning, Player Tracking

Contents

List of Figures

List of Tables

1 Introduction

1.1 Context and motivation

The use of advanced technologies to analyze football performance has become an integral part of the professional sports industry. Systems that track player movements, measure key metrics such as ball possession, speed, distance covered, and assess areas of influence are commonly used by elite football clubs to optimize tactics and improve player performance. However, these technologies are often too costly and complex for football academies and small clubs. The lack of access to such tools creates a disparity between professional teams and lower-tier clubs, limiting the ability of these smaller organizations to make data-driven decisions and improve their training processes.

This project aims to bridge that gap by developing an affordable and easy-to-implement system based on computer vision and machine learning. The intended outcome is a system capable of tracking and analyzing football players' movements, generating valuable performance metrics, and making these insights accessible to coaches and football academies. By doing so, the project intends to democratize access to cutting-edge technology, enabling clubs with limited resources to benefit from the same level of analysis as professional teams.

From a personal perspective, the motivation for this project stems from my passion for both data analysis and football. In my previous academic work, I developed a scouting platform for roller hockey, a small sport but very practiced where I live, Vic. This project allowed me to see the potential of technology in enhancing player evaluation and team strategy. My experience in that project fueled my desire to apply similar techniques to football, my true passion. By focusing on modest clubs and football academies, this project not only addresses a real-world need but also aligns with my goal of making technology more accessible in the world of sports, fostering development in lower-budget environments.

1.2 Goals

Primary objective

The main objective of this project is to develop a computer vision system capable of detecting, tracking, and analyzing football players during matches. The system will be initially tested using video footage from professional football matches to ensure its functionality and accuracy in tracking player movements and generating relevant performance data.

Secondary objectives

- Generate performance metrics: The system will calculate key metrics such as ball possession, player influence zones, speed, and distance covered, providing actionable insights for performance analysis.
- Test the system with amateur football footage: After the initial testing with professional match footage, the system will be further validated using recorded segments from amateur football matches to ensure it performs effectively in diverse environments and with varying video quality.

1.3 Sustainability, diversity, and ethical/social challenges

Sustainability

The project developed does not have a significant impact on sustainability, either during its development or in its final outcomes. The most resource-intensive aspect is the retraining of the YOLO model, which is performed using a GPU from the cloud service runpod.io. This approach involves some energy consumption but optimizes the need for high-performance hardware by outsourcing this task to an external provider. The remainder of the project, including player detection, tracking, and analysis, runs on a conventional computer, further reducing energy consumption and environmental impact. While cloud computing services like runpod.io can have high resource consumption for large-scale projects, in this specific case, the occasional use of a cloud-based high-performance GPU eliminates the need to acquire and maintain costly hardware. This not only reduces resource consumption and the ecological footprint but also contributes to sustainability by avoiding the wear and continuous energy consumption that would be involved in maintaining the equipment physically. Although retraining the model involves some energy consumption, the environmental impact is limited since this is a process that occurs at specific intervals rather than continuously. Overall, this approach balances the need for computational resources with energy efficiency, contributing to a more sustainable use of technology.

Ethical behaviour and social responsibility

The outcome of this project primarily impacts the sports industry by providing a tool for performance analysis in football. It does not directly involve highly sensitive areas like data privacy or personal security, as the system is built to analyze publicly available or authorized match footage without infringing on individuals' private data. However, it is essential to ensure that any footage used is obtained legally and respects intellectual property rights, especially regarding professional match recordings.

From a professional ethics standpoint, the project adheres to the principles of computer science and engineering by promoting transparency, fairness, and responsible use of technology. There is no risk of the system being used for harmful purposes, as its application is strictly related to sports analysis. In terms of social responsibility, the project could potentially democratize access to performance analysis tools, benefiting clubs with fewer resources and helping them improve decision-making and player development. This could contribute to job improvements for analysts and coaches, who will have access to better data for player assessment, without threatening any existing job positions.

Diversity, gender, and human rights

The technical nature of this project means it does not have a direct positive or negative impact on issues such as gender, diversity, or human rights. Its primary purpose is to analyze football matches and evaluate player performance, which is neutral in terms of diversity and inclusion. However, the project has the potential to indirectly promote diversity and equity in football by providing accessible analysis tools to smaller, underfunded clubs, especially in regions or leagues where access to such technology is limited.

By offering affordable performance analysis solutions, this project can empower women's football clubs by bridging the gap in resources and technological access between men's and women's teams. This contributes to creating a more equitable environment where clubs, regardless of gender or financial capacity, can leverage data-driven insights to enhance their performance. The democratization of advanced analytics in football helps level the playing field and supports the continued growth and professionalization of women's football.

Moreover, the project benefits low-budget clubs and football academies, both male and female, that often lack the resources to invest in high-end analysis tools. By providing an affordable and effective solution, the project ensures that clubs at all levels, from grassroots academies to semi-professional teams, can access performance data typically reserved for top-tier organizations. This inclusivity not only fosters diversity but also strengthens the overall football ecosystem by offering all clubs—regardless of size—the ability to improve through data-driven strategies.

1.4 Approach and methodology

1. **Data collection:** Video footage from both professional and amateur football matches will be gathered. Professional match footage will be used in the first phase to evaluate detection and tracking accuracy, followed by testing with amateur footage to validate the system's robustness in various environments.

2. **Model development:** The YOLO model will be fine-tuned for detecting football players

within each frame of the video. Once detected, the ByteTrack algorithm will be applied to track players across consecutive frames, ensuring consistency in player tracking.

3. **Team differentiation:** K-means clustering will be applied to the detected players, using the color of their jerseys to distinguish between the two teams.

4. **Performance metrics generation:** The system will calculate key performance metrics in real time, such as ball possession (differentiated by team), player influence zones, speed, and distance covered.

5. **Testing and validation:** The system will first be tested with professional match footage to verify detection, tracking, and team differentiation accuracy. It will then be validated using amateur football footage to assess its performance in more varied and lower-quality video conditions.

1.5 Schedule

The project is structured into two main phases: project implementation and report documentation, with a total duration of 20 weeks spanning from August to December 2024. The tasks for each phase have been carefully planned and scheduled, ensuring that milestones align with the Continuous Assessment Tests.

In the implementation phase, key tasks include defining the project scope, setting up the environment, testing and fine-tuning the YOLO model, integrating ByteTrack for player tracking, and implementing advanced metrics. This phase begins on August 5th, 2024, and concludes on December 1st, 2024.

The report documentation phase focuses on documenting the project results, methodology, and conclusions, with a detailed review at the end. It spans from September 23rd, 2024, to December 22nd, 2024.

The Gantt diagram (Figure 1) illustrates the schedule, task durations, and dependencies, providing a clear overview of the project's timeline.

Task name	Duration	Start date	Finish date	W1 05-08	W2 12-08	W3 19-08	W4 26-08	W5 02-09	W6 09-09	W7 16-09	W8 23-09	W9 30-09	W10 07-10	W11 14-10	W12 21-10	W13 28-10	W14 04-11	W15 11-11	W16 18-11	W17 25-11	W18 02-12	W19 09-12	W20 16-12
Report Documentation	13	23-09-2024	22-12-2024																				
Writing the project definition	1	23-09-2024	29-09-2024																				
Writing the introduction of the report	2	30-09-2024	13-10-2024																				
Writing the state of the art	3	14-10-2024	03-11-2024																				
Writing the methodology and resources	3	04-11-2024	24-11-2024																				
Writing the results	3	25-11-2024	15-12-2024																				
Writing the conclusions	2	09-12-2024	22-12-2024																				
Final review of the project	1	16-12-2024	22-12-2024																				
Implementation	17	05-08-2024	01-12-2024																				
Definition of project goals and scope	1	05-08-2024	11-08-2024																				
Environment setup and Git repository creation	1	12-08-2024	18-08-2024																				
Data collection and initial preparation	2	12-08-2024	25-08-2024																				
Testing the YOLO model	2	26-08-2024	08-09-2024																				
Fine-tuning YOLO for improved detection	3	09-09-2024	29-09-2024																				
Integration of ByteTrack for player tracking	3	23-09-2024	13-09-2024																				
Detection improvement and position refinement	2	14-10-2024	27-10-2024																				
Implementation of player color assignment	2	28-10-2024	10-11-2024																				
Field mapping and camera adjustment	2	11-11-2024	24-11-2024																				
Calculation of speed and distance metrics	2	18-11-2024	01-12-2024																				

Figure 1: Project timeline and task distribution

1.6 Summary of the outputs of the project

The primary output of this project is the development of a system for detecting, tracking, and generating performance metrics of football players using computer vision and machine learning techniques. The focus of the project is on creating a robust backend solution for player detection and analysis, without the development of a graphical user interface.

Key outputs of the project include:

1. **Player detection and tracking:** The system employs the YOLO model for player detection and ByteTrack for tracking player movements across video frames with high accuracy.

2. **Team differentiation:** A k-means clustering algorithm is implemented to distinguish between teams based on the color of their jerseys.

3. **Performance metrics:** The system extracts key performance metrics, including ball possession, player influence zones, speed, and distance covered during the match.

These outputs provide the core functionality required for accurate football player analysis and performance tracking, intended for use in grassroots and underfunded football clubs.

1.7 Brief description of the remaining chapters of the report

The structure of this report has been carefully organized to ensure clarity and coherence, providing a comprehensive overview of the project and its findings. Each chapter builds upon the previous one, offering a logical progression through the various stages of the work. Below is a brief description of each chapter and its relationship to the overall project:

- **Chapter 2: State of the art**
  This chapter presents a thorough review of the existing literature, methodologies, and tools related to computer vision in football. It provides context by summarizing achievements, current challenges, and potential applications, as well as discussing similar software and relevant hardware. This chapter lays the foundation for understanding the project's contributions.

- **Chapter 3: Methods and resources**
  This chapter describes the methodologies, datasets, and tools employed in the project. It includes details on the dataset preparation, environment configuration, model selection, training and validation processes, and postprocessing techniques. The information in this chapter is essential to understanding the technical approach and experimental setup.

- **Chapter 4: Results**
  In this chapter, the results of the experiments are presented and analyzed. It includes model performance, comparisons between models, tracking and player assignment results, speed and distance metrics, and the application of the model to amateur football. The findings are discussed in detail to evaluate the success and limitations of the approach.

- **Chapter 5: Conclusions and future work**
  This chapter summarizes the key findings of the project, reflecting on the extent to which the objectives have been achieved. It also outlines potential future work and improvements to extend the project's impact and applicability.

- **Chapter 6: Glossary**
  This section provides definitions of technical terms and concepts used throughout the report to ensure accessibility for readers with varying levels of expertise.

- **Chapter 7: Appendices**
  The appendices include supplementary material, such as the Conda environment configuration and evaluation code for metrics, that supports the main text and facilitates reproducibility.

Together, these chapters create a cohesive narrative that details the motivation, methodology, results, and impact of the project.

2 State of the art

2.1 Introduction

Artificial Intelligence (AI) has undergone profound evolution over the past few decades, shifting from rule-based systems in the 1950s to complex machine learning models in the present day. Initially, AI sought to create machines capable of emulating human intelligence, focusing on tasks such as problem-solving and symbolic reasoning (McCarthy et al., 1956). The field of computer vision, a sub-discipline of AI, originated in the 1960s with a primary goal of enabling machines to interpret visual data. Early methods in computer vision were limited, relying on basic pattern recognition and manual feature extraction, which constrained their applicability to real-world problems (Rosenfeld, 1969).

The development of machine learning in the late 1980s marked a turning point, enabling computers to "learn" patterns from data rather than relying on explicitly programmed rules. This was further accelerated by the advent of Deep Learning in the 2000s, which introduced Convolutional Neural Networks (CNNs) for image processing (LeCun et al., 1998). CNNs, a form of deep neural networks, became the foundational architecture for most modern computer vision applications due to their ability to automatically extract hierarchical features from images. This breakthrough laid the groundwork for real-time object detection and tracking, which are crucial in fields like sports analytics.

Deep Learning represents a subset of machine learning focused on using neural networks with many layers to learn from large datasets. CNNs, introduced by LeCun et al. in the late 1990s, revolutionized image recognition by automatically identifying features through layers of convolutions. The layers in a CNN are designed to mimic the human visual cortex, processing image pixels in small sections (receptive fields) and passing information through multiple layers to identify increasingly complex patterns (LeCun et al., 1998).

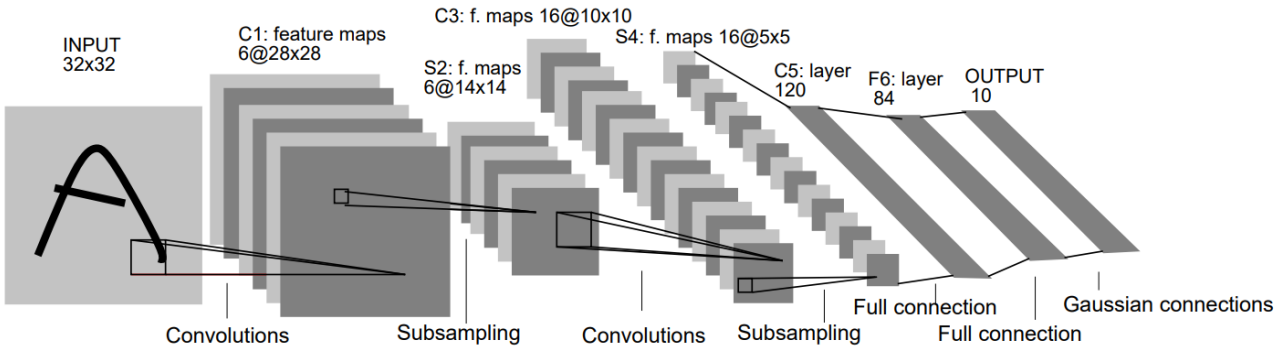

Figure 2: Basic structure of a convolutional neural network. [Source: LeCun et al. (1998)]

The development of architectures such as AlexNet (Krizhevsky et al., 2012) and later improvements like VGGNet (Simonyan and Zisserman, 2014) and ResNet (He et al., 2016) further

enhanced CNN performance, allowing computer vision applications to reach unprecedented accuracy. This laid the groundwork for the development of specialized object detection networks, such as YOLO (Ultralytics, 2023) and Faster R-CNN (Ren et al., 2015), which are now integral to real-time tracking systems in sports.

2.2 Computer vision applied to football

In professional football, computer vision has been pivotal in transforming player performance monitoring and tactical decision-making. Algorithms such as YOLO (Ultralytics, 2023) have become central to object detection and tracking due to their ability to accurately identify multiple objects in real time. This real-time capability is crucial in football, where player positioning and movement significantly impact game dynamics. YOLO, developed by Redmon et al. in 2016, uses a single neural network to divide the image into regions and predict bounding boxes and class probabilities simultaneously, making it highly efficient for rapid object detection in complex environments (Redmon et al., 2016).

YOLO's performance in sports analytics is particularly notable in capturing metrics such as player speed, influence zones, ball possession, and passing patterns. In football, this allows clubs to analyze team dynamics and assess individual player contributions effectively. Figure 3 shows YOLO's detection capabilities, identifying multiple objects such as a person, a dog, and a horse with high accuracy. This accuracy makes YOLO well-suited for real-time tracking in complex environments like sports.

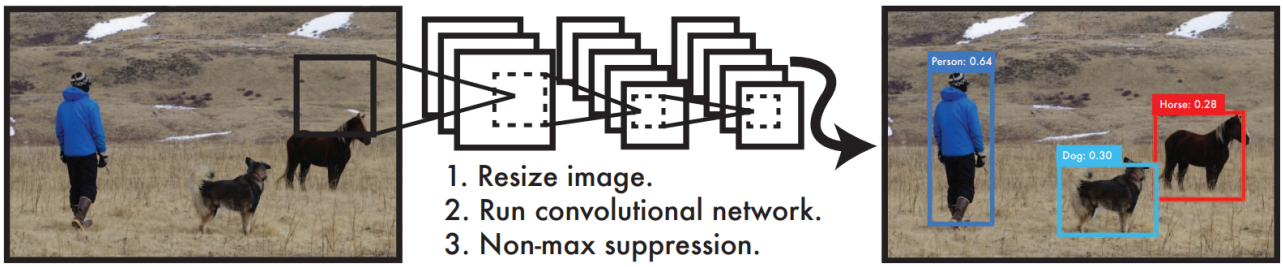

Figure 3: Example of YOLO detection. [Source: Redmon et al. (2016)]

Another key algorithm in computer vision for football analytics is ByteTrack, an advanced multi-object tracking system. ByteTrack has gained popularity for its robustness in crowded and dynamic settings, such as football fields where players frequently overlap and change directions rapidly. ByteTrack's algorithm addresses the issue of maintaining high tracking precision in scenarios with multiple moving objects by re-associating lost objects and maintaining consistency over consecutive frames. As a result, it has become a preferred choice for player tracking in high-density scenes (Zhang et al., 2021). Professional platforms like TRACAB (TRACAB, 2023) leverage these advanced tracking methods, providing elite leagues like the Premier League

with in-depth, real-time performance analytics.

Automated camera systems have further enhanced the capabilities of computer vision in sports by providing autonomous, multi-angle footage of matches. Companies like Pixellot have developed camera solutions that use computer vision algorithms to track game action without the need for human operators. Pixellot's system utilizes multi-camera setups that can capture different angles of the field simultaneously, and its algorithms are capable of autonomously following the ball and players throughout the match (Pixellot, 2023). This technology not only streamlines the recording process but also enables comprehensive data collection, which can be analyzed post-game to extract valuable performance metrics.

2.3 Achievements in the field

The field of sports data analytics has achieved remarkable progress over the past decade, largely due to advancements in computer vision and data processing capabilities. Companies like Opta and StatsBomb have been at the forefront of this transformation, using advanced data collection techniques to analyze every aspect of football matches. These companies gather thousands of data points per match, enabling the creation of comprehensive metrics such as heat maps, passing networks, and expected goals (xG) (Opta, 2023; StatsBomb, 2023). This level of detail allows for an unprecedented understanding of both individual player performance and team strategies, facilitating a data-driven approach to tactical decision-making and scouting.

The data collected by Opta and StatsBomb is not only available in real time but also provides extensive historical records, allowing for comparative analysis across different teams, players, and seasons. These databases support longitudinal studies and trend analysis, which can reveal insights into team dynamics, player development, and tactical evolution. Furthermore, APIs provided by these platforms enable independent analysts and fans to access and utilize data, democratizing football analytics and fostering a broader understanding of the game. Figure 4 illustrates a typical Opta-generated heat map, showing player movement density on the field.

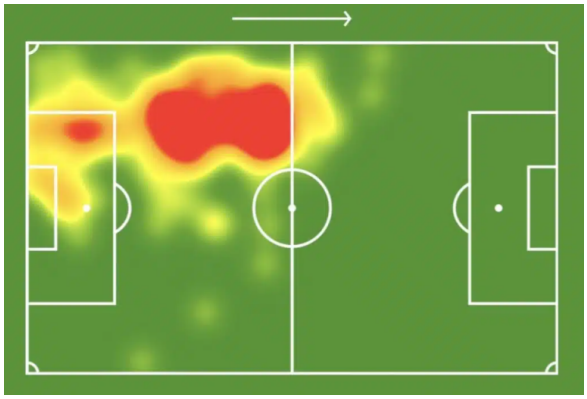

Figure 4: Heat map showing player movement density. [Source: FigPii Team (2024)]

In addition to advancing team and player analysis, the achievements in this field have implications for the broader sports industry. The detailed, data-driven insights provided by these platforms are used by coaches to develop training plans, by scouts to identify potential talent, and by broadcasters to enhance the viewing experience through in-depth match analysis.

2.4 Current challenges

Despite the advancements in computer vision and data analysis in football, several challenges persist, especially for smaller clubs and academies. A primary challenge is the high cost associated with implementing these technologies. Professional platforms like Opta and Second Spectrum require costly subscriptions and high-quality recording infrastructure, making them inaccessible for many clubs outside the elite (Second Spectrum, 2023).

Another significant challenge lies in data accuracy when using low-quality video footage. Smaller clubs often record matches with basic cameras or mobile devices, which can hinder the ability of computer vision algorithms to accurately identify and track players. Factors such as video resolution, lighting, and camera distance significantly affect tracking accuracy, which can limit the effectiveness of these technologies in amateur settings (Fang et al., 2018).

Furthermore, scalability is a major consideration. While professional systems are designed to operate within leagues that have substantial investments in technology and resources, adapting these tools for semi-professional or amateur environments (such as football academies or local competitions) presents a unique set of challenges, particularly when considering budget constraints and technical limitations (Gade et al., 2019).

2.5 Potential applications in other fields

Computer vision applications in football are part of a broader trend where similar technologies have been deployed in other sports to monitor performance. In basketball, for example, detection and tracking technology is used extensively in the NBA through platforms like Second Spectrum, which provides detailed data on player movements and ball interactions (Second Spectrum, 2023). Additionally, computer vision is applied in motor sports for tracking drivers and in athletic competitions for monitoring athletes, allowing for precise analysis of performance metrics such as speed, trajectory, and movement patterns. These applications have greatly enhanced the data available to coaches, athletes, and analysts, allowing for more data-driven approaches to training, strategy, and performance improvement.

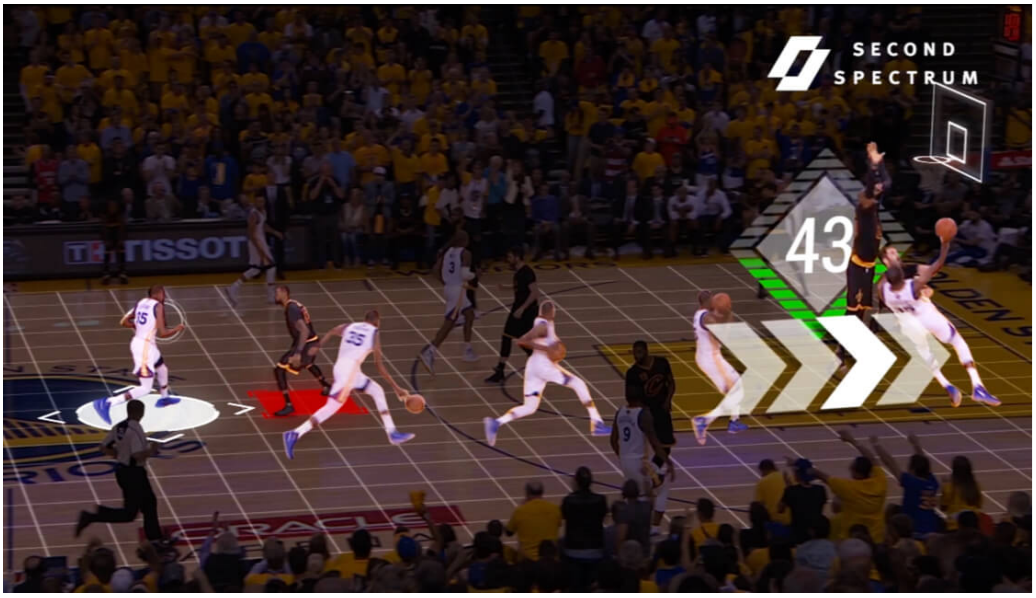

Figure 5: Second Spectrum application in the NBA. [Source: Second Spectrum (2023)]

Beyond sports, computer vision has made significant inroads into other fields, demonstrating its versatility and potential for impact across industries.

In the medical field, computer vision has proven to be transformative, particularly in areas such as patient monitoring, diagnostic imaging, and surgery assistance. One prominent application is in patient rehabilitation, where computer vision systems can monitor the progress of physical therapy sessions, tracking patient movements to ensure correct execution of exercises and assessing improvements over time. These systems help therapists tailor treatments to individual needs by providing objective data on patient performance (Yang et al., 2021). In diagnostics, computer vision algorithms analyze medical images (such as X-rays, MRIs, and CT scans) to detect anomalies like tumors or fractures with high accuracy, assisting radiologists in making faster and more precise diagnoses.

Furthermore, computer vision is increasingly used in surgery, where real-time image processing helps guide surgeons during procedures. For instance, in minimally invasive surgeries, computer vision systems can highlight anatomical structures or track surgical instruments, providing surgeons with critical information in real time. This technology is being integrated into robotic surgery platforms to enhance precision and improve patient outcomes (Litjens et al., 2017).

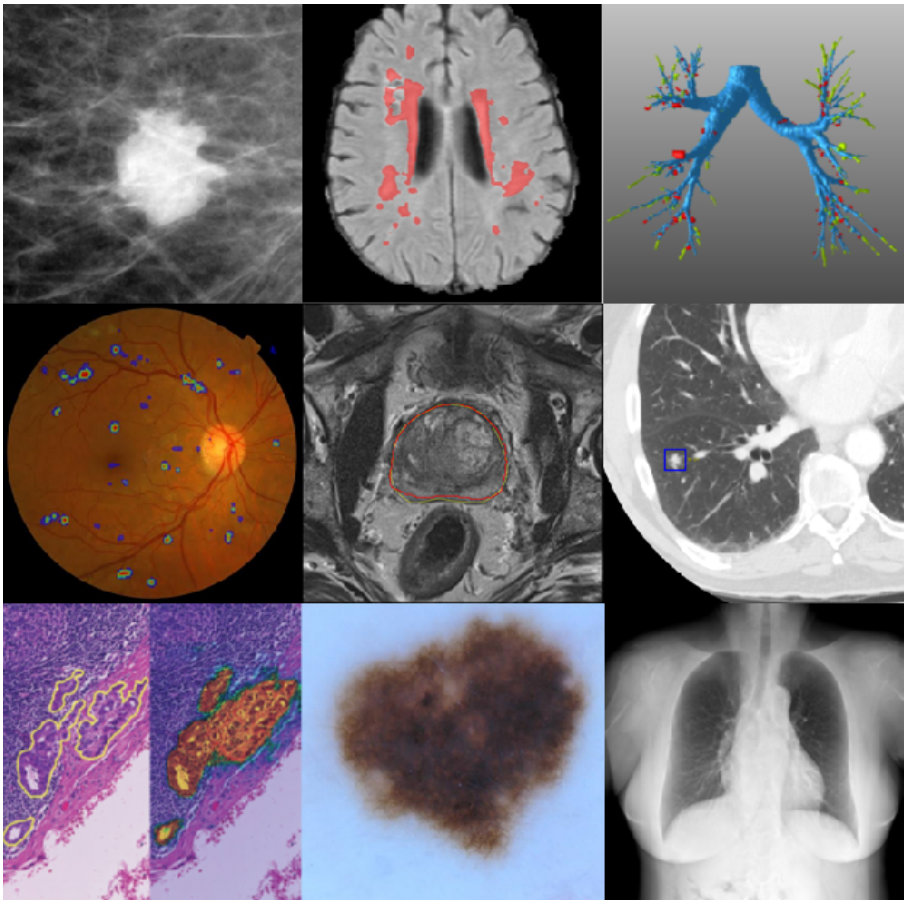

Figure 6: Illustration from "A survey on deep learning in medical image analysis." [Source: (Litjens et al., 2017)]

In the logistics and supply chain sector, computer vision is used to optimize inventory management, streamline warehouse operations, and enhance package tracking systems. Automated systems equipped with computer vision can monitor the movement of products within warehouses, track inventory levels in real time, and detect damaged goods before they are shipped. This improves operational efficiency and reduces human error, which is especially valuable in large-scale distribution centers (Garcia et al., 2020).

Moreover, computer vision is employed in package sorting and distribution. By identifying labels, barcodes, or even specific product characteristics, computer vision systems enable fast and accurate sorting of parcels in distribution hubs. This technology is also integrated into autonomous vehicles and drones used for last-mile delivery, allowing them to navigate complex environments, recognize obstacles, and ensure safe and efficient deliveries (Zhu et al., 2021).

2.6 Similar software

Various software solutions, both open-source and commercial, provide a foundation for comparison with this project. Kinovea is an open-source motion analysis tool for sports that enables

users to manually label players and compute basic metrics. However, its efficiency is limited due to the need for manual intervention and its lack of real-time analysis and automatic player detection (Kinovea, 2023).

Conversely, commercial software like Second Spectrum has set a benchmark in sports like football and basketball. This software leverages advanced computer vision techniques for real-time match analysis, but its cost and reliance on high-end infrastructure make it inaccessible for many smaller clubs (Second Spectrum, 2023). APIs such as the one offered by StatsBomb provide match data for post-game analysis, but the data is manually generated by analysts marking each action, thus revealing a need for more automation (StatsBomb, 2023).

2.7 Related hardware

The hardware required for computer vision applications in sports can vary widely depending on budget constraints. Elite clubs often employ high-resolution, multi-angle cameras to capture every detail of a match, with systems like TRACAB using specialized cameras that track players across the field with millimeter precision (TRACAB, 2023).

For smaller clubs that cannot invest in such equipment, more affordable options, such as high-definition cameras or even mobile devices, are feasible. However, optimal recording conditions (such as appropriate angles and lighting) are essential to capture images with sufficient clarity for precise detection by computer vision algorithms (Li et al., 2021).

In terms of processing, professional solutions typically depend on high-performance GPUs to handle large volumes of real-time data. However, for clubs with limited budgets, an alternative approach is to process the data after the match rather than in real time. This method allows the use of lower-capacity computers that can analyze the footage post-match to extract performance metrics without the need for expensive hardware or advanced cloud computing services, offering an accessible alternative for modest clubs that aim to apply sports analytics technology without significant infrastructure investment (Gade et al., 2019).

3 Methods and resources

3.1 Dataset

The dataset used in this project was obtained from the Roboflow platform, specifically from the *Football Players Detection* project Roboflow (2024). It consists of a total of 372 images divided into three subsets: 298 images for training, 49 for testing, and 25 for validation.

Each image includes multiple players, a ball, and, in many cases, goalkeepers and referees. In addition to the images, the dataset provides associated text files that describe the class labels (players, ball, goalkeepers, and referees) and the corresponding bounding boxes, which define the areas in the images where these elements are located.

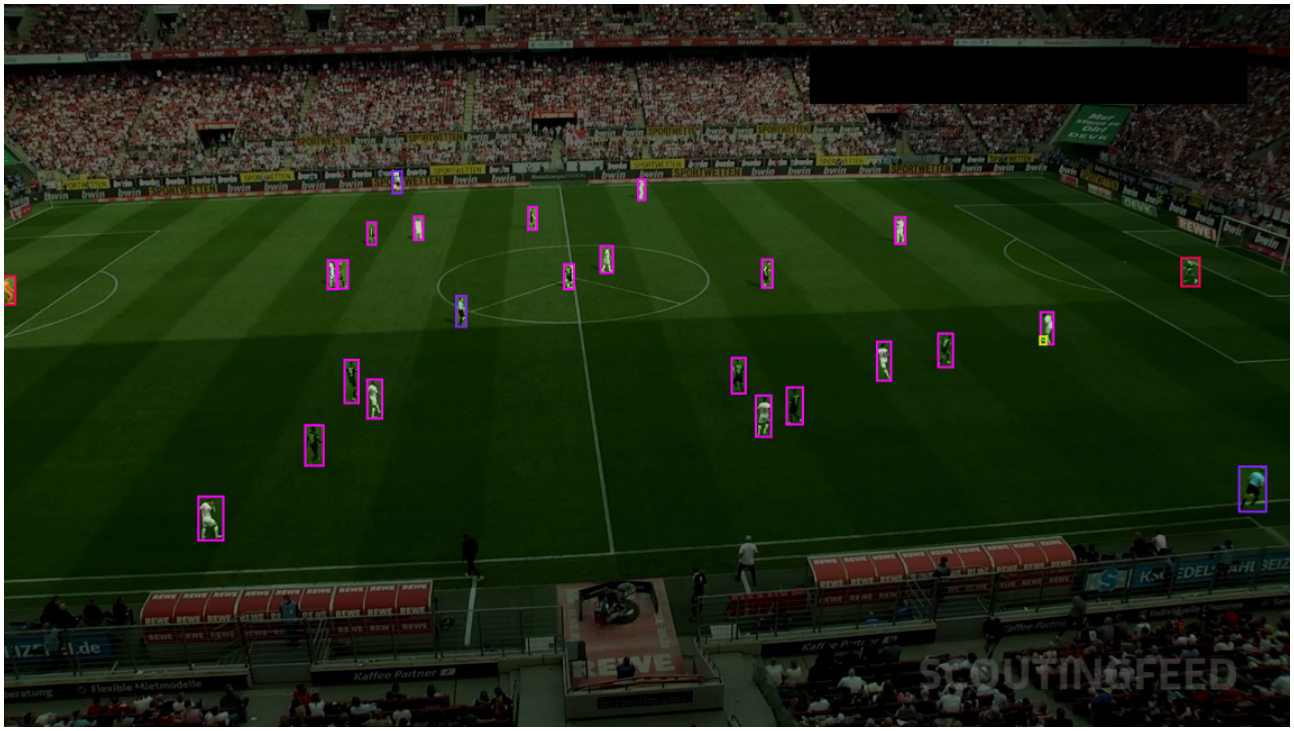

Figure 7: Example of the dataset. [Source: Roboflow (2024)]

In an initial stage, the data was processed and organized into separate directories for each subset (training, testing, and validation), ensuring that the proportions of images in each partition were maintained. This structured format allows for efficient data loading and processing using specialized libraries such as YOLO, ensuring that each text file associated with an image contains the necessary information for object detection and annotation.

The primary goal of using this dataset is to evaluate the performance of the retrained model in the task of detecting and tracking objects in the context of football matches. This process involves calculating metrics such as precision and recall, using the *Intersection over Union*

(IoU) metric to compare the model's predictions with the ground truth annotations. Table 1 summarizes the distribution of images across the dataset subsets.

Subset	**Number of images**
Training	298
Testing	49
Validation	25

Table 1: Distribution of images across training, testing, and validation subsets.

A detailed quantitative analysis of the dataset was performed to provide insights into the object distribution and size characteristics, which are critical for evaluating model performance.

Object distribution

- **Players per Image:** The dataset contains an average of **19.96** players per image, with a range of **18 to 22** players.
- **Balls per Image:** On average, there are **0.96** balls per image, with a range of **0 to 1** balls.

Bounding box analysis

- **Player bounding boxes**
  - Average area: **1332.13 px^2**
  - Standard deviation: **776.18 px^2**
- **Ball bounding boxes**
  - Average area: **132.57 px^2**
  - Standard deviation: **70.63 px^2**

Image resolution: All images have a fixed resolution of **1280 x 1280 pixels**, ensuring consistent input quality for the model.

This dataset serves as a fundamental resource for training and validating computer vision models in sports applications, enabling detailed analysis of players, ball trajectories, goalkeepers, and referees during football matches. Compared to other datasets in the football domain, this dataset is relatively modest in size but represents a diverse set of scenarios, including varying lighting conditions. While larger datasets might offer broader generalization, this dataset

is primarily composed of images from professional football matches, ensuring a high level of quality and relevance for detecting and analyzing objects in dynamic and complex real-world conditions typical of high-level play.

To ensure the applicability of the developed methodology to grassroots or modest football, I recorded several matches of a local youth football team, the Cadete A of AEC Manlleu, from my hometown. These recordings were specifically intended to test the resulting model on personally captured videos, allowing for a practical evaluation of the methodology.

3.2 Method Overview

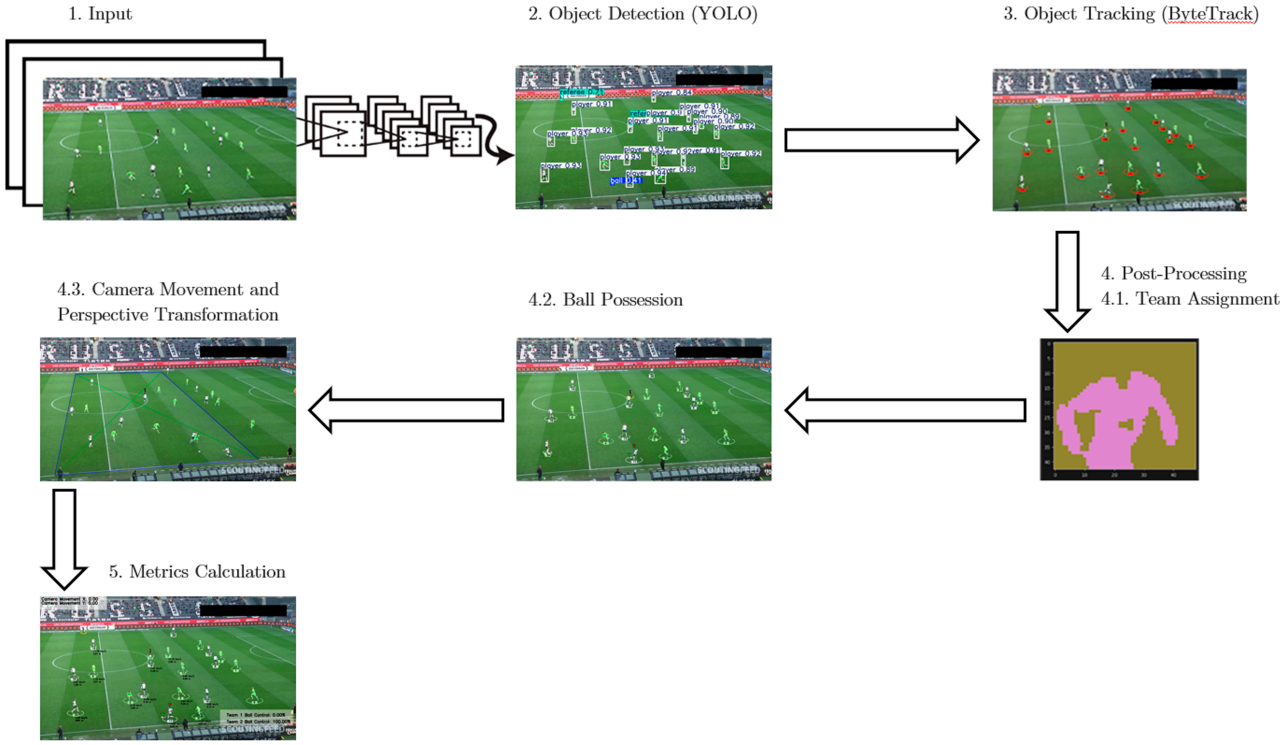

Figure 8: Processing pipeline from raw video input to metrics calculation.

The processing pipeline for analyzing soccer videos is outlined in Figure 8. The method transforms raw video data into meaningful metrics through the following steps:

1. Input (Raw Video):

- The system begins with raw video footage captured during a match.

2. Object Detection (YOLO):

- **Resize image:** Adjusts image dimensions for model input.

- **Convolutional network:** Predicts bounding boxes, class labels, and confidence scores.
- **Non-max suppression:** Filters overlapping boxes based on confidence scores.
- **Output detections:** Generates bounding boxes for players, referees, and the ball.

3. Object Tracking (ByteTrack):

- **Input detections:** Uses YOLO outputs as input.
- **Split detections:** Separates high-confidence and low-confidence detections.
- **Data association:** Matches detections between frames using IoU and Kalman Filter.
- **Track assignment:** Updates existing tracks or creates new ones, recovering lost tracks using low-confidence detections.
- **Track filtering:** Removes inactive tracks.
- **Output tracks:** Generates unique IDs and updated positions for each object.

4. Post-Processing:

- **Team Assignment:** Uses K-Means clustering to assign players to teams based on jersey colors.
- **Ball Possession:** Interpolates ball position and assigns possession to the nearest player within a defined distance threshold.
- **Camera Movement Estimator:** Estimates camera movement using optical flow to correct object positions.
- **Perspective Transformation:** Transforms coordinates from pixel space to real-world field positions using perspective mapping.

5. Metrics Calculation:

- **Ball possession time:** Measures the time each player or team holds the ball.
- **Speed and distance:** Calculates velocity (km/h) and distance covered for each player.

This structured approach ensures accurate detection, tracking, and metric computation to facilitate performance analysis in soccer matches.

3.3 Environment setup and configuration

The environment where the process was executed was created using Conda, a widely used package and environment management system. To ensure consistency and reproducibility, a dedicated Conda environment was set up by utilizing a YAML file. This YAML file contained the specifications of all required dependencies, including Python, YOLO-related libraries, OpenCV, and other auxiliary tools. By leveraging Conda's environment configuration capabilities, the setup process was streamlined, and potential version conflicts were avoided. This approach ensured that the development and execution environment was fully aligned with the project's requirements, providing a stable and reliable platform for training, evaluation, and analysis. Additionally, the exact Conda environment configuration used in this project can be found in Annex 1, offering a complete reference for replicating the setup.

3.4 Model selection

The models used in this project include YOLOv8 for object detection and ByteTrack for object tracking. These models were chosen due to their proven efficiency and accuracy in real-time applications, which is essential for analyzing football matches.

YOLOv8 is an advanced object detection model renowned for its high precision and efficiency model that is well-suited for tasks requiring fast inference and high precision. It provides an optimal balance between speed and accuracy, making it ideal for detecting players, referees, and the ball in dynamic sports environments. YOLOv8 was selected because of its ability to handle complex scenes with multiple objects and its scalability when dealing with varying image resolutions.

ByteTrack was chosen for tracking detected objects across video frames. ByteTrack is known for its robustness in maintaining consistent identities for tracked objects, even in challenging scenarios with occlusions or rapid movements. This characteristic is particularly important in football, where players frequently overlap or move quickly across the field.

The combination of YOLOv8 and ByteTrack allows for effective detection and tracking of players, referees, and the ball, ensuring that all key elements of a football match are accurately analyzed. Additionally, these models are well-documented, actively maintained, and have a strong community support, which facilitated their integration and fine-tuning for the specific needs of this project.

3.5 Training and validation process

The training and validation process was carefully configured to ensure optimal performance of the models. Specifically, the approach involved fine-tuning a pre-trained YOLO model to

adapt it to the specific requirements of football player detection and tracking. Fine-tuning allows leveraging the powerful features learned by YOLO on a large dataset while tailoring it to the nuances of the football context, which includes handling occlusions, varying lighting conditions, and rapid player movements. Below are the key aspects of the process:

3.5.1 Training configuration

The model was trained using key hyperparameters such as:

- **Epochs:** 100 and 500 epochs were used for different experiments to analyze the impact of training duration.
- **Batch size:** A batch size of 16 was selected to balance GPU memory usage and training speed.
- **Learning rate:** A learning rate of 0.01 was set initially, with a step decay strategy to reduce it progressively during training. This helps the model converge more effectively.
- **Image size:** Images were resized to 640x640 pixels to maintain a balance between model performance and computational requirements.

3.5.2 Data augmentation

Various data augmentation techniques were applied to increase the robustness of the model, such as random rotations, scaling, flipping, and color jittering. YOLO, by default, applies several data augmentation techniques, including mosaic augmentation, random cropping, random flipping, and hue/saturation adjustments. Mosaic augmentation, in particular, combines four images into one during training, which helps improve the model's ability to detect objects in complex scenes. These techniques help the model generalize better across different scenarios, as they simulate various real-world conditions that the model might encounter.

Figure 9: Training batch visualization showing YOLO detections of players, referees, and the ball with labeled bounding boxes during model training.

3.5.3 Loss function

The YOLO Loss Function was used, which consists of components for classification loss, localization loss (using IoU), and confidence loss. The classification loss helps the model determine which class each object belongs to, the localization loss ensures the bounding boxes are accurately placed around detected objects, and the confidence loss helps the model determine the certainty of an object's presence in a particular bounding box. The loss function was monitored throughout the training process to ensure effective learning, minimizing these errors to achieve better performance. This holistic approach helps balance the precision of object detection, accuracy of localization, and reliability of confidence in predictions, leading to an overall effective model.

Additionally, I have implemented a function to calculate precision and recall using the IoU metric, as detailed in Annex 2. This function is applied to validate the model with various test images, ensuring a consistent and accurate evaluation of detection performance.

3.5.4 Validation process

During training, some of the training data was used as a validation set to monitor the model's performance on unseen data. This subset was carefully selected to ensure a diverse representation of different match scenarios, including varying lighting conditions, player positions, and occlusions. The validation metrics included precision, recall, mAP@50, and mAP@50-95 to evaluate detection quality, providing a comprehensive view of the model's performance across different object classes. These metrics, which adhere to the widely recognized COCO evaluation standard, offer a robust framework for assessing object detection models and are well-documented in resources such as the Picsellia (2022) blog, where their implementation and significance are explained in detail.

An early stopping mechanism, which is a built-in feature in YOLO's training process, was implemented to prevent overfitting. This technique stops the training when the validation loss does not improve after several epochs. In this case, training was stopped if the validation loss did not improve for multiple consecutive epochs. This ensures that the model does not continue learning noise from the data, which could degrade its generalization capabilities. Early stopping is particularly useful for YOLO models, as it helps maintain a balance between underfitting and overfitting, allowing the model to generalize well to unseen data.

3.5.5 Model selection criteria

The best model was selected based on evaluating multiple key metrics to ensure accuracy and consistency across various scenarios. The primary criterion was the mean Average Precision at 50% Intersection over Union (mAP@50) score on the validation set, which measures the model's ability to correctly detect objects while minimizing false positives and negatives. Additionally, I considered the balance between precision and recall to ensure generalizability and reliability across different match conditions, considering factors like lighting, occlusions, and crowd density.

By evaluating both mAP@50 and the precision-recall trade-off, I ensured the selected model was robust in challenging situations, preventing issues such as high precision but low recall or vice versa. This balanced approach aimed to develop a model that excels in both accuracy and consistency, which is especially important in dynamic environments like football matches, where accurate detection impacts player tracking, ball possession metrics, and tactical analysis.

3.6 Hardware and software used

Hardware

GPU on RunPod.io: The training of YOLO models requires significant computational power due to the complexity and the size of the dataset. For this project, the GPU infrastructure

provided by RunPod.io was utilized, specifically an NVIDIA A40 GPU with 48 GB of VRAM. This powerful setup was complemented by 50 GB of RAM and 9 vCPUs, enabling efficient training of the models. Such high-capacity hardware allows for extensive parallel computations, essential for handling large datasets and complex neural network architectures, significantly reducing the training time while maintaining high performance. RunPod.io was specifically used to conduct model training with GPUs. By leveraging RunPod.io, the availability of high-capacity resources was ensured, which is crucial for handling the complex computations of YOLO during training. This platform allowed experimentation with different hyperparameter configurations and model versions, making it a key component of the training process.

Personal Computer: Besides the model training, much of the post-processing, such as detection analysis, player tracking, and metric visualization, was performed on a personal computer. This device was used for developing and testing Python scripts, conducting exploratory data analysis, and generating visualizations. While not as powerful as GPU infrastructure, the personal computer was sufficient for the processing and analysis tasks that did not require high computational power.

Software

Python: Python was the programming language used to develop most of the project. Scripts were created to integrate the entire workflow, from detection with YOLO, tracking with ByteTrack, to metric analysis and result visualization. Python's extensive community and specialized libraries for data analysis and machine learning provided the flexibility needed to execute this project.

YOLO (v5, v8, v11x): Several versions of YOLO were used for detecting players, referees, and the ball. These models are designed for real-time object detection with high precision. YOLOv8 was primarily used due to its optimal balance between speed and accuracy, while other versions (e.g., YOLOv11x) were tested to compare their performance under different configurations and data sizes. YOLO also provides built-in data augmentation tools that enhance the model's generalization capabilities in real-world scenarios, such as changing lighting conditions and player positions.

ByteTrack (Supervision library): ByteTrack was used for tracking players after detection. ByteTrack's robustness in identifying and tracking multiple objects simultaneously made it ideal for sports scenarios, where the dynamic and rapid movement makes precise tracking challenging. This tool helped maintain the identity of players and the ball across different video sequences, even in cases of occlusion.

Roboflow: Roboflow was a crucial tool for managing the dataset, including downloading labeled data and creating different sets for training and validation. Roboflow facilitated image transformation and augmentation, preparing the data optimally for use with YOLO.

OpenCV: OpenCV, through its Python library cv2, was used for a variety of image and video processing tasks essential to the project. These included reading videos, annotating frames with detections, and performing perspective transformations to map player positions from the camera view to real field coordinates, enabling tactical analysis of player movement and positioning. Additionally, cv2 was used for calculating optical flow to estimate camera movement and adjust player positions, providing a more accurate understanding of relative movement on the field.

K-Means (Scikit-learn): The K-Means clustering algorithm was employed for classifying player jersey colors, allowing identification and differentiation of teams based on the predominant colors of their jerseys. K-Means was applied after player detection and adjusted to identify two main clusters corresponding to the teams present in the match.

Pickle: To reduce processing time during tracking and analysis stages, Pickle was used to save and load track data and camera movements. This allowed intermediate results to be persisted and avoided recalculating data each time an analysis or result visualization was needed.

GitHub: The entire project was stored on GitHub to ensure organization, change tracking, and the preservation of the project's progress. The repository is publicly available on GitHub.

3.7 Postprocessing

The post-training process involved several key steps to enhance the analysis capabilities of the model and extract meaningful insights from the detected objects. These include tracking with ByteTrack, jersey color classification with K-Means, interpolation of ball position, camera movement estimation, perspective transformation, ball possession calculation, and speed and distance metrics. Below is a detailed explanation of the methodologies employed:

After detecting players, referees, and the ball with YOLO, the ByteTrack algorithm was used for tracking these objects across multiple frames. ByteTrack is highly effective in maintaining consistent identities of objects even in challenging situations such as occlusions, rapid movements, and crowded scenes, which are common in football matches. This tracking approach allowed the system to follow players throughout the match, providing continuity in the analysis of each player's actions and movements.

To differentiate between players from different teams, the K-Means clustering algorithm was employed to classify the colors of the players' jerseys. This process involved extracting color features from the upper regions of each detected player and applying K-Means to identify clusters representing the predominant colors in the match. The clustering was adjusted to identify two main groups, corresponding to the two opposing teams, allowing for an automated and accurate classification of players by team.

There were instances where the ball was not detected in a frame due to occlusion or other reasons. To address this, an interpolation method was implemented to predict the ball's position during these frames. By using the detected positions in the frames before and after the missed detection, a linear interpolation was used to estimate the ball's trajectory. This approach ensured continuity in the analysis of ball movement and possession, which is crucial for accurately determining game events such as passes, dribbles, and shots.

Football matches are typically recorded with dynamic camera movement, which makes it challenging to accurately track player positions. To mitigate this, optical flow techniques were used to estimate the movement of the camera between frames. The estimated camera movement was then used to adjust the positions of all detected objects, effectively compensating for the camera's panning, tilting, or zooming. This adjustment provided a stable reference frame, making it possible to analyze player positioning and movement relative to the field rather than the camera view.

To gain meaningful insights into player positioning and ball trajectories, it was essential to transform the detected positions from pixel coordinates in the video frame to real-world coordinates on the football field. A perspective transformation was applied to map the camera view to a bird's-eye view of the pitch. This transformation was performed using predefined reference points on the field, which allowed the system to convert the detected bounding boxes into accurate field positions. This step was crucial for calculating metrics such as distance covered and spatial distribution.

Figure 10: Perspective transformation mapping pixel coordinates to real-world field positions.

Ball possession was calculated by determining the proximity of each player to the ball in every frame. A distance threshold was defined, and if a player's bounding box was within this threshold from the ball, that player was considered to have possession. This calculation was performed continuously throughout the match, allowing for the analysis of possession statistics for each team and individual player. These statistics are valuable for understanding team dynamics and individual contributions.

Finally, the speed and distance covered by each player were calculated based on the transformed positions on the field. For each player, the distance traveled between consecutive frames was computed, and this value was accumulated over time to determine the total distance covered during the match. The speed was also calculated by dividing the distance traveled by the time between frames, providing insights into player performance, such as sprinting ability and work rate. These metrics were essential for evaluating the physical performance of players and comparing their contributions during the match.

4 Results

4.1 Overall detection results

This section provides a detailed analysis of the performance of the trained YOLO models (YOLOv5, YOLOv8, YOLOv11) under different configurations. Key metrics such as training and validation losses, precision, recall, and mean average precision (mAP @ 50 and mAP @ 50-95) are evaluated. Visualizations, including loss graphs and confusion matrices, are presented to illustrate model convergence, performance, and areas of improvement.

4.1.1 Model performance

Loss graphs

The following figures (Figures 11 to 16) present the evolution of training and validation losses for each YOLO model and configuration. These graphs provide valuable insights into how each model converges during training, showcasing the learning stability and potential overfitting as epochs progress.

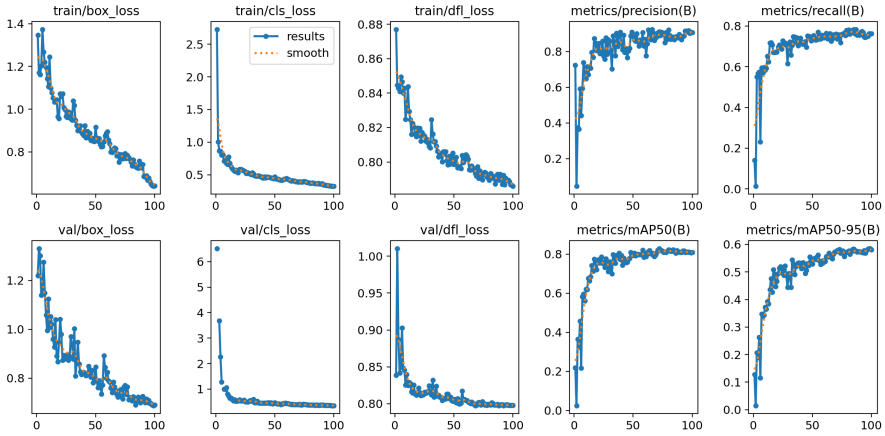

Figure 11: Loss evolution for YOLOv5x with 100 epochs.

Figure 11 shows the loss evolution for YOLOv5x trained for 100 epochs. The graph illustrates a rapid initial decline in training loss, which stabilizes towards the end, suggesting that the model reaches a satisfactory convergence within the allocated epochs.

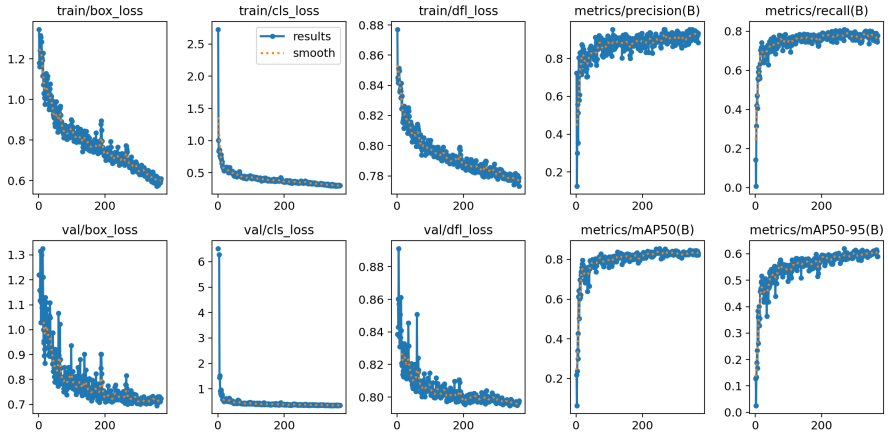

Figure 12: Loss evolution for YOLOv5x with 500 epochs.

Figure 12 extends this to 500 epochs, where the training loss continues to decrease gradually. However, the validation loss shows a slight increase in later epochs, potentially indicating minor overfitting.

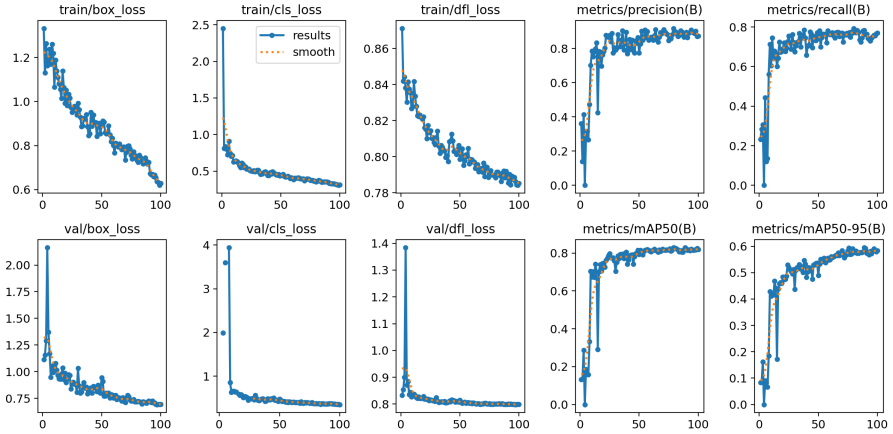

Figure 13: Loss evolution for YOLOv8x with 100 epochs.

Figure 13 shows the performance of YOLOv8x with 100 epochs. The losses stabilize faster than YOLOv5x, demonstrating improved convergence efficiency due to advances in the architecture.

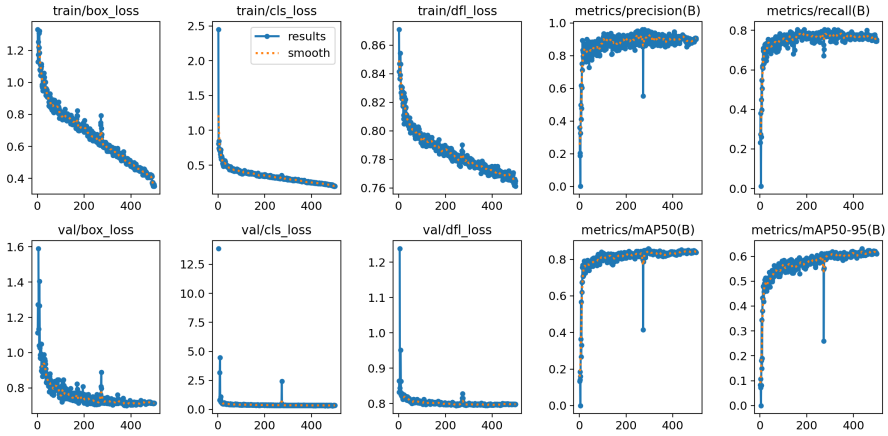

Figure 14: Loss evolution for YOLOv8x with 500 epochs.

Figure 14 highlights the results with 500 epochs. The training loss decreases further, but the validation loss remains stable without significant overfitting, showcasing the robustness of YOLOv8x for longer training sessions.

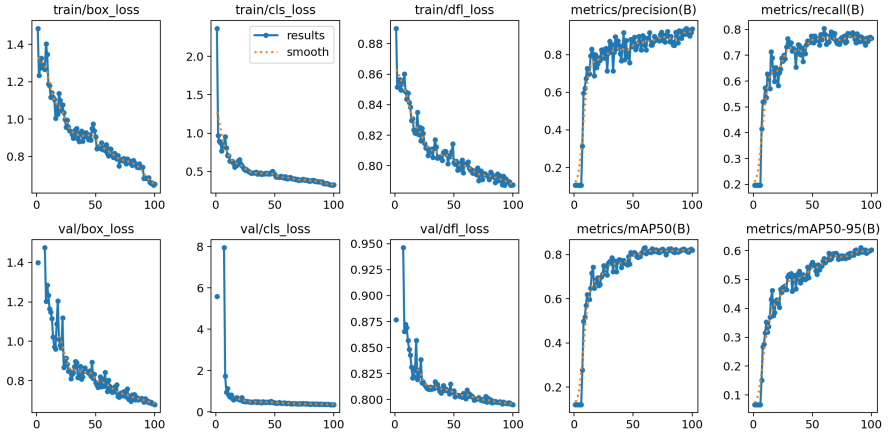

Figure 15: Loss evolution for YOLOv11x with 100 epochs.

Figure 15 presents the loss trends for YOLOv11x trained for 100 epochs. This version exhibits the fastest convergence among all models, with a steep decline in losses early in the training.

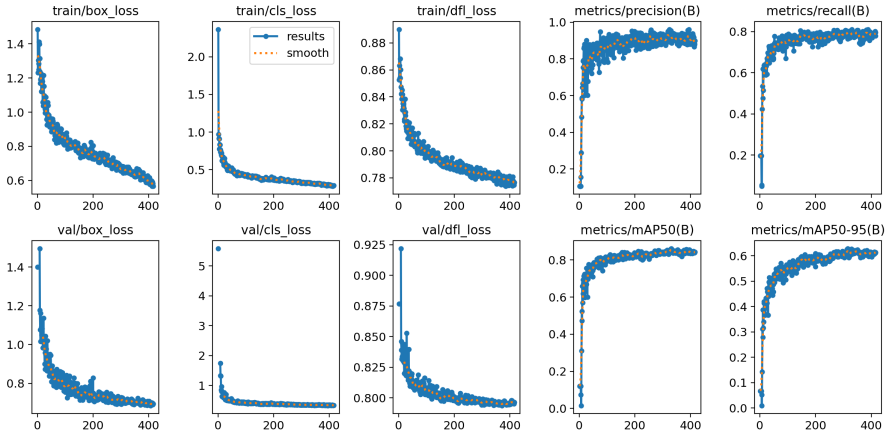

Figure 16: Loss evolution for YOLOv11x with 500 epochs.

Figure 16 demonstrates the results for 500 epochs. Similarly to YOLOv8x, the validation loss remains stable, indicating that YOLOv11x effectively avoids overfitting even with extended training.

Confusion matrices

The following figures (Figures 17 to 22) present the normalized confusion matrices for each YOLO model across different configurations. These matrices provide a detailed view of the classification performance by illustrating the proportion of predictions correctly classified versus those misclassified for each class.

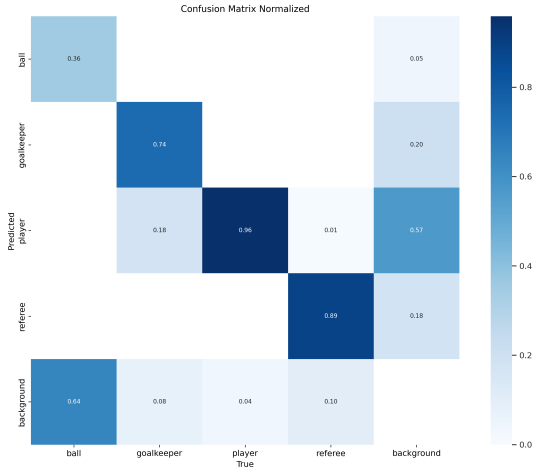

Figure 17: Confusion matrix normalized for YOLOv5x with 100 epochs, showing class-wise prediction distribution and misclassifications.

Figure 17 shows that the YOLOv5x model trained for 100 epochs achieves a 96% detection rate for players and 89% for referees. However, 18% of the true goalkeepers are misclassified as

players, indicating a challenge in distinguishing between these roles. Ball detection is particularly poor, with only 36% accuracy, suggesting significant difficulty in identifying this crucial game element.

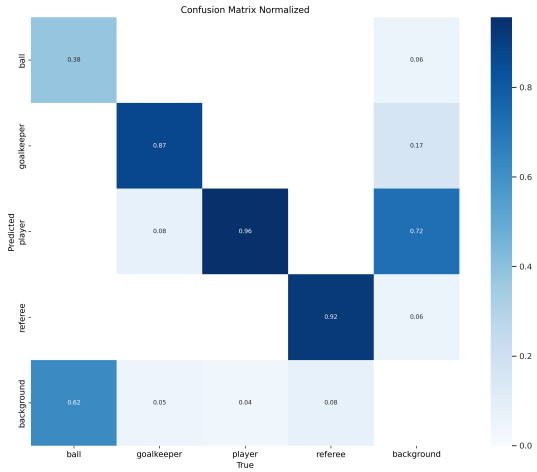

Figure 18: Confusion matrix normalized for YOLOv5x with 500 epochs, showing class-wise prediction distribution and misclassifications.

Figure 18 demonstrates an improvement in goalkeeper detection to 87%, reducing confusion with players from 18% to 8%. Player detection remains high at 96%, while referee accuracy improves slightly to 92%. Ball detection increases marginally to 38%, though it still represents a significant weakness of the model.

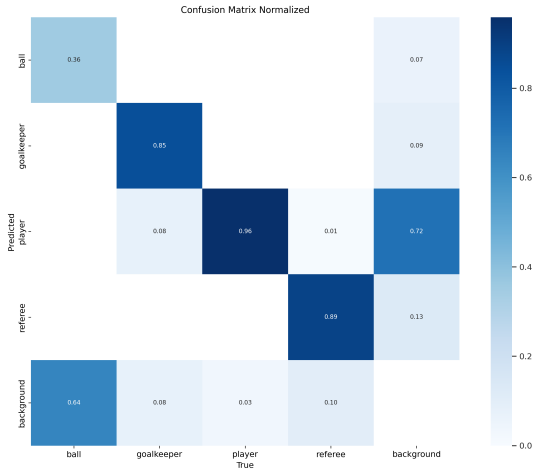

Figure 19: Confusion matrix normalized for YOLOv8x with 100 epochs, showing class-wise prediction distribution and misclassifications.

Figure 19 shows that YOLOv8x achieves an 85% detection rate for goalkeepers, improving over YOLOv5x. Player detection remains consistent at 96%, with only 8% of goalkeepers

misclassified as players. Ball detection, however, stagnates at 36%, highlighting its persistent challenges across models.

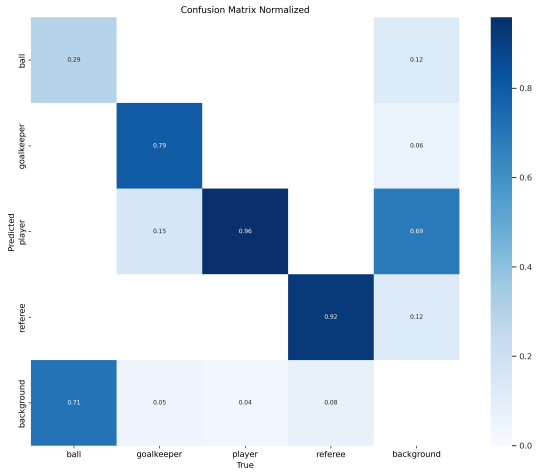

Figure 20: Confusion matrix normalized for YOLOv8x with 500 epochs, showing class-wise prediction distribution and misclassifications.

Figure 20 highlights a reduction in goalkeeper detection to 79%, with 15% misclassified as players. Player detection remains strong at 96%, while referee accuracy increases slightly to 92%. Ball detection declines to 29%, underscoring the difficulty of identifying small, dynamic objects.

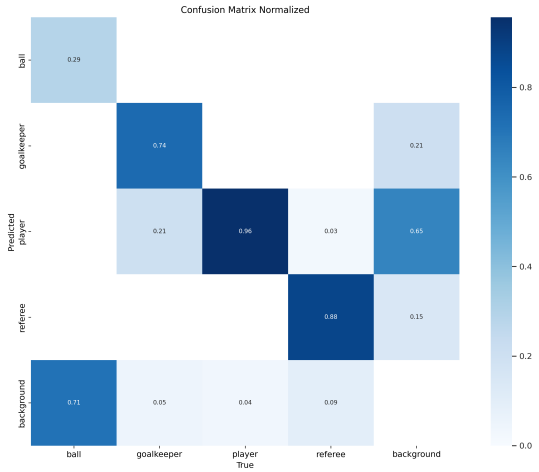

Figure 21: Confusion matrix normalized for YOLOv11x with 100 epochs, showing class-wise prediction distribution and misclassifications.

Figure 21 reveals consistent player detection at 96%, but 21% of goalkeepers are misclassified as players. The ball detection rate remains low at 29%, while referee accuracy is moderate at 88%. These results suggest that distinguishing between goalkeepers and players remains a key area for improvement.

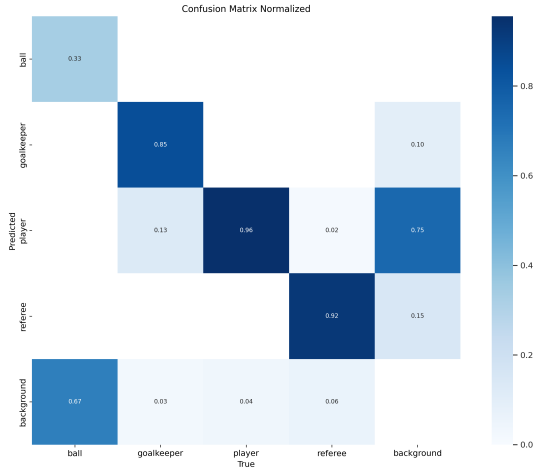

Figure 22: Confusion matrix normalized for YOLOv11x with 500 epochs, showing class-wise prediction distribution and misclassifications.

Figure 22 demonstrates a high goalkeeper detection rate (85%), with a reduced misclassification rate as players (13%). Player detection remains stable at 96%, while referee accuracy reaches 92%. Ball detection shows a slight improvement to 33%, making this configuration the most balanced across all classes.

Evaluation metrics

Table 2 summarizes key evaluation metrics for each model:

Model	Epochs	Precision	Recall	mAP50	mAP50-95	train/box_loss	train/cls_loss	val/box_loss	val/cls_loss
YOLOv5	100	0.821	0.704	0.746	0.504	0.892	0.500	0.844	0.513
YOLOv5	500	0.872	0.749	0.799	0.553	0.807	0.468	0.787	0.482
YOLOv8	100	0.801	0.700	0.736	0.496	0.886	0.485	0.849	0.499
YOLOv8	500	0.876	0.754	0.808	0.570	0.754	0.429	0.759	0.465
YOLOv11	100	0.794	0.692	0.726	0.491	0.908	0.505	0.841	0.613
YOLOv11	500	0.868	0.750	0.800	0.560	0.759	0.437	0.768	0.423

Table 2: Comparison of YOLOv5, YOLOv8, and YOLOv11 model performance with different training epochs.

The evaluation metrics presented in Table 2 provide a comprehensive overview of the performance of YOLOv5, YOLOv8, and YOLOv11 under two training configurations: 100 epochs and 500 epochs. These metrics include precision, recall, mean average precision (mAP) at 50% IoU (mAP50), mAP at IoU thresholds ranging from 50% to 95% (mAP50-95), as well as the training and validation losses for box and classification predictions.

YOLOv5 shows a noticeable improvement when trained for 500 epochs compared to 100 epochs. Precision increases from 0.821 to 0.872, recall improves from 0.704 to 0.749, and mAP50

rises from 0.746 to 0.799. However, the mAP50-95 metric only improves modestly, indicating that while the model performs well at lower IoU thresholds, it struggles slightly with higher IoU thresholds. Training and validation losses decrease consistently, which indicates that the model benefits from the extended training without significant signs of overfitting.

Similarly, YOLOv8 achieves better results with 500 epochs. Precision and recall rise to 0.876 and 0.754, respectively, while mAP50 and mAP50-95 increase to 0.808 and 0.570. YOLOv8 exhibits a smaller training and validation loss compared to YOLOv5, demonstrating its architectural efficiency. It also outperforms YOLOv5 slightly across all metrics, particularly in recall and mAP50-95, which are crucial for tasks requiring accurate detection at multiple scales.

YOLOv11, the most advanced architecture tested, follows a similar trend. With 500 epochs, the model achieves a precision of 0.868, a recall of 0.750, and an mAP50 of 0.800, closely matching YOLOv8's performance. However, its mAP50-95 value of 0.560 is slightly lower than YOLOv8's, indicating a marginal difficulty in handling detections at higher IoU thresholds. Despite this, YOLOv11 achieves the lowest validation losses among all models and configurations, suggesting strong generalization capabilities.

The comparison highlights the importance of training duration and architectural advancements. Extended training consistently improves all metrics for each model, while YOLOv8 and YOLOv11 outperform YOLOv5 across most categories, particularly in recall and mAP50-95, which are critical for applications requiring nuanced detections.

4.1.2 Comparison between models

After evaluating the performance of YOLOv5, YOLOv8, and YOLOv11, YOLOv8 is selected as the most optimal model due to its superior balance of precision, recall, and generalization capabilities across diverse object classes. This section delves into a comparative analysis of YOLOv8 and the other models, highlighting why YOLOv8 stands out as the preferred choice.

YOLOv8 vs. YOLOv5

YOLOv8 significantly outperforms YOLOv5 in nearly all aspects. While YOLOv5 shows improvements with extended training, it still struggles with distinguishing between closely related classes and maintaining stable performance across longer training durations. Its lower mAP scores and slower convergence indicate its older architecture's limitations. YOLOv8, on the other hand, benefits from architectural advancements that enable faster convergence and more stable validation losses, making it better suited for tasks requiring high accuracy and generalization.

Another critical observation is ball detection. YOLOv5 demonstrates slightly better accuracy in detecting the ball compared to YOLOv8, as shown in the confusion matrices. However,

both models face challenges with this class.

YOLOv8 vs. YOLOv11

YOLOv8 and YOLOv11 are closely matched in performance, particularly with extended training. YOLOv11 demonstrates slightly better generalization for challenging classes like goalkeepers and referees, indicating its robustness for multi-class detection. However, YOLOv11 exhibits higher training and validation losses, suggesting a less efficient learning process compared to YOLOv8. For applications where computational efficiency is critical, YOLOv8 holds an edge due to its faster convergence and lower resource demands.

In the context of ball detection, neither YOLOv8 nor YOLOv11 excels, but YOLOv8 maintains a slight advantage in consistency across different configurations. YOLOv11's strength lies in its stability across classes, making it a viable alternative for use cases emphasizing balanced performance over computational efficiency.

4.2 Results of tracking and player assignment

The analysis of the tracker and the assignment of players to teams demonstrates a high degree of effectiveness in most cases, with certain limitations that require further refinement. Below, the results are presented in detail, focusing on the detection of players, the assignment of roles, and the challenges associated with ball detection and tracking continuity.

4.2.1 Player detection

The system achieves high accuracy in detecting players, with 96% of players correctly identified. This demonstrates the robustness of the model in recognizing key entities on the field. However, the system struggles to differentiate goalkeepers from outfield players. Misclassifications occur due to similarities in movement patterns and overlapping bounding box features. To address this issue, a post-processing step was introduced to reassign detected goalkeepers to the "player" category, ensuring consistent labeling and simplifying the classification task.

4.2.2 Ball detection

Ball detection presents significant challenges, achieving only 36% accuracy. This low detection rate is a critical limitation, as the ball plays a central role in game analysis, directly influencing metrics such as possession and passes. Several factors contribute to the difficulty in detecting the ball:

- **Small size and high speed:** The ball is a small, fast-moving object that frequently blends into the background or becomes occluded by players.

- **False positives:** The model often mistakes penalty spots, white field markings, or even the heads of bald players for the ball. This results in multiple false detections in a single frame, further complicating the tracking process.
- **Inconsistent detection:** In some cases, the model detects multiple balls in a single frame, creating confusion in the tracking logic.

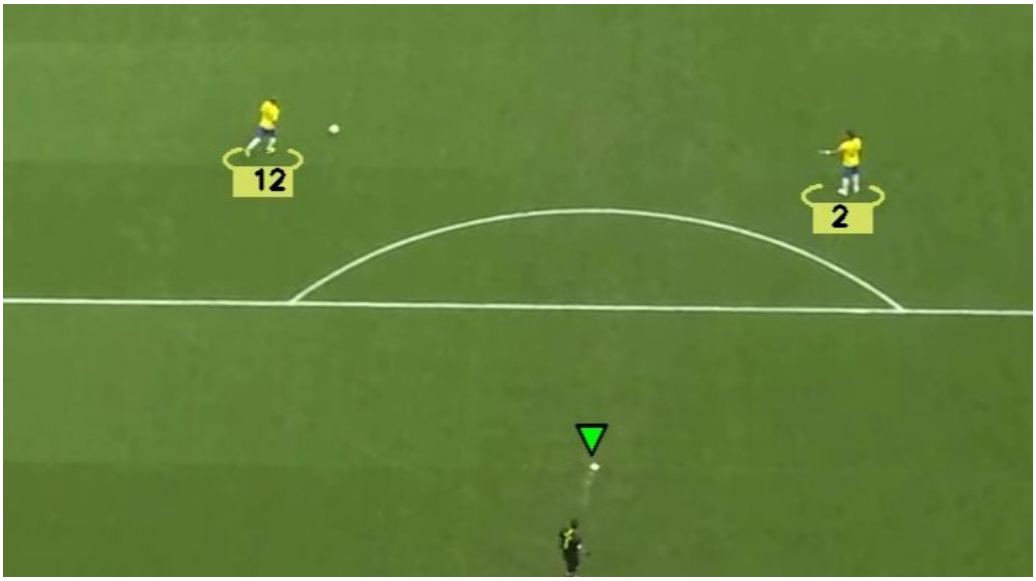

Figure 23: Incorrect ball detection: the model confuses a penalty spot with the ball [Video Source: Chepsi007 (2018)].

A method was implemented to limit ball detection to one instance per frame by prioritizing the detection closest to the ball's last known position. While this reduces false positives, it does not entirely resolve the issue, as some frames still lack correct detections.

4.2.3 Tracking performance

The tracking system functions effectively in the majority of cases, reliably maintaining the unique identifiers (IDs) of players across frames. However, there are scenarios where the tracking system encounters challenges:

- **Players exiting and re-entering the field of view:** When a player leaves the camera's field of view and subsequently re-enters, the system often assigns them a new identifier. This disrupts the continuity of their tracking data, treating them as a new player rather than resuming their previous track.
- **Overlapping players:** When two players collide or one player overlaps another, the tracker occasionally swaps their IDs. This results in incorrect assignment of tracking data, where the trajectory and statistics of one player are mistakenly attributed to another.

4.2.4 Player team assignment

The differentiation of teams and the assignment of players to their respective teams are performed with high accuracy, provided the team uniforms are distinctly distinguishable. The methodology involves selecting the upper region of each player's bounding box to identify the shirt area, followed by averaging the detected color. This approach allows to classify players into teams based on the colors of their jerseys.

When the team uniforms have very different colors, the system works perfectly, and every player is assigned to the correct team. However, problems happen when the team uniforms have similar colors because the system finds it harder to tell them apart. In these situations, it is more likely that players are assigned to the wrong team, which can affect later analyses, such as ball possession and team statistics.

A related problem occurs when two players ID swap. When two players from different teams collide or overlap, their IDs can become changed, leading to an erroneous exchange of team assignments. Since the tracker does not continuously monitor the shirt region to verify team membership, such errors remain even after the players separate. The figure (24) shows an example of this situation, showing how players with IDs 11 and 14 experience both an ID swap and an incorrect team assignment during their overlap.

Figure 24: ID swapping example between ids 11 and 14 during player team assignment [Video Source: Chepsi007 (2022)]

Even with these limitations, the system works well in most situations, especially in games

where the team uniforms are easy to tell apart.

4.3 Speed and distance metrics

The system demonstrates significant potential in analyzing player performance by calculating speed and distance metrics. These metrics provide valuable insights into players' activity on the field, such as identifying periods of intense effort or understanding overall movement patterns during a match. However, achieving accurate and reliable results is heavily dependent on the quality and continuity of the tracking data.

One of the primary challenges is that tracking errors—such as when a player's identifier (ID) changes incorrectly or when a player temporarily moves out of the camera's field of view—can disrupt the data. Such errors break the continuity required for precise calculations, leading to potential inaccuracies in the reported metrics. For instance, if a player's ID changes mid-match, their total distance covered or average speed might be underestimated or overestimated due to the fragmented data.

Professional football matches provide a structured and controlled environment that significantly aids in ensuring the accuracy of these calculations. Standardized field dimensions and consistent grass patterns, such as field markings, allow for reliable conversions of pixel measurements in the video to real-world units (meters). This process ensures that calculated distances and speeds align closely with the actual values.

However, the situation becomes more complex in youth football or amateur matches, where fields often lack standardization. Many non-professional fields have inconsistent or unclear markings, making it difficult to establish a precise pixel-to-meter conversion ratio. Additionally, the absence of clear reference points, such as field dimensions or well-defined sidelines, further complicates the process of translating video data into accurate real-world measurements.

Despite these challenges, the system provides meaningful relative insights into player performance. For example, it can reliably track trends such as how often a player sprints or how much ground they cover during different phases of the match. While the absolute values of speed and distance might have some uncertainty, these relative metrics remain consistent and useful for comparing players' activity within a single game or across similar environments.

4.4 Results of the model to amateur football

When testing the system with videos I recorded of the AEC Manlleu's Cadet A team, the results were not satisfactory. The model struggled to detect players, the ball, and the referee. These difficulties were due to the perspective of the videos, which were filmed at a lower angle compared to the tactical camera perspective used to train the YOLO model.

Since the model was trained with labeled images from tactical camera footage, it was not well-adapted to the flatter perspective of these recordings. As a result, the detections were inconsistent, and the performance metrics for player and ball tracking could not be accurately measured.

Figure 25: Example frame from an AEC Manlleu Cadet A match

5 Conclusions and Future Work

5.1 Overall detection results

5.1.1 Model performance

Loss graphs

The analysis of the loss graphs reveals important insights into the performance and behavior of the YOLO models across different configurations. Models trained for longer durations, such as 500 epochs, achieve lower training losses compared to those trained for 100 epochs. However, this improvement does not always translate into better overall performance, as extended training may increase the risk of overfitting. This is evident in the case of YOLOv5x, where the validation loss shows a slight upward trend during later epochs, indicating the model begins to overfit the training data. In contrast, YOLOv8x and YOLOv11x demonstrate faster convergence rates and greater stability throughout the training process. Both models effectively avoid overfitting even with longer training durations, as evidenced by their stable validation losses. These results highlight the architectural advancements introduced in YOLOv8x and YOLOv11x, which contribute to improved generalization capabilities and more efficient learning dynamics. Among the analyzed models, YOLOv11x emerges as the most consistent performer. It balances the minimization of losses with the prevention of overfitting, making it particularly well-suited for extended training sessions. This consistency underscores its robustness and suitability for tasks requiring high levels of precision and reliability.

Confusion matrices

The confusion matrices provide a detailed view of the classification performance for each YOLO model under different configurations. One of the most consistent findings is the high accuracy achieved in player detection across all configurations, with a detection rate of 96%. This indicates that the models are well-suited to reliably identify players, even in complex scenarios. However, a recurring challenge arises in distinguishing goalkeepers from players. In several configurations, a significant proportion of goalkeepers are misclassified as players. For example, in the YOLOv5x model trained for 100 epochs, 18% of goalkeepers are incorrectly labeled as players. Although this misclassification rate decreases with longer training durations or improved model architectures, it remains a persistent issue across all YOLO models.

On the other hand, ball detection proves to be the most challenging aspect of this task. The accuracy for detecting the ball ranges between 29% and 38%, regardless of the model or training configuration. This indicates that identifying small, dynamic objects like the ball in a soccer match is particularly difficult for these models, even with advanced architectures and

longer training sessions.

Given these findings, a key decision has been made in the project to simplify the classification task by considering goalkeepers as part of the player class. This decision was taken because, for the specific metrics being calculated in this project—such as ball possession, player tracking, and movement analysis—it is not necessary to distinguish goalkeepers from other players. By treating goalkeepers as regular players, the model's focus is shifted away from attempting to resolve a classification task that is not critical for the objectives of this project. This simplification not only reduces the computational burden but also improves the reliability of the metrics being calculated.

This approach ensures that the project remains aligned with its goals of providing accurate and meaningful insights, such as player movement and ball possession. Future iterations of the model could revisit the differentiation of goalkeepers if new requirements emerge that necessitate this level of detail. For now, simplifying the classification task has proven to be a practical and efficient solution, allowing the project to maintain its focus on delivering high-quality, actionable metrics without being hindered by unnecessary complexities.

Evaluation metrics

The evaluation metrics offer valuable insights into the comparative performance of the YOLOv5, YOLOv8, and YOLOv11 models under different training configurations. These results highlight the impact of training duration and architectural advancements on model performance.

Longer training durations, as seen in the 500-epoch configurations, consistently result in higher precision, recall, and mean average precision (mAP50 and mAP50-95). YOLOv5, for example, improves its precision by 5.1% and its recall by 4.5% when trained for 500 epochs instead of 100. However, while extended training enhances the ability of all models to generalize, the gains are less pronounced for mAP50-95, particularly in YOLOv5. This suggests that while the model effectively learns to handle easier detection tasks at lower IoU thresholds, it struggles with more challenging detections requiring higher precision in bounding box localization.

Comparing architectures, YOLOv8 and YOLOv11 consistently outperform YOLOv5 in both shorter (100 epochs) and longer (500 epochs) training configurations. YOLOv8 emerges as the most balanced performer, achieving the highest recall and mAP50-95 among all models and demonstrating a stable reduction in training and validation losses. YOLOv11, while competitive with YOLOv8, shows slightly lower mAP50-95 scores. Nevertheless, it achieves the lowest validation losses, indicating that it generalizes well and is robust to overfitting, even with extended training.

The results also highlight specific strengths and limitations of each architecture. YOLOv5, as an earlier version, performs adequately but struggles to match the efficiency and advanced capabilities of YOLOv8 and YOLOv11. YOLOv8 demonstrates architectural improvements

that lead to faster convergence and better generalization, particularly for nuanced detections requiring higher IoU thresholds. YOLOv11, as the newest architecture, balances performance and efficiency, making it suitable for scenarios where computational efficiency is a key consideration.

These findings emphasize the importance of selecting the appropriate model architecture and training duration based on the specific requirements of the task. For this project, the high recall and mAP50-95 values achieved by YOLOv8 and YOLOv11 make them particularly suitable for detecting and tracking objects in complex soccer scenarios. Future work could explore fine-tuning these models with targeted augmentations or transfer learning to further enhance their performance, especially for challenging detections such as the ball.

5.1.2 Comparison between models

YOLOv8 emerges as the optimal model due to its balance of efficiency, accuracy, and flexibility. Considering the decision to reassign the goalkeeper class to the player category, misclassification issues can be significantly mitigated, allowing the model to focus on more challenging detection tasks such as the ball. YOLOv8's superior mAP scores (0.808 at 500 epochs) and faster convergence highlight its advanced architecture and ability to generalize effectively.

The importance of ball detection cannot be overstated in football analysis, as it directly impacts the accuracy of metrics like possession and play patterns. YOLOv8's consistent performance in this area, although not perfect, demonstrates its potential for further refinement. Additionally, its low training and validation losses suggest a more stable learning process, reducing the risk of overfitting and ensuring reliable performance across datasets.

While YOLOv11 offers competitive metrics and better class consistency in some cases, its higher losses and computational demands make it less favorable for this application. YOLOv5, despite its improvements with extended training, lacks the architectural advancements to match the performance of YOLOv8 and YOLOv11.

In conclusion, YOLOv8 is the best model for this task. Its ability to adapt to evolving class definitions, coupled with its strong generalization and efficient learning process, makes it the ideal choice for analyzing football videos and detecting key objects like players and the ball.

5.2 Results of tracking and player assignment

5.2.1 Player detection

The decision to reassign goalkeepers to the "player" category proved effective for maintaining consistency in labeling. This approach aligns with the project's objectives, as distinguishing goalkeepers from other players is not essential for the calculated metrics. The high detection accuracy for players highlights the system's capability to reliably identify the primary entities

on the field. However, improving the system's ability to differentiate roles could be explored in future iterations if required.

5.2.2 Ball Detection

Detecting the ball in football matches is very challenging. This is because the ball is small, moves quickly, and is often hidden by players or parts of the field. Although the current approach, which limits detections to one ball per frame, reduces errors, it is still not strong enough for fast and dynamic situations. Improving this detection is essential because tracking algorithms cannot work well unless the detection accuracy reaches at least 70%.

Several improvements could help achieve better results.

First, the dataset used for training the model should include more examples of difficult situations. These could be images where the ball is partially hidden, moving fast, or blending with the field lines. Adding such examples would teach the model to handle complex cases more effectively.

Second, the model could be upgraded to use advanced detection techniques that are specifically designed to find small objects. These techniques focus more precisely on the ball, even when it is hard to see.

Another approach is to add information about the field. The system could identify fixed elements like field lines, penalty spots, and goal areas. By mapping these areas, it could ignore detections that do not match the size or shape of the ball, reducing errors.

Combining these solutions would make ball detection more reliable and allow tracking algorithms to work more accurately during football matches.

5.2.3 Tracking performance

Keeping track continuity is critical for reliable analysis. The system performs well in most cases, but issues such as ID changes and swapping highlight the need for more robust reidentification techniques. Using temporal consistency or integrating additional contextual information about player positions could further improve the reliability of the tracking.

5.2.4 Player team assignment

The system effectively assigns players to teams when the uniforms are easily distinguishable. However, future enhancements should address limitations with similar-colored uniforms and ID swaps during player collisions. Periodic verification of team assignments or the use of positional data relative to the field could help mitigate these issues.

5.3 Speed and distance metrics

The analysis of speed and distance reveals the valuable insights this system can provide, but also highlights areas that require further refinement to improve accuracy and applicability. The ability to calculate these metrics offers a powerful tool for understanding player performance, but its reliability is highly dependent on the quality of tracking data and environmental conditions.

In professional football settings, the use of standardized fields and clear reference points ensures that the system can deliver highly accurate measurements of speed and distance. The consistent field dimensions and markings allow for precise pixel-to-meter conversions, making it possible to trust the reported values. However, even in these settings, tracking errors—such as ID changes or temporary loss of tracking—can introduce some degree of error that needs to be addressed in future iterations of the system.

In nonprofessional settings, such as youth football matches, the challenges become more pronounced. The lack of standardized field dimensions and inconsistent markings reduces the reliability of pixel-to-meter conversions, which is critical to ensuring accurate calculations. Expanding the applicability of the system to these environments will require additional calibration steps or the development of new methodologies to estimate the pixel-to-meter ratios effectively. For example, incorporating manual measurements of field dimensions or leveraging data from GPS-equipped devices could help provide the necessary references for more accurate calculations.

Although the system's absolute measurements may face challenges in non-standard environments, its ability to track relative trends remains a strength. These relative metrics, such as comparisons of sprint frequency or total distance covered between players, offer meaningful insights even when exact values are less reliable. This suggests that the system can still be a valuable tool for performance analysis, particularly in contexts where relative comparisons are more important than precise absolute values.

Future work should prioritize addressing the identified challenges to improve the accuracy and versatility of the system. This includes improving tracking algorithms to minimize ID changes and data disruptions, as well as developing flexible methods for adapting the system to non-standard environments. By refining these aspects, the system can become a more reliable and widely applicable solution for analyzing speed and distance metrics across diverse football settings.

5.4 Results of the model to amateur football

The results highlight the importance of adapting the system to different recording perspectives. The low-angle recordings significantly reduced the model's ability to detect and track players,

the ball, and the referee. This limitation arises because the model was trained with tactical camera views, making it less effective when applied to perspectives common in grassroots football.

To improve performance in amateur football environments, a new dataset with labeled images from low-angle recordings is necessary. This dataset should include bounding boxes and annotations for all relevant entities, allowing the model to learn and generalize from non-standard perspectives. Fine-tuning the system with such data would enhance its adaptability and ensure more consistent results in these settings.

By addressing this limitation, the system could become more useful for grassroots football teams, providing them with advanced analysis capabilities that are currently more accessible to professional organizations. This improvement would make the system a valuable tool for player and team performance evaluation across different levels of football.

5.5 Summary of key findings

This project created and tested a system for tracking players and analyzing their performance. In this section, we review the main findings of this work:

- **Successful player detection:** The system was very accurate at detecting players, with a detection rate of 96%, which provides a strong base for further analyses like team assignment and performance metrics.
- **Unsatisfactory ball detection:** Detecting the ball was much more difficult, with only 36% accuracy, mainly because the ball is small, moves quickly, and can look like other elements on the field. Position-based adjustments helped reduce errors, but there is still room for improvement.
- **Effective team assignment:** The system was able to assign players to teams based on jersey colors effectively in most cases. Challenges appeared when team uniforms had similar colors or when tracking errors caused players to be misidentified.
- **Reliable player tracking with occasional disruptions:** Tracking players worked well overall, but issues like overlapping players or players re-entering the camera view sometimes disrupted the continuity of tracking. This occasionally affected calculations of trajectories and performance metrics, like the distance a player covered.
- **Dependence on standardized field dimensions:** Speed and distance metrics were more reliable in professional football matches where fields have standardized dimensions, making calibration easier. On smaller or irregular fields, like those in grassroots football, the lack of reliable calibration increased uncertainty in the measurements.

- **Challenges in amateur match analysis:** When testing the system on amateur football matches, such as those recorded from the AEC Manlleu Cadet A team, the results were less reliable. The model struggled with detecting players, the ball, and the referee due to the flatter camera perspective of the videos, which differed significantly from the tactical camera perspective used for training. These inconsistencies highlight the need to fine-tune the model with labeled images from matches recorded at similar low angles.

- **Potential for improvement:** Despite these challenges, the system has shown great potential for football analysis. Its high accuracy in detecting players and its ability to measure performance metrics provide a strong foundation. Addressing the issues with ball detection, tracking consistency, and adapting the model to various camera perspectives will make the system even more reliable and versatile for both professional and grassroots football.

6 Glossary

Below is a list of the most relevant terms and acronyms used in this report:

- **Bounding Box**: A rectangular box used in object detection to define the region in an image where an object is detected.
- **IoU (Intersection over Union)**: A metric used to evaluate the accuracy of object detection models, calculated as the overlap between the predicted and ground truth bounding boxes divided by their union.
- **Precision**: A metric that calculates the ratio of true positive detections to the total number of positive predictions.
- **Recall**: A metric that calculates the ratio of true positive detections to the total number of actual positives in the dataset.
- **mAP (Mean Average Precision)**: A performance metric used to evaluate object detection models, averaged across different Intersection over Union (IoU) thresholds.
- **Tracker**: A method or algorithm used to follow objects across video frames.
- **YOLO (You Only Look Once)**: A real-time object detection algorithm widely used in computer vision tasks.
- **ByteTrack**: A tracking algorithm that associates detected objects across frames in a video, improving the continuity of detections.
- **K-Means**: A clustering algorithm used to classify data points into distinct groups based on their features. In this project, it is used for color-based team classification.
- **Fine-tuning**: A process in machine learning where a pre-trained model is retrained on a new, typically smaller, dataset to specialize in a specific task.
- **Augmentation**: Techniques used to artificially expand a dataset by applying transformations like rotation, scaling, and flipping to the original images.
- **Ball Possession**: The attribution of the ball to a player based on proximity and predefined thresholds.
- **Camera Movement Estimation**: The process of determining the motion of the camera between consecutive frames in a video.

- **Perspective Transformation**: A method for converting 2D pixel coordinates into real-world coordinates using geometric transformations.
- **COCO (Common Objects in Context)**: A large-scale dataset for object detection, segmentation, and captioning, commonly used as a benchmark in computer vision.

Bibliography

Chepsi007 (2018). Full match — brazil v mexico — world cup 2018 — exclusive tactical camera hd 1080p —. .

Chepsi007 (2022). Full match — liverpool vs real madrid — final — exclusive tactical camera hd 1080p — 2022 —. YouTube.

Fang, Z. et al. (2018). Real-time object tracking and recognition in low-quality videos. *International Journal of Computer Vision*.

FigPii Team (2024). Heatmap analysis 101. FigPii Blog, https://www.figpii.com.

Gade, R. et al. (2019). Scalability challenges in object detection. *Journal of Sports Analytics*.

Garcia, A. et al. (2020). Optimizing warehouse operations with computer vision. *Logistics Management Journal*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Kinovea (2023). Kinovea: Open-source motion analysis software. Retrieved from https://www.kinovea.org.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*.

Li, Y. et al. (2021). Optimizing camera angles and lighting for accurate player tracking. *IEEE Transactions on Multimedia*.

Litjens, G. et al. (2017). A survey on deep learning in medical image analysis. *Medical Image Analysis*. Retrieved from https://www.semanticscholar.org/paper/A-survey-on-deep-learning-in-medical-image-analysis-Litjens-Kooi/6ff909c6fe089fc8ebfc64eca0f0c3cc34ba277f/figure/4.

McCarthy, J., Minsky, M. L., Rochester, N., and Shannon, C. E. (1956). A proposal for the dartmouth summer research project on artificial intelligence. *AI Magazine*.

Opta (2023). Opta sports data solutions. Retrieved from https://www.statsperform.com/opta/.

Picsellia (2022). Coco evaluation metrics explained. Accessed: 2024-12-19.

Pixellot (2023). Ai-powered sports production solutions. Retrieved from https://www.pixellot.tv/.

Redmon, J. et al. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*. Retrieved from https://arxiv.org/abs/1506.01497.

Roboflow (2024). football-players-detection dataset. https://universe.roboflow.com/roboflow-jvuqo/football-players-detection-3zvbc. visited on 2024-12-01.

Rosenfeld, A. (1969). Picture processing by computer. *ACM Computing Surveys*.

Second Spectrum (2023). Data-driven sports analytics. Retrieved from https://www.secondspectrum.com.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

StatsBomb (2023). Advanced football data and analysis. Retrieved from https://www.statsbomb.com.

TRACAB (2023). Sports tracking technology for elite football. Retrieved from https://www.tracab.com.

Ultralytics (2023). Yolo by ultralytics documentation. Retrieved from https://docs.ultralytics.com/.

Yang, J. et al. (2021). Computer vision in physical therapy: Applications and future directions. *Journal of Medical Robotics and Computer-Assisted Surgery*.

Zhang, X. et al. (2021). Bytetrack: An advanced multi-object tracking system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Zhu, Q. et al. (2021). Autonomous delivery systems and computer vision: A comprehensive review. *Journal of Logistics and Automation*.

7 Appendices

7.1 Annex 1: Conda environment configuration

The exact Conda environment configuration used in this project is provided below. This annex serves as a complete reference for replicating the setup.

Listing 1: Conda environment configuration

```
name: football_analysis_system
channels:
  - defaults
  - conda-forge
  - nvidia
  - pytorch
dependencies:
  - python=3.10.0
  - pip
  - cudatoolkit=11.3
  - cudnn=8.2.1
  - pytorch=2.4.0
  - torchvision=0.19.0
  - jupyter=1.0.0
  - ipykernel=6.29.5
  - pip:
    - ultralytics==8.3.34
    - opencv-python==4.10.0.84
    - roboflow==1.1.36
    - supervision==0.22.0
    - pandas==2.2.2
    - numpy==1.23.5
    - matplotlib==3.8.0
    - scikit-learn==1.5.1
```

7.2 Annex 2: Evaluation code for metrics

The Python code used to calculate metrics such as precision and recall for this project is provided below. This annex serves as a detailed reference for replicating the evaluation process.

```
1 def iou(boxA: Tuple[float, float, float, float], boxB: Tuple[float,
     float, float, float]) -> float:
2     xA = max(boxA[0], boxB[0])
```

```
    yA = max(boxA[1], boxB[1])
    xB = min(boxA[0] + boxA[2], boxB[0] + boxB[2])
    yB = min(boxA[1] + boxA[3], boxB[1] + boxB[3])
    interArea = max(0, xB - xA) * max(0, yB - yA)
    boxAArea = boxA[2] * boxA[3]
    boxBArea = boxB[2] * boxB[3]
    unionArea = boxAArea + boxBArea - interArea
    return interArea / unionArea if unionArea != 0 else 0

def evaluate(true_boxes: List[Tuple[float, float, float, float]],
   pred_boxes: List[Tuple[float, float, float, float]], iou_threshold:
    float = 0.5) -> Tuple[int, int, int]:
    tp, fp, fn = 0, 0, 0
    matched_pred_indices = set()
    for true_box in true_boxes:
        match_found = False
        for i, pred_box in enumerate(pred_boxes):
            if i in matched_pred_indices:
                continue
            if iou(true_box, pred_box) >= iou_threshold:
                tp += 1
                matched_pred_indices.add(i)
                match_found = True
                break
        if not match_found:
            fn += 1
    fp = len(pred_boxes) - len(matched_pred_indices)
    return tp, fp, fn

global_tp = {"ball": 0, "goalkeeper": 0, "player": 0, "referee": 0}
global_fp = {"ball": 0, "goalkeeper": 0, "player": 0, "referee": 0}
global_fn = {"ball": 0, "goalkeeper": 0, "player": 0, "referee": 0}

for image_file in os.listdir(image_dir):
    if not image_file.endswith(".jpg"):
        continue
    image_path = os.path.join(image_dir, image_file)
    label_path = os.path.join(label_dir, image_file.replace(".jpg", ".
       txt"))
```

```
39
40      image = Image.open(image_path)
41      img_width, img_height = image.size
42
43      true_boxes_ball, true_boxes_goalkeeper, true_boxes_player,
           true_boxes_referee = [], [], [], []
44      with open(label_path, 'r') as file:
45          for line in file:
46              label, x_center, y_center, width, height = map(float, line
                   .split())
47              bbox = (x_center * img_width, y_center * img_height, width
                    * img_width, height * img_height)
48              if label == 0:
49                  true_boxes_ball.append(bbox)
50              elif label == 1:
51                  true_boxes_goalkeeper.append(bbox)
52              elif label == 2:
53                  true_boxes_player.append(bbox)
54              elif label == 3:
55                  true_boxes_referee.append(bbox)
56
57      results = model(image_path)
58      predicted_boxes_ball, predicted_boxes_goalkeeper,
           predicted_boxes_player, predicted_boxes_referee = [], [], [],
           []
59      for box in results[0].boxes:
60          label = box.cls.item()
61          bbox = (box.xywh[0][0].item(), box.xywh[0][1].item(), box.xywh
                [0][2].item(), box.xywh[0][3].item())
62          if label == 0:
63              predicted_boxes_ball.append(bbox)
64          elif label == 1:
65              predicted_boxes_goalkeeper.append(bbox)
66          elif label == 2:
67              predicted_boxes_player.append(bbox)
68          elif label == 3:
69              predicted_boxes_referee.append(bbox)
70
71      for category, true_boxes, pred_boxes in zip(
```

```
        ["ball", "goalkeeper", "player", "referee"],
        [true_boxes_ball, true_boxes_goalkeeper, true_boxes_player,
           true_boxes_referee],
        [predicted_boxes_ball, predicted_boxes_goalkeeper,
           predicted_boxes_player, predicted_boxes_referee]
    ):
        tp, fp, fn = evaluate(true_boxes, pred_boxes)
        global_tp[category] += tp
        global_fp[category] += fp
        global_fn[category] += fn

for category in global_tp.keys():
    precision = global_tp[category] / (global_tp[category] + global_fp
       [category]) if (global_tp[category] + global_fp[category]) > 0
       else 0
    recall = global_tp[category] / (global_tp[category] + global_fn[
       category]) if (global_tp[category] + global_fn[category]) > 0
       else 0
    print(f"{category.capitalize()} Precision: {precision:.2f}")
    print(f"{category.capitalize()} Recall: {recall:.2f}")
```

Listing 2: Code for calculating precision and recall