A Major Project Report

On

TENNIS ANALYSIS SYSTEM WITH YOLO V5

Submitted to CMREC (UGC Autonomous)

In Partial Fulfilment of the requirements for the Award of Degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING (AI & ML)

Submitted By

MACHA PAVAN	**(228R1A66G0)**
MEDIPELLY TEJASWI	**(228R1A66G5)**
MYANA NIHANTH	**(228R1A66H0)**
SURA DHARANI	**(228R1A66J5)**

Under the Esteemed guidance of

Dr. A. PRAMOD KUMAR

Associate Professor,

Department of CSE(AI&ML).

Department of Computer Science and Engineering (AI & ML)

CMR ENGINEERING COLLEGE

(UGC AUTONOMOUS)

(Accredited by NAAC & NBA, Approved by AICTE, New Delhi, Affiliated to JNTU, Hyderabad)
(Kandlakoya, Medchal Road, Medchal-Malkajgiri Dist., Hyderabad-501 401)

(2025-2026)

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Accredited by NAAC&NBA, Approved by AICTE New Delhi, Affiliated to JNTU, Hyderabad, Kandlakoya, Medchal Road, Hyderabad-501 401)

Department of Computer Science & Engineering (AI & ML)

CERTIFICATE

This is to certify that the Major project entitled **"TENNIS ANALYSIS SYSTEM WITH YOLO V5"** is a bonafide work carried out by

MACHA PAVAN	**(228R1A66G0)**
MEDIPELLY TEJASWI	**(228R1A66G5)**
MYANA NIHANTH	**(228R1A66H0)**
SURA DHARANI	**(228R1A66J5)**

in partial fulfillment of the requirement for the award of the degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING (AI&ML) from CMR Engineering College, under our guidance and supervision.

The results presented in this Major project have been verified and are found to be satisfactory. The results embodied in this Major Project have not been submitted to any other university for the award of any other degree or diploma.

Internal Guide	**Major Project Coordinator**	**Head of the Department**
Dr. A. Pramod Kumar	Mr. G. Venkateswarlu	Dr. Madhavi Pingili
Associate Professor	Assistant Professor	Professor & HOD
Department of	Department of	Department of
CSE (AI & ML).	CSE (AI & ML).	CSE (AI & ML).

External Examiner:_____

DECLARATION

This is to certify that the work reported in the present Major project entitled "**TENNIS ANALYSIS SYSTEM WITH YOLO V5**" is a record of bonafide work done by us in the Department of Computer Science and Engineering (AI&ML), CMR Engineering College. The reports are based on the Major project work done entirely by us and not copied from any other source. We submit our Major project for further development by any interested students who share similar interests to improve the Major project in the future.

The results embodied in this Major project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

MACHA PAVAN (228R1A66G0)
MEDIPELLY TEJASWI (228R1A66G5)
MYANA NIHANTH (228R1A66H0)
SURA DHARANI (228R1A66J5)

ACKNOWLEDGEMENT

We are extremely grateful to **Dr. A. Srinivasula Reddy**, Principal and **Dr. Madhavi Pingili**, Professor & HOD, Department of CSE (AI&ML), CMR Engineering College for their constant support.

We are extremely thankful to **Dr. A. Pramod Kumar,** Associate Professor, Internal Guide, Department of CSE (AI&ML), for his constant guidance, encouragement and moral support throughout the Major project.

We will be failing in duty if we do not acknowledge with gratitude thanks to the authors of the references and other literature referred to in this Major Project.

We thank **Mr. G. Venkateswarlu,** Major Project Coordinator for his constant support in carrying out the Major project activities and reviews.

We express our thanks to all staff members and friends for all the help and co-ordination extended in bringing out this Major project successfully in time.

Finally, We are very much thankful to our parents who guided us for every step.

MACHA PAVAN (228R1A66G0)
MEDIPELLY TEJASWI (228R1A66G5)
MYANA NIHANTH (228R1A66H0)
SURA DHARANI (228R1A66J5)

CONTENTS

ABSTRACT

The field of tennis training and analysis has seen significant advancements with the application of machine learning and computer vision techniques. While the base paper introduced a Graph Convolutional Network (GCN) based model for human ac on recognition (HAR) using skeletal tracking, it exhibited key limitations, including high computational cost, reliance on skeletal data, and suboptimal real time performance. To overcome these challenges, we implemented a YOLO v5 based system that enhances player and ball detection, court key point extraction, and performance analysis in a video based tennis training system. Our approach replaces skeletal-based tracking with deep learning driven object detection, significantly improving real-time tracking accuracy and computational efficiency. The proposed system leverages fine-tuned YOLO models for detecting players and tennis balls while integrating court key point detection to map player movements onto a mini-court representation. This enables precise measurement of player speed, ball shot speed, and shot count, effectively addressing the limitations of the original GCN based approach. Additionally, by eliminating the need for 3D pose estimation and complex graph based processing, our method ensures higher scalability, ease of deployment, and adaptability to diverse video sources. Through extensive testing, our implementation demonstrates superior accuracy, reduced processing time, and enhanced feature extraction capabilities compared to the baseline model. This work provides a robust, real-me tennis ac on analysis system, making it a practical solution for training enhancement and performance evaluation.

Keywords :

YOLO v5, Player Detection, Ball Detection, Court Keypoint Extraction, Real-time Tracking, Deep Learning

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Introduction and Objectives

The rapid growth of Artificial Intelligence (AI) and computer vision has significantly transformed the field of sports analytics, particularly in tennis training and performance evaluation. Traditional coaching methods relied heavily on manual observation, video playback, and subjective analysis, which often lacked precision and consistency. However, with the advancement of deep learning and real-time video processing techniques, it is now possible to automatically analyze complex player movements, ball trajectories, and stroke patterns with high accuracy **[6], [16]**. These innovations have enabled the development of intelligent systems that provide data-driven insights, helping players and coaches make informed decisions to improve performance. As tennis becomes increasingly competitive, the demand for automated, reliable, and real-time analysis systems has grown substantially **[5], [9]**.

In recent years, Human Action Recognition (HAR) systems based on Graph Convolutional Networks (GCNs) have been introduced to analyze sports activities using skeletal joint data. These systems focus on tracking body keypoints and modeling their relationships over time to classify player actions **[18], [17]**. Although GCN based approaches show promising results in controlled environments, they suffer from several practical limitations. They require specialized hardware such as depth cameras, depend heavily on accurate skeletal detection, and struggle under real-world conditions such as occlusions, motion blur, varying lighting, and dynamic camera angles. Moreover, these methods primarily focus on action classification and fail to capture essential contextual information such as ball movement, player positioning, and court awareness, which are critical for comprehensive tennis analysis **[14], [16]**.

To overcome these challenges, this project proposes a robust tennis analysis system based entirely on YOLO v5 (You Only Look Once Version 5), a state of the art object detection algorithm known for its speed and accuracy. Unlike traditional pose based approaches, YOLO v5 performs direct object detection on video frames, enabling the system to identify players, the tennis ball, and important court features in a single forward pass **[2], [7]**. Its one stage detection architecture makes it highly suitable for real time applications, while its ability to detect small and fast-moving objects ensures reliable tracking of the tennis ball during high speed rallies. This shift from skeletal tracking to object detection significantly improves system efficiency and practical usability **[4], [13]**.

The proposed system integrates multiple components, including player detection, ball tracking, and court keypoint extraction, to create a complete analytical framework. By processing each frame of the input video, the system generates bounding boxes and positional data for detected objects. These detections are further used to compute meaningful performance metrics such as ball speed, player movement speed, distance covered, and shot count **[7], [9]**. Additionally, the system maps player and ball positions onto a virtual mini court representation, enabling spatial analysis of gameplay. This approach not only enhances the accuracy of performance evaluation but also provides intuitive visual feedback that can be easily understood by users **[3], [8]**.

The primary objective of this project is to design a scalable, efficient, and real-time tennis action analysis system that operates using standard video input without requiring specialized hardware. By leveraging YOLO v5, the system aims to deliver high detection accuracy, reduced computational complexity, and adaptability to diverse real-world environments **[2], [5]**. It seeks to bridge the gap between theoretical research and practical application by providing a user friendly solution for coaches, players, and sports analysts. Ultimately, this work contributes to the advancement of AI driven sports analytics by offering a reliable framework for automated tennis performance analysis and training enhancement **[6], [16]**.

1.2 Project Objectives

The primary objective of this project is to design and develop a tennis action analysis system using YOLO v5, capable of accurately detecting and tracking key elements such as players, the tennis ball, and court features directly from standard video input. By leveraging deep learning based object detection, the system eliminates the need for complex skeletal tracking and specialized hardware, making it more practical and adaptable for real world applications **[2], [7]**. The project aims to transform raw video data into meaningful insights by analyzing player movements, ball trajectories, and gameplay patterns in a structured and automated manner **[9], [12]**.

Another important objective is to compute and present performance metrics that can assist in improving player performance and coaching strategies. The system focuses on extracting quantitative data such as ball speed, player movement speed, and shot count through frame-by-frame analysis **[9], [20]**. Additionally, it generates annotated output videos with visual overlays including bounding boxes, trajectories, and mini court mapping, enabling users to easily interpret the results **[3], [8]**.

Objectives:

- To implement a YOLO v5 based detection system for identifying players, tennis balls, and court keypoints in real time video.
- To track player and ball movements across frames for continuous analysis.
- To calculate performance metrics such as ball speed, player speed, and shot count.
- To replace traditional GCN based skeletal models with a more efficient object detection approach.
- To generate annotated output videos with visual overlays and statistics.
- To provide a mini court visualization for better understanding of player positioning and movement.
- To ensure the system is scalable, efficient, and suitable for real-world tennis environments.

1.3 Purpose of the Project

The purpose of this project is to develop an efficient and real time tennis action analysis system using advanced computer vision techniques, with a primary focus on the YOLO v5 based object detection model. The system is designed to enhance the accuracy and effectiveness of tennis training by automating the analysis process, which was traditionally performed manually **[6], [5]**. By processing video input frame by frame, the system can identify and track key elements such as players, the tennis ball, and court features, enabling precise and consistent analysis. This reduces human error and provides a reliable platform for evaluating player performance in a structured manner **[7], [9]**.

Another key purpose of this project is to overcome the limitations of traditional skeletal-based systems, particularly those based on Graph Convolutional Networks (GCNs). These systems rely heavily on pose estimation and specialized hardware, which restricts their usability in real world scenarios **[16], [18]**. By replacing skeletal tracking with YOLO v5 based detection, the proposed system offers a more flexible and computationally efficient solution that works directly on standard video footage. This approach improves detection accuracy, especially in challenging conditions such as fast movements, occlusions, varying lighting, and different camera angles, making it suitable for real-time deployment **[13], [14]**.

Furthermore, the project aims to provide meaningful and actionable insights that can support data-driven coaching and performance improvement.

The system computes important metrics such as ball speed, player movement speed, distance covered, and shot count, which are essential for analyzing gameplay and identifying strengths and weaknesses **[9], [20]**. In addition, the generation of annotated output videos with visual overlays and mini-court mapping enhances the interpretability of results. Overall, the project seeks to create a practical, scalable, and user friendly solution that can be easily adapted to various tennis training environments, contributing to the advancement of AI based sports analytics **[3], [8]**.

1.4 Problem Statement

Traditional tennis training and analysis systems rely heavily on skeletal tracking techniques and Graph Convolutional Network (GCN) based human action recognition models, which depend on extracting body joint coordinates using specialized hardware such as depth cameras. While these approaches have shown promising results in controlled laboratory environments, they suffer from significant practical limitations when applied to real-world tennis scenarios **[17], [18]**.

These systems require high computational resources, complex preprocessing pipelines, and structured datasets to function effectively. Moreover, they are highly sensitive to environmental variations such as lighting conditions, camera angles, motion blur, and player occlusions. Fast-paced movements in tennis, including rapid swings and ball interactions, further reduce the accuracy of skeletal tracking. As a result, these systems often fail to deliver reliable performance in dynamic match environments and cannot generalize well across different playing conditions **[13], [14]**.

In addition to detection limitations, existing systems lack the capability to provide comprehensive performance analysis required for modern tennis training. They primarily focus on action classification rather than delivering meaningful metrics such as ball speed, player movement speed, shot count, trajectory analysis, and spatial positioning on the court. The absence of court awareness makes it difficult to analyze player strategies, positioning, and shot placement effectively **[5], [9]**.

Furthermore, reliance on expensive hardware and complex setup restricts accessibility and scalability for widespread use among players and coaches. Therefore, there is a strong need for an efficient, real time, and video based tennis analysis system that can operate using standard video input, accurately detect players and the ball, map movements onto the court, and generate actionable performance insights **[2], [7]**.

1.5 Existing System

The existing tennis analysis system is primarily based on Graph Convolutional Networks (GCNs), which are widely used for Human Action Recognition (HAR). In this approach, the system focuses on capturing the skeletal structure of players by detecting key body joints using depth sensors such as Kinect cameras. These joints are connected to form a graph structure, where each node represents a body joint and edges represent the relationships between them **[18], [17]**. The movement of these joints over time is processed using Spatio Temporal Graph Convolutional Networks (ST-GCN) to recognize different tennis actions such as forehand, backhand, or serve. This method performs well in controlled environments where accurate skeletal data is available and background conditions are stable **[16], [19]**.

However, despite its effectiveness in research settings, the GCN based system faces significant challenges when applied to real world tennis scenarios. It heavily depends on precise skeletal tracking, which is difficult to achieve in dynamic environments involving fast movements, occlusions, and varying camera angles **[13], [14]**. Additionally, the system lacks contextual understanding of the game, as it does not consider important elements such as the tennis ball or court layout. As a result, the existing system is not well suited for real time applications and fails to deliver practical insights required for modern tennis training and evaluation **[5], [16]**.

Disadvantages

- Strong dependency on skeletal tracking, which fails under occlusion, fast movements, and non ideal camera angles.
- Requires specialized hardware (e.g., Kinect sensors), making it less practical and costly for real-world use.
- High computational cost due to complex graph operations (GCN, adjacency matrices), reducing real time performance.
- Poor adaptability to real world videos due to lighting variations, motion blur, and camera angle changes.
- Lack of court awareness, preventing analysis of player positioning and spatial strategies.
- Inability to detect or analyze the tennis ball, limiting overall game understanding.
- Does not compute performance metrics such as ball speed, player speed, or shot count.

1.6 Proposed System

The proposed system is designed using a YOLO v5 based deep learning approach to overcome the limitations of traditional GCN based tennis analysis systems. The overall workflow of the system is illustrated in Fig. 1.1. The process begins when the user uploads a tennis video, which is processed by the Video Input Module to extract frames for analysis.

These frames are then passed to the YOLO v5 Detection Module, where players, the tennis ball, and court boundaries are detected in real time. Unlike skeletal tracking methods, this approach directly works on raw video frames, eliminating the need for specialized hardware such as depth cameras. This makes the system more flexible, cost effective, and suitable for real world tennis environments with varying lighting conditions and camera angles.

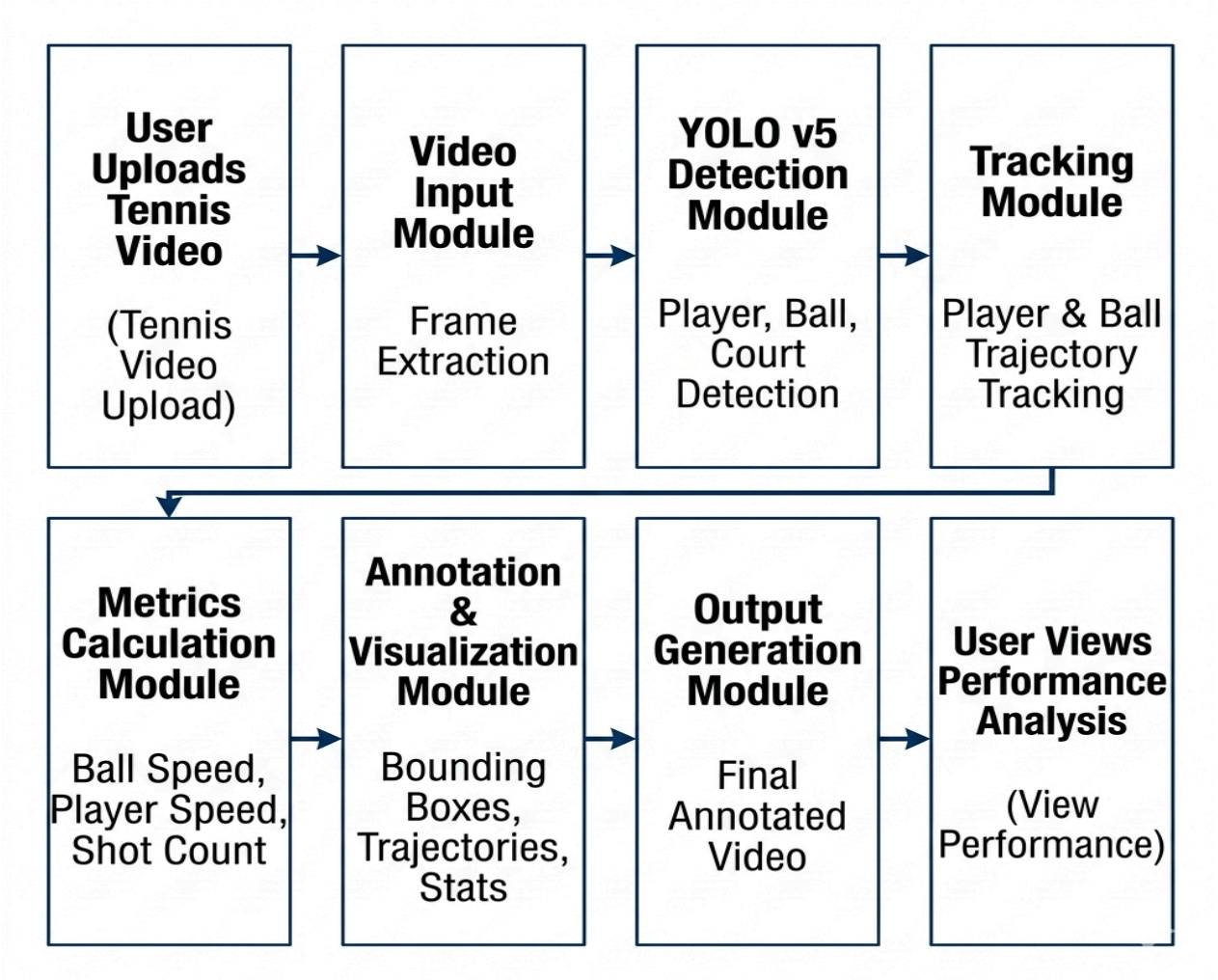

Fig. 1.1: Block Diagram of the Proposed YOLO v5 System

Following detection, the system utilizes a Tracking Module to maintain continuous tracking of players and the ball across frames, enabling accurate trajectory mapping. The tracked data is then processed in the Metrics Calculation Module, where important performance parameters such as ball speed, player movement speed, and shot count are computed. These calculations are based on frame-by-frame positional changes and time intervals, ensuring precise and meaningful analytics. The Annotation and Visualization Module overlays bounding boxes, trajectories, and statistical data onto the video frames, making the output visually informative and easy to interpret for coaches and players.

The Output Generation Module compiles the annotated frames into a final video, which is presented to the user for performance evaluation. The entire system is designed to operate efficiently in real time while maintaining high accuracy and scalability. By integrating detection, tracking, analysis, and visualization into a unified pipeline, the proposed system provides a comprehensive solution for tennis training and performance analysis. It not only improves detection accuracy but also enables deeper insights into gameplay for modern sports analytics.

Advantages

- The system operates in real time, ensuring faster processing compared to traditional GCN based models.
- It eliminates the need for skeletal tracking and specialized hardware, working effectively on standard video footage.
- YOLO v5 provides high detection accuracy for players, tennis balls, and court features, even under challenging conditions such as motion blur and occlusion.
- The inclusion of court keypoint detection introduces spatial awareness, enabling better analysis of player positioning and shot placement.
- The system generates detailed performance metrics such as ball speed, player speed, and shot count, supporting data-driven training decisions.
- The modular architecture ensures scalability, allowing future enhancements like advanced analytics, AI based coaching, and real time streaming.
- Improved reliability is achieved through a robust deep learning pipeline rather than dependence on a single traditional model.

1.7 Input and Output Design

1.7.1 Input Design

The input design acts as the primary interface between the user and the tennis analysis system, ensuring that raw data is collected, validated, and transformed into a structured format suitable for processing. In this system, the input mainly consists of tennis matches or training videos uploaded by the user through a simple and interactive interface. The Video Input Module is responsible for accepting various video formats such as MP4, AVI, and MOV, and converting them into sequential frames for further analysis. This process involves multiple preprocessing steps including frame extraction, resolution normalization, noise reduction, and format standardization to ensure compatibility with deep learning models like YOLO v5.

Additionally, techniques such as frame rate adjustment and image enhancement are applied to maintain consistency across different video sources. A well defined input design minimizes the chances of incorrect data entry, reduces preprocessing errors, and ensures smooth execution of subsequent modules such as detection, tracking, and performance analysis, there by improving the overall efficiency of the system.

In addition to handling video input, the system also incorporates user interaction parameters such as file selection, input validation, and real time system prompts to enhance usability. The interface is designed to be intuitive and user friendly, allowing users with minimal technical knowledge to upload and process videos. Advanced validation mechanisms are implemented to verify file type, resolution, duration, and size before processing begins, ensuring that only suitable inputs are accepted.

This prevents system failures, reduces processing delays, and enhances reliability. Error-handling mechanisms such as warning messages, status indicators, and guided prompts assist users in correcting mistakes during the input stage. Furthermore, the system ensures secure handling of uploaded data, maintaining data integrity and privacy throughout the process. Overall, the input design emphasizes accuracy, consistency, ease of use, and robustness, forming a strong foundation for reliable and efficient tennis performance analysis.

The proposed system focuses on converting user oriented data into a system usable format by transforming uploaded tennis videos into structured frame sequences suitable for processing. It ensures compatibility with deep learning models through proper format normalization, thereby reducing ambiguity and maintaining consistency in input data. In addition, the system provides a user friendly data entry interface that allows users to upload videos with minimal effort. It supports easy navigation and interaction for both beginners and advanced users, while also enabling smooth handling of large video files without introducing complexity.

Furthermore, the system emphasizes ensuring data validity and accuracy by validating video format, resolution, and file size before processing, and preventing invalid or corrupted inputs from entering the system. It incorporates prompts and notifications to guide users during input selection. To reduce processing errors and delays, the system eliminates redundant data and ensures clean input for faster processing, while minimizing preprocessing time through efficient frame extraction techniques. It also avoids system failures caused by improper input handling. Additionally, the system enhances security and data integrity by ensuring safe handling and storage of uploaded video data, protecting it from unauthorized access or modification, and maintaining integrity throughout the entire processing pipeline.

1.7.2 Output Design

The output design focuses on presenting the processed results of the tennis analysis system in a clear, structured, and meaningful manner. After processing the input video through detection, tracking, and analytics modules, the system generates an annotated video as the primary output. This video includes bounding boxes around players and the tennis ball, trajectory paths, and visual overlays of performance metrics such as speed and shot count. The design ensures that the output is easy to interpret, even for users without technical knowledge, by organizing information in a visually appealing and systematic format.

In addition to visual output, the system also provides analytical data in the form of performance statistics and summaries. These include player movement patterns, ball trajectory analysis, and key performance indicators such as average speed and total shots. The use of structured layouts, readable fonts, and color coded elements enhances clarity and reduces cognitive load while viewing the output.

The integration of mini court visualization further improves understanding by mapping player positions and ball movements in a simplified and standardized view. Furthermore, the output design emphasizes usability and decision making support. The annotated video and statistical insights help coaches and players identify strengths, weaknesses, and areas for improvement. Real-time or near real time feedback enables quick analysis during training sessions.

The system ensures that outputs are consistent, accurate, and adaptable for different use cases such as training evaluation, performance comparison, and match analysis. By transforming complex data into intuitive visual and numerical representations, the output design significantly enhances the effectiveness of the tennis analysis system.

2. LITERATURE SURVEY

1. **Y. Ding, Z. Fan, and Y. Zhao, "YOLO-Ball: Real-time Tennis Ball Detection under Occlusion and Motion Blur," Journal of Sports Engineering and Technology, 2026.**
   Proposes a YOLO based tennis ball detection model designed to handle challenges such as motion blur and occlusion. Achieves improved detection accuracy and robustness in high speed scenarios using attention mechanisms and multi scale feature extraction, but increased computational complexity and limited large-scale real-world validation remain concerns.

2. **M. D. Rafeeq and M. Mallikarjun, *"Assessment of Real-Time Object Identification Using Convolutional Neural Networks with YOLO V4,"* in Smart Computing Paradigms: Sustainable Computing, LNNS, vol. 1318, pp. 453–463, 2025.**
   Proposes a real time object identification system using YOLO v4 integrated with convolutional neural networks for improved detection accuracy and speed. The model demonstrates efficient performance in identifying multiple objects under varying conditions, making it suitable for real-time applications.

3. **X. Zhang and B. Li, "Tennis Ball Detection Based on YOLOv5 with TensorRT," *Scientific Reports*, vol. 15, pp. 21011–21023, 2025.**
   Proposes a YOLO v5 based tennis ball detection model optimized with TensorRT for faster inference. Achieves high detection accuracy and real-time performance, but evaluation under complex occlusion scenarios is limited.

4. **V. M. Desu and S. F. Ali, "Automated Tennis Player and Ball Tracking with Court Keypoints Detection (Hawk-Eye Simulation)," *arXiv preprint arXiv:2511.04126*, 2025.**
   Implements an automated tennis tracking system simulating Hawk Eye using player, ball, and court keypoint detection. Demonstrates precise tracking in controlled settings, but real world robustness under rapid player movement remains to be tested.

5. **S. Geetha et al., "High-Speed and Tiny Object Tracking in Racquet Sports Using Deep Learning," *Procedia Computer Science*, 2025.**
   Focuses on tracking small fast-moving objects like balls in racquet sports using deep learning models. Shows strong performance for high speed tracking, though deployment in real time live matches was not fully explored.

6. **A. Banoth, N. Hashmi, and R. Bokde, "A review of object detection empowering sports: Key technologies and future outlook," AI in Sports, 2025.**
   Provides a comprehensive review of object detection techniques applied in sports analytics, highlighting models like YOLO, SSD, and Faster R-CNN. Discusses applications across multiple sports and outlines future research directions, but lacks detailed experimental validation for specific sports scenarios.

7. **K. Fujii, "Computer vision and deep learning for professional sports analytics," in CV-Based Sports Analysis Handbook, Springer, 2025.**
   Explores the use of computer vision and deep learning techniques in professional sports analytics, including player tracking, motion analysis, and tactical evaluation. Presents a broad theoretical foundation, but offers limited implementation details for real time sports systems.

8. **R. Singh and M. Rao, "YOLO v5-based tennis ball and player tracking for performance analysis," IJMRSET, vol. 13, no. 2, pp. 88–96, 2025.**
   Introduces a YOLO v5 based system for detecting and tracking tennis players and balls to generate performance metrics. Demonstrates effective detection accuracy and tracking performance, but struggles with consistency in high-speed motion and occlusion scenarios.

9. **J. W. Chen, "Automated video-based analytics framework for tennis doubles," NUS Dissertation, 2025.**
   Proposes an automated video analytics framework tailored for tennis doubles matches, focusing on multi player tracking and interaction analysis. Achieves structured tactical insights, but requires further optimization for real time processing and scalability in live match conditions.

10. **S. C. Vanga, R. Vangari, and S. Vasre, *"Lane Line and Object Detection Using YOLO v3,"* International Journal of Innovative Science and Research (IJISR), vol. 9, no. 6, p. 2371, 2024.**
    Presents an object and lane line detection approach using YOLO v3, focusing on simultaneous detection tasks in dynamic environments. The model achieves reliable detection performance with good speed, serving as a foundation for real time systems, though its accuracy may be lower compared to newer YOLO versions when dealing with small scale or high speed objects.

Table 2.1: Literature Review Summary

Focused Area / Title	Key Findings	Reference
YOLO Ball for Tennis Ball Detection **[1]**	Proposes a YOLO based model optimized for detecting tennis balls under motion blur and occlusion. Improves detection accuracy using attention mechanisms and multi-scale feature extraction, enabling better performance in high-speed gameplay scenarios.	Y. Ding, Z. Fan, and Y. Zhao, "YOLO-Ball:Real-time Tennis Ball Detection under Occlusion and Motion Blur," Journal of Sports Engineering and Technology, 2026.
YOLO v4 based Real-Time Object Identification**[2]**	Proposes a real-time object identification system using YOLO v4 integrated with convolutional neural networks. Enhances detection speed and accuracy for multiple objects in dynamic environments, making it suitable for real-time applications.	M. D. Rafeeq and M. Mallikarjun, "Assessment of Real-Time-Object-Identificati on Using Convolutional Neural Networks with YOLO V4," in *Smart Computing Paradigms: Sustainable Computing*, LNNS, vol. 1318, pp. 453–463, 2025.
YOLO v5 with TensorRT Optimization **[3]**	Introduces a YOLO v5 based detection model optimized using TensorRT for faster inference. Achieves high detection accuracy and real-time performance, but limited evaluation under occlusion conditions.	X. Zhang and B. Li, "Tennis Ball Detection Based on YOLOv5 with TensorRT," *Scientific Reports*, vol. 15, pp. 21011–21023, 2025.
Automated Tracking with Court Keypoints **[4]**	Implements an automated system simulating Hawk-Eye using player, ball, and court keypoint detection. Provides accurate tracking in controlled environments but requires validation in real-world dynamic scenarios.	V. M. Desu and S. F. Ali, "Automated Tennis Player and Ball Tracking with Court Keypoints Detection (Hawk-Eye Simulation)," *arXiv preprint arXiv:2511.04126*, 2025.
High-Speed Object Tracking in Racquet Sports **[5]**	Focuses on detecting and tracking fast-moving small objects in racquet sports using deep learning techniques. Improves tracking precision but faces challenges with extreme speed variations and occlusions.	S. Geetha et al., "High-Speed and Tiny Object Tracking in Racquet Sports Using Deep Learning,"*Procedia Computer Science*, 2025.

Focused Area / Title	Key Findings	Reference
Object Detection in Sports Analytics Review **[6]**	Provides a comprehensive review of object detection technologies used in sports analytics. Highlights deep learning advancements and identifies challenges such as real-time processing and accuracy trade-offs.	A. Banoth, N. Hashmi, and R. Bokde, "A review of object detection empowering sports: Key technologies and future outlook," AI in Sports, 2025.
Deep Learning for Sports Analytics **[7]**	Discusses the application of computer vision and deep learning techniques in professional sports analytics. Emphasizes performance evaluation, player tracking, and tactical analysis using video data.	K. Fujii, "Computer vision and deep learning for professional sports analytics," in CV-Based Sports Analysis Handbook, Springer, 2025.
YOLOv5-based Tennis Tracking System **[8]**	Proposes a YOLO v5 based system for detecting and tracking tennis players and balls. Demonstrates improved accuracy and real-time performance for performance analysis applications.	R. Singh and M. Rao, "YOLO v5 based tennis ball and player tracking for performance analysis," IJMRSET, vol. 13, no. 2, pp. 88–96, 2025.
Video-based Analytics for Tennis Doubles **[9]**	Develops a video-based framework for analyzing tennis doubles matches. Focuses on player coordination, positioning, and movement patterns using automated detection techniques.	J. W. Chen, "Automated video-based analytics framework for tennis doubles," NUS Dissertation, 2025.
YOLO v3 based Lane and Object Detection**[10]**	Presents a detection framework using YOLO v3 for identifying lane lines and objects simultaneously in real-time scenarios. Demonstrates efficient detection performance with good processing speed, forming a baseline for real-time systems.	S. C. Vanga, R. Vangari, and S. Vasre, "Lane Line and Object Detection Using YOLO v3," *International Journal of Innovative Science and Research (IJISR)*, vol. 9, no. 6, p. 2371, 2024.

The reviewed literature highlights the growing importance of deep learning based object detection and tracking techniques in tennis analytics, with a strong emphasis on YOLO based models for real-time performance. Advanced approaches such as YOLO Ball and YOLO v5 with Tensor RT demonstrate significant improvements in detecting fast-moving tennis balls under challenging conditions like motion blur and occlusion, while other systems integrate player, ball, and court keypoint detection to simulate technologies like Hawk-Eye. Several studies extend beyond detection to include tracking, motion prediction, and tactical analysis, enabling deeper insights into player performance and gameplay strategies.

3. SOFTWARE REQUIREMENTS ANALYSIS

3.1 Modules and Their Functionalities

3.1.1 Data Analysis

Data analysis plays a crucial role in understanding the nature and structure of the input video datasets used for tennis match analysis and player tracking. The datasets consist of recorded tennis matches captured under different conditions, including variations in camera angles, lighting environments, court types, and video resolutions. These videos often include complex scenarios such as rapid player movements, ball speed variations, occlusions caused by players or the net, and changing backgrounds. The presence of multiple object classes such as players, tennis balls, rackets, and court boundaries adds to the complexity of the dataset. Additionally, challenges such as motion blur of the ball, varying player sizes due to camera perspective, and partial visibility of objects require careful consideration during the analysis phase.The dataset may contain noise and inconsistencies, including camera shake, shadows, audience interference, and fluctuating frame rates.

These factors can negatively impact the performance of detection and tracking models if not properly addressed. Therefore, detailed data analysis is essential to identify such issues and design strategies to handle them effectively. This understanding helps in selecting appropriate preprocessing techniques, tuning YOLO v5 detection parameters, and developing a robust tracking mechanism. By thoroughly analyzing the dataset characteristics, the system can be optimized to handle real-world complexities, ensuring reliable detection, accurate tracking, and consistent performance across different match scenarios.

3.1.2 Data Preprocessing

Data preprocessing is a critical step that transforms raw, unstructured video data into a clean and standardized format suitable for further analysis and model processing. The input video is first divided into individual frames, which serve as the basic units for object detection and tracking. During this stage, various noise reduction techniques are applied to eliminate disturbances such as camera shake, motion blur, lighting variations, and irrelevant background elements. Frame normalization processes, including resizing and aspect ratio adjustments, are performed to maintain consistency across all frames.

These steps ensure that the input data is uniform and compatible with the requirements of deep learning models. In addition to basic cleaning, advanced preprocessing techniques are applied to enhance the quality and usefulness of the data. Background subtraction and contrast enhancement improve the visibility of players and the tennis ball, making detection more accurate. Court detection algorithms and perspective transformation methods are used to identify court boundaries and map the playing area, providing a spatial reference for tracking movements. Frames are also sampled at consistent intervals to capture meaningful sequences while reducing redundancy. The final output of this stage is a set of optimized frames that are ready for YOLO v5 based object detection and tracking. Effective preprocessing not only improves model accuracy but also reduces computational complexity and enhances overall system performance.

3.1.3 Deep Learning Algorithm for Prediction

The proposed system utilizes a deep learning based object detection approach centered around the YOLO v5 (You Only Look Once version 5) architecture to achieve accurate and real-time detection of players and the tennis ball. YOLO v5 is selected due to its high processing speed, ability to detect small objects, and robustness in handling complex visual conditions. The model processes preprocessed video frames and identifies objects by generating bounding boxes along with confidence scores for each detected entity.

These objects include tennis players, the ball, and court features. The model's ability to perform detection in a single pass significantly reduces computation time, making it suitable for real-time applications.Once detection is completed, the results are passed to a tracking mechanism that ensures continuity of object identities across frames. Techniques such as Intersection over Union (IoU) matching and Kalman filtering are used to maintain consistent tracking of players and the ball, even during rapid movements or temporary occlusions.

This enables the system to generate smooth trajectories and accurate positional data over time. The architecture is designed to handle both single player and multi player scenarios, making it adaptable for different match formats such as singles and doubles. By combining efficient detection with reliable tracking, the deep learning module forms the core of the system, enabling precise analysis of movements, interactions, and overall tennis performance.

3.2 Functional Requirements

The functional requirements define the essential operations that the tennis analysis system must perform to achieve accurate and efficient performance evaluation. The system is designed to process user-uploaded video input, extract meaningful information, and generate analytical outputs that assist in understanding player performance.

It integrates multiple modules such as input processing, object detection, tracking, and visualization to ensure smooth workflow execution. These requirements ensure that the system performs core tasks such as identifying players and the tennis ball, analyzing their movement, and presenting results in an understandable format for users including coaches and players. Additionally, the system emphasizes real time or near real time processing to provide immediate feedback during training sessions. It is designed to be flexible and adaptable, allowing integration of additional features such as advanced analytics or AI based recommendations in the future.

The functional requirements also ensure that the output is both visually informative and analytically useful, enabling users to gain insights into gameplay, strategy, and performance metrics. Overall, these requirements define how effectively the system transforms raw video input into actionable tennis performance data.

1. The system shall accept video input and convert it into frames for analysis.
2. The system shall detect players, the tennis ball, and court keypoints using YOLO v5 based models.
3. The system shall track player and ball movements across frames to ensure continuity.
4. The system shall compute performance metrics such as speed, trajectory, and shot count.
5. The system shall generate annotated output videos with bounding boxes and visual overlays.

3.3 Non-Functional Requirements

The Non-Functional Requirements define the quality attributes and operational constraints of the tennis analysis system. These requirements focus on how the system performs rather than what it does, ensuring reliability, efficiency, and usability under various conditions. The system must handle different types of video inputs, including variations in resolution, lighting, and camera angles, while maintaining consistent performance. It should also ensure smooth execution without crashes or interruptions, even when processing large or complex video files.

Furthermore, the system is expected to provide fast processing with minimal latency, especially for real time or near real time applications. Scalability is an important aspect, allowing the system to adapt to different hardware configurations and future enhancements. Security and data integrity are also critical, ensuring that user uploaded videos are handled safely and confidentially. These requirements collectively ensure that the system is robust, efficient, and suitable for practical deployment in real world tennis training and analysis scenarios.

1. The system shall ensure high reliability and stable performance during video processing and detection.
2. The system shall provide scalable performance across different video qualities, resolutions, and camera angles.
3. The system shall maintain efficient processing with minimal latency, especially during realtime detection.
4. The system shall ensure secure handling of user video data while maintaining confidentiality and data integrity.

3.4 Feasibility Study

The feasibility of the project is analyzed in this phase, and a business proposal is put forth with a general plan for the project and cost estimates. During system analysis, the feasibility study ensures that the proposed system is viable and not a burden to the company. Understanding the major system requirements is essential for feasibility analysis.

Key Considerations in Feasibility Analysis :

- Economic Feasibility
- Technical Feasibility
- Social Feasibility

3.4.1 Economic Feasibility

A feasibility study was conducted to evaluate whether the Tennis Analysis System using YOLO v5 can be successfully developed and deployed within the available resources. The purpose of this analysis is to determine whether the proposed system is practical, cost effective, technically achievable, and beneficial for end users such as players, coaches, and sports academies. The study also ensures that the system is not a burden in terms of implementation, maintenance, or operational requirements.

3.4.2 Technical Feasibility

The project is technically feasible because the required technology resources such as Python, OpenCV, TensorFlow/PyTorch, and YOLO v5 are readily available and well-documented. The computing requirement for the model is moderate and can be met using GPU enabled systems or cloud platforms. The techniques involved in object detection, tracking, and court keypoint extraction are achievable with current deep learning standards. The system design ensures scalability, meaning additional functionalities like shot classification or pose-based action recognition can be integrated in the future without major structural changes.

3.4.3 Social Feasibility

From a social perspective, the system has strong feasibility because it provides direct benefits to the tennis community. It assists players and coaches in analyzing performance, movement patterns, and ball behavior without manual tagging. The system does not affect the normal gameplay or require athletes to wear sensors, making it convenient and user friendly. The output visualizations are easy to interpret and help in performance improvement, injury prevention, and training strategy development. Therefore, the system promotes positive acceptance among its intended users.

4. SYSTEM SPECIFICATIONS

4.1 Software Requirements

The software requirements specify the essential tools, libraries, and development environments needed to design and implement the YOLO v5 based Tennis Analysis System. Since the project involves object detection, video processing, tracking, and performance evaluation, it requires a software setup that can efficiently support both computer vision and deep learning tasks. A stable programming environment is necessary to ensure that all modules such as video input, frame extraction, object detection, trajectory tracking, and metric calculation work together without compatibility issues. The use of an appropriate software stack also helps in reducing development complexity and allows smooth integration of different functionalities into a single working system.

In addition to the programming environment, the project depends on several supporting libraries and frameworks that play an important role in implementation and testing. Computer vision libraries are required for handling video frames, preprocessing images, and generating annotated outputs, while deep learning frameworks are used to execute the YOLO v5 model for accurate detection of players, the tennis ball, and court features. The software environment must also support data analysis, result visualization, and debugging during the development process. Altogether, these tools ensure smooth execution, better scalability, easier maintenance, and compatibility for future enhancements such as real time analytics, advanced tracking, and deployment in practical tennis training applications.

- Operating System: Windows / Linux / macOS
- Programming Language: Python 3.x
- Core Libraries: OpenCV, PyTorch, NumPy, Pandas, Matplotlib
- Deep Learning Framework: YOLOv5 (PyTorch-based)
- Development Environment: VS Code / PyCharm / Jupyter Notebook
- Documentation Tools: MS Word / LaTeX

4.2 Hardware Requirements

The hardware requirements define the minimum computational resources necessary for developing and running the proposed Tennis Analysis System using YOLO v5. Since the project involves processing video frames, detecting players and the tennis ball, tracking movement, and generating performance metrics, the system requires a computer with sufficient processing capability and memory. During the initial stages of development, testing, and experimentation, standard computing hardware such as a laptop or desktop with moderate specifications is generally sufficient. This allows the system to perform basic operations such as frame extraction, preprocessing, and model testing without the need for highly advanced equipment.

However, for achieving faster execution and real time performance, especially when working with high resolution videos or larger datasets, GPU support becomes highly beneficial. A dedicated graphics processing unit can significantly improve the speed of YOLO v5 model inference and reduce the time required for video analysis and output generation. Adequate RAM and storage are also important for handling datasets, saving model files, and storing processed outputs efficiently. The recommended hardware configuration ensures that the system remains stable, scalable, and compatible with future requirements such as model optimization, extended training, and deployment in real-world tennis performance analysis environments.

- Processor: Intel Core i5/i7 or equivalent
- Memory (RAM): Minimum 8 GB (16 GB recommended)
- Graphics Processing Unit: NVIDIA GTX/RTX series (for real-time detection)
- Storage: 250 GB HDD/SSD or higher
- Display: Standard 14" or higher
- Optional: Camera/Video source and internet connectivity for model updates and dataset.

5. SOFTWARE DESIGN

5.1 System Architecture

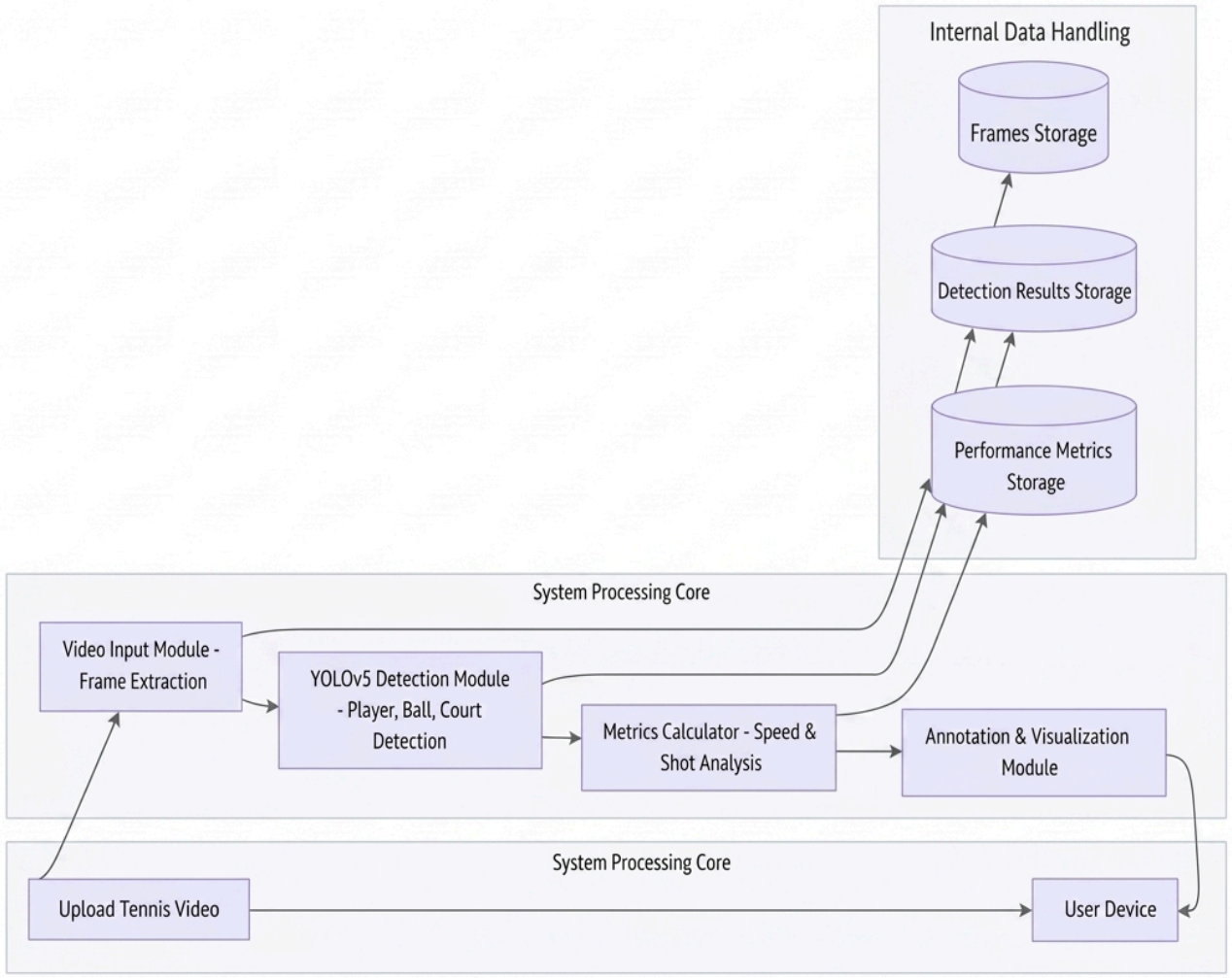

Fig. 5.1: System Architecture of YOLO v5

The System architecture of the YOLO v5 based Tennis Action Analysis System is designed as a structured sequence of interconnected modules that work together to process tennis match or training videos and generate an annotated, analytics rich output.The overall architecture of the system is illustrated in Fig. 5.1. The architecture follows a modular pipeline approach, where each stage is responsible for performing a specific task in the overall analysis process.This organized structure helps in managing the flow of data efficiently, from the moment a video is uploaded to the point where the final performance analysis is presented to the user.

The process begins when the User uploads a tennis video into the system through the input interface. This video is received by the Input Video Module, where it is prepared for analysis by extracting it into individual frames. These frames act as the fundamental input units for the next stage. The extracted frames are then passed into the YOLO v5 Detection Module, which performs object detection to identify key elements present in the tennis environment. These elements include tennis players, the tennis ball, and important court features such as court lines or keypoints. YOLO v5 is selected for this stage due to its high speed and strong detection accuracy, especially for small and fast moving objects like the tennis ball. The output of this module includes bounding box coordinates, object labels, and confidence scores for every detected object in each frame.

After detection, the identified objects are forwarded to the Tracking Module, which ensures continuity of player and ball movement across consecutive frames. This stage is essential because tennis involves rapid motion, sudden direction changes, and temporary occlusions, all of which require reliable tracking for accurate analysis. By preserving the identities and trajectories of players and the ball over time, the system can generate meaningful motion paths and positional data. The tracked information is then passed to the Metrics Calculation Module, which serves as the analytical core of the architecture. In this stage, important performance metrics such as ball speed, player running speed, shot count, and movement distance are computed. These values provide useful insights into player performance, movement efficiency, and gameplay dynamics, making the system highly relevant for sports training and analysis.

Once the performance metrics are generated, the system proceeds to the Annotation and Visualization Module, where the analytical information is overlaid onto the original video frames. This includes drawing bounding boxes around detected objects, plotting movement trajectories, and displaying numerical metrics such as speed values and shot counts. The annotated frames are then sent to the Output Generator, which combines them into a final processed video. This output is delivered back to the User in the form of a visually enhanced and data rich video, along with supporting metadata if required. The modular nature of this architecture ensures flexibility, scalability, and efficient execution. It also allows future enhancements such as real time streaming, multi player analysis, pose estimation, predictive analytics, and AI based coaching feedback to be incorporated without redesigning the entire system.

5.2 Dataflow Diagram

The Data Flow Diagram represents the complete flow of data within the Tennis Analysis System using YOLO v5. The overall data flow of the system is illustrated in Fig. 5.2. It begins with the user uploading a tennis video, which serves as the primary input to the system. Once the video is provided, it undergoes video acquisition and frame extraction, where the continuous video stream is converted into individual frames. This step ensures that each frame can be processed independently for accurate detection and analysis. Preprocessing techniques such as resizing and normalization may also be applied to improve detection performance and maintain consistency across frames. After frame extraction, the system applies YOLO v5 based detection to identify key objects such as players and the tennis ball in each frame. The detected objects are then passed to the object tracking module, which assigns unique IDs and maintains their continuity across consecutive frames. This tracking mechanism ensures that the movement of players and the ball can be followed throughout the video, even during fast paced rallies or partial occlusions.

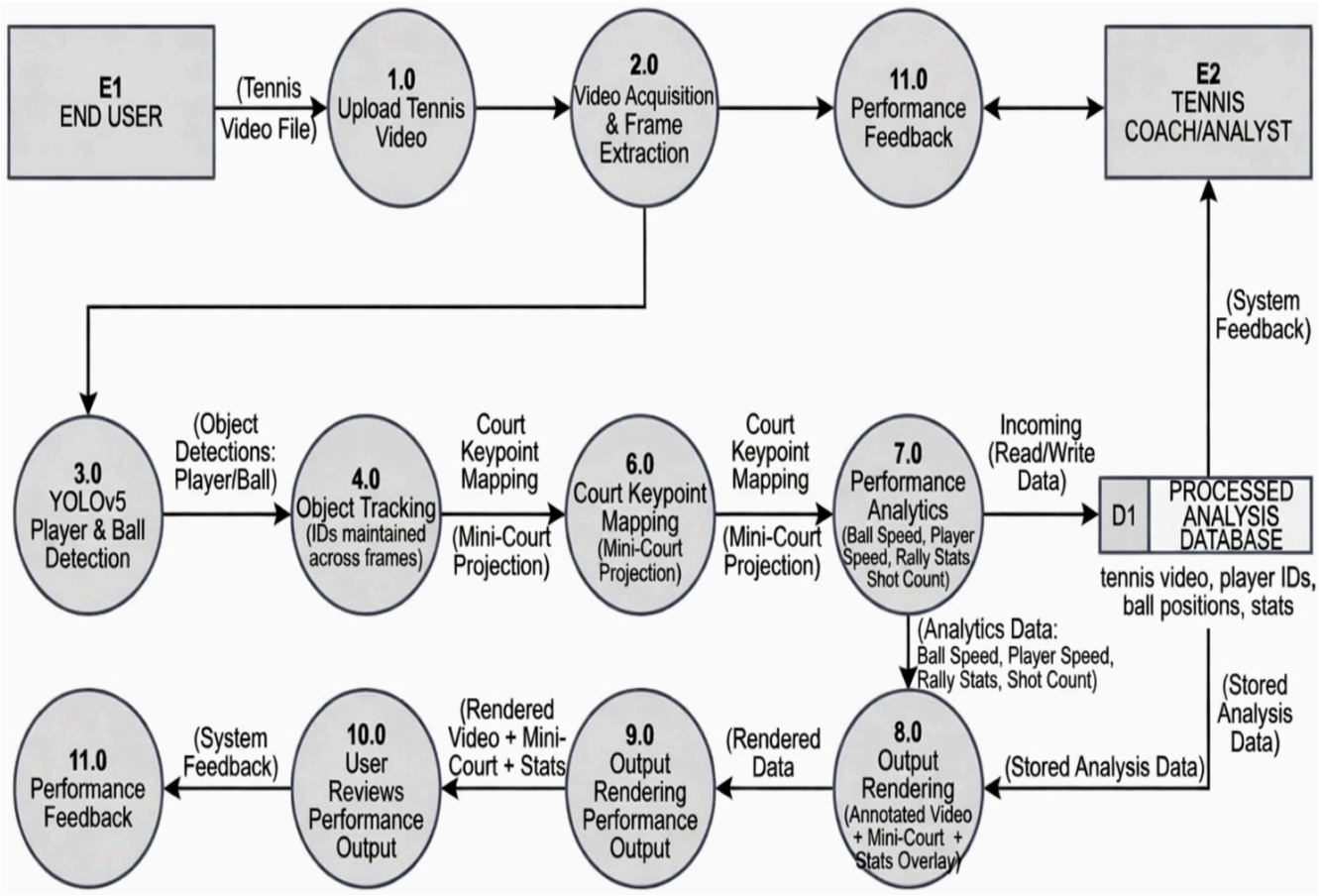

Fig. 5.2: Data Flow Diagram of Tennis Analysis System

Following tracking, the system performs court keypoint mapping to understand the spatial structure of the tennis court. By detecting important court lines and projecting them onto a mini court representation, the system can map player positions and ball trajectories in a standardized format. This spatial transformation enables more meaningful analysis of movement patterns and positioning, which is essential for performance evaluation.

Finally, the processed data is used in the performance analytics stage, where key metrics such as ball speed, player speed, rally statistics, and shot count are calculated. These insights are then visualized in the output rendering stage, where annotated video frames, mini court views, and statistical overlays are generated. The final output is presented to the user, allowing players and coaches to review performance, analyze gameplay, and make data driven improvements.

The transformed spatial data is used by the performance analytics module to compute metrics such as ball speed, player movement speed, rally count, and shot count. Finally, the output rendering module presents the results to the user through an annotated performance video containing statistics overlays and mini court visualization, enabling the user to evaluate gameplay and receive meaningful performance feedback.

5.3 UML Diagrams

UML (Unified Modeling Language) is a standardized visual modeling language used for specifying, visualizing, constructing, and documenting the components and behavior of software systems. It provides a clear and systematic way to represent how a system is designed and how its different parts interact with one another. UML was developed to simplify the process of software design by offering a common language that can be understood by developers, analysts, designers, and other stakeholders involved in the system development process. It plays an important role in software engineering because it helps transform abstract system ideas into structured diagrams that are easier to interpret, communicate, and implement. Created and standardized by the Object Management Group (OMG), UML 1.0 was formally proposed in January 1997, and since then it has become one of the most widely used modeling tools in software design.

UML is closely associated with object oriented analysis and design, making it especially useful for systems that are developed using modular and reusable software principles. It allows developers to describe both the structure and behavior of a system before actual coding begins, thereby reducing design errors and improving software quality. UML diagrams help in identifying the key elements of a system, their attributes, methods, relationships, and the way users interact with the application. In the case of the Tennis Analysis System using YOLO v5, UML can be used to represent the interactions between the user, input module, detection module, tracking system, analytics engine, and output generator. By visualizing these interactions and internal structures, developers can better understand system flow, dependencies, and implementation requirements.

UML diagrams are generally divided into two major categories: Behavioral Diagrams and Structural Diagrams. Behavioral UML diagrams focus on the dynamic aspects of the system, describing how the system behaves over time, how users interact with it, and how different components communicate during execution. Examples include Use Case Diagrams, Activity Diagrams, and Sequence Diagrams. In contrast, Structural UML diagrams represent the static architecture of the system by showing its classes, objects, components, and their relationships. Examples include Class Diagrams, Component Diagrams, and Deployment Diagrams. UML has been widely accepted as an industry standard because it provides a formal basis for understanding software design, supports effective communication among team members, and encourages the use of best practices in system development.

The Goals of UML are:

- To provide an expressive and standardized visual modeling language for designing software systems.
- To help developers represent system structure, behavior, and interactions clearly.
- To establish a formal basis for understanding and documenting software models.
- To simplify communication between developers, analysts, designers, and stakeholders.
- To support object oriented analysis and design principles effectively.
- To reduce system design complexity by breaking the software into manageable components.
- To assist in identifying system requirements, modules, and relationships before coding begins.
- To improve software quality by enabling better planning and error detection during design.
- To encourage the growth and use of object oriented tools and CASE tools.
- To provide reusable and scalable design models for future system enhancements.

Types of UML Diagrams:

1. Use Case Diagram.
2. Class Diagram.
3. Sequence Diagram.
4. Activity Diagram.

5.3.1 Use Case Diagram

The Use Case Diagram represents the interaction between the external actors and the YOLO v5 based Tennis Action Analysis System.The overall interaction of the system is illustrated in Fig. 5.3. It provides a high level view of how different users and system components participate in the tennis analysis process. In this diagram, the main actors include the System Admin, Video Source, Coach, and Player, each having a specific role in the operation of the system. The use case diagram helps in identifying the major functionalities provided by the system and shows how these functionalities are connected to the users who interact with them. It is especially useful in understanding the overall workflow of the proposed system before moving into detailed implementation.

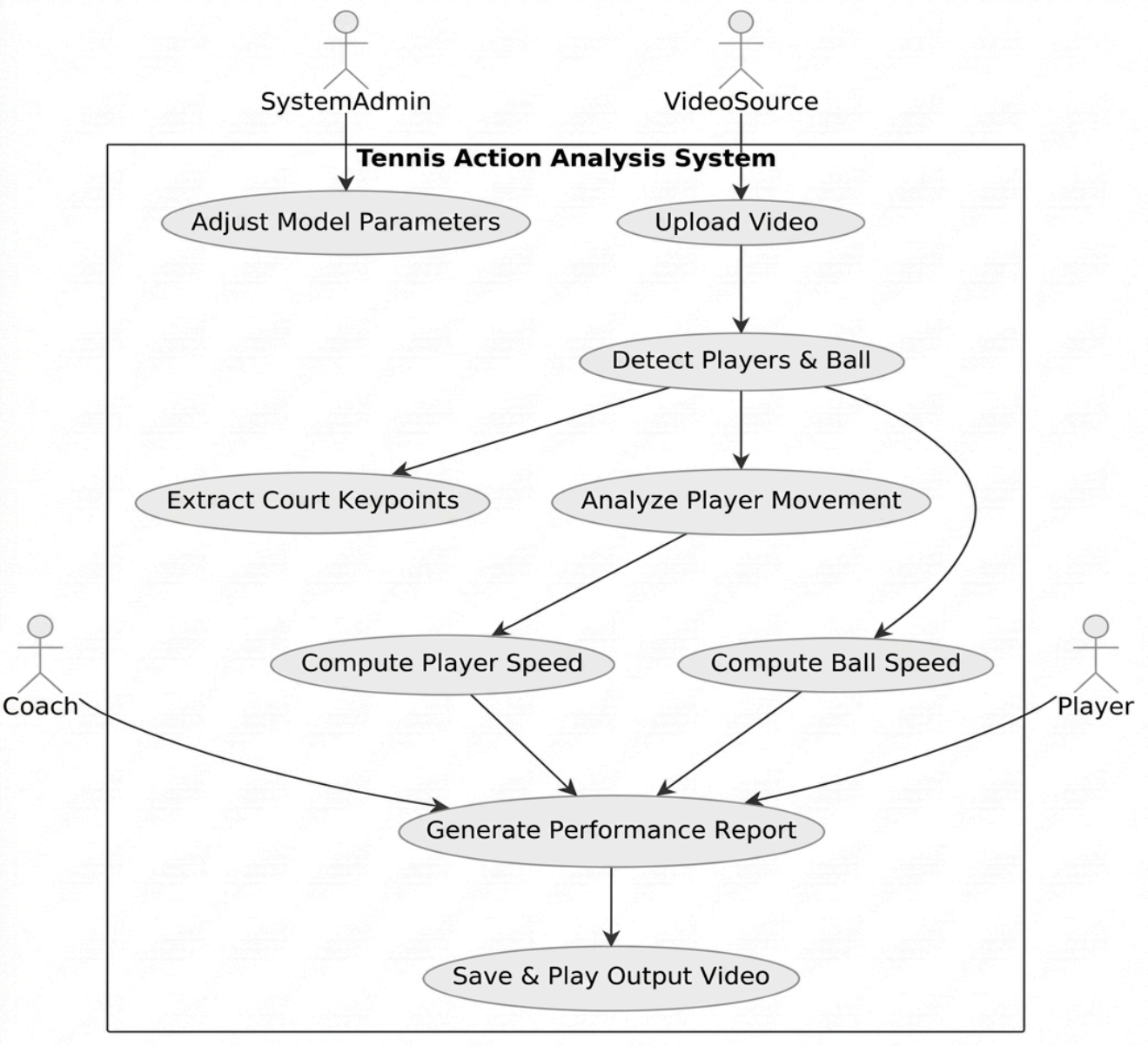

Fig. 5.3: Use Case Diagram of YOLO v5 System

The process begins when the Video Source uploads a tennis match or practice session video into the system through the Upload Video use case. Once the video is provided, the system automatically performs a sequence of actions including Detect Players & Ball, Extract Court Keypoints, and Analyze Player Movement. These functions form the core processing pipeline of the tennis action analysis system. Based on the detected and tracked objects, the system then computes important performance parameters such as Player Speed and Ball Speed. These computed values are then used to generate a Performance Report, which summarizes the player's movement and gameplay statistics.

The Coach and Player actors are directly associated with the Generate Performance Report use case, as they are the primary beneficiaries of the analytical output produced by the system. The System Admin is responsible for managing and fine tuning the system through the Adjust Model Parameters use case, ensuring that the detection and analysis models perform efficiently under different conditions. Overall, the use case diagram clearly illustrates how the proposed system supports video based tennis analysis through coordinated interaction between actors and functional modules.

5.3.1.1 Components of Use Case Diagram:

1. System Admin:

The System Admin manages the overall settings and technical configurations of the system. It ensures that the model performs correctly and efficiently.

2. Video Source:

The Video Source provides the tennis match or training video as input to the system. It acts as the starting point for the complete analysis process.

3. Coach:

The Coach uses the generated analysis results for training and performance evaluation. It helps in identifying player strengths and areas for improvement.

4. Player:

The Player views the processed output and performance metrics generated by the system. This helps in understanding personal gameplay and improving skills.

5. Upload Video:

The Upload Video use case allows the user to submit a tennis video into the system. It initiates the workflow for further processing and analysis.

6. Detect Players & Ball:

This use case identifies tennis players and the tennis ball from the input video frames. It is one of the main functions required for accurate analysis.

7. Extract Court Keypoints:

The Extract Court Keypoints use case detects important court boundaries and reference points. It helps the system understand spatial positioning during gameplay.

8. Analyze Player Movement:

This use case tracks the movement of players across the court during the match. It supports analysis of mobility, positioning, and player activity.

9. Compute Player Speed:

The Compute Player Speed use case calculates how fast the player moves on the court. It is useful for evaluating agility and movement performance.

10. Compute Ball Speed:

This use case measures the speed of the tennis ball during shots and rallies. It helps in analyzing shot power and gameplay intensity.

11. Generate Performance Report:

The Generate Performance Report use case creates a summary of player and ball analysis. It presents useful performance metrics in an understandable form.

12. Save & Play Output Video:

This use case stores the final annotated video after processing is completed. It allows the user to view the analyzed results visually.

5.3.2 Class Diagram

The Class Diagram represents the static structure of the proposed Tennis Analysis System using YOLO v5 and illustrates how the major classes of the system are organized and related to one another as shown in Fig. 5.4. It provides a clear view of the internal design of the application by showing the main classes, their responsibilities, and the methods associated with each one. In this system, the architecture is designed in a modular and pipeline oriented manner so that each class performs a specific task in the overall tennis video analysis process. This modular structure makes the system easier to understand, develop, maintain, and extend in the future.

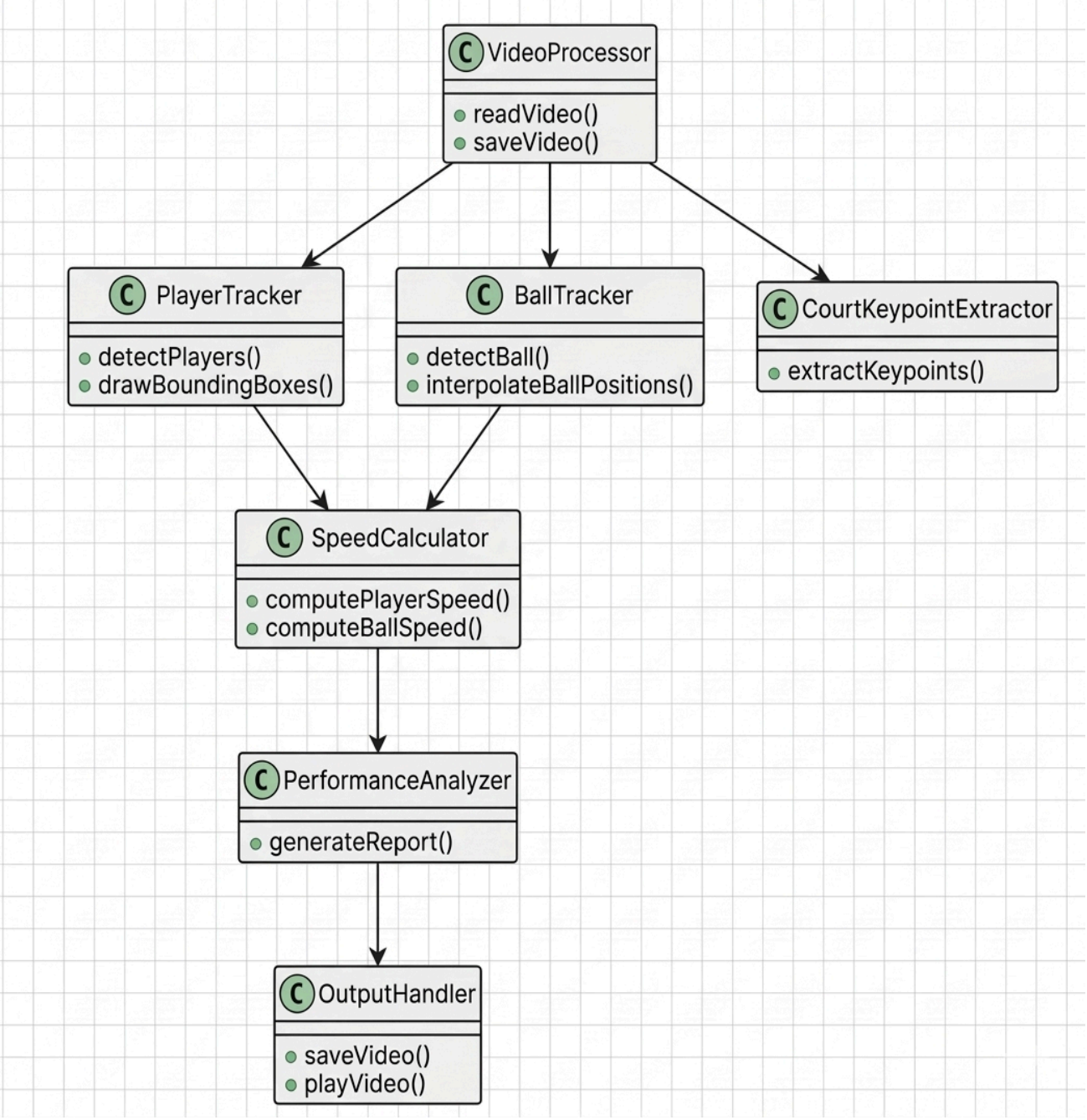

Fig. 5.4: Class Diagram of YOLO v5 System

The central controlling class in this diagram is the VideoProcessor, which acts as the entry point for video handling and connects the input video with the different analytical modules required for processing.The workflow begins with the VideoProcessor class, which is responsible for reading the input tennis video and later saving the processed output. From there, the extracted frames are distributed to the PlayerTracker, BallTracker, and CourtKeypointExtractor classes. The PlayerTracker identifies players and marks them using bounding boxes, while the BallTracker detects the tennis ball and estimates its position across frames, even when temporary detection gaps occur.

The CourtKeypointExtractor provides court reference points that help in understanding spatial movement on the court. The outputs of the player and ball tracking modules are then passed to the SpeedCalculator, which computes important motion based metrics such as player speed and ball speed. These values are then analyzed by the PerformanceAnalyzer, which generates a report of the player's performance.

5.3.2.1 Components of Class Diagram:

1. VideoProcessor:

The VideoProcessor is responsible for managing all video related input and output operations within the proposed tennis analysis system. It reads the uploaded tennis match or training video, extracts the necessary frames for analysis, and prepares them for processing by the detection and tracking modules. After all analytical operations are completed, it also helps in combining the processed frames and saving the final annotated output video.

2. PlayerTracker:

The PlayerTracker is used to detect and continuously track tennis players across the video frames throughout the analysis process. It identifies the players in each frame and marks them using visible bounding boxes so that their movement can be monitored clearly over time. This class helps maintain consistent player identification, even when players move quickly or change positions on the court. It is essential for analyzing player behavior, movement patterns, and court coverage during gameplay.

3. BallTracker:

The BallTracker is responsible for identifying and tracking the tennis ball as it moves across the court in the input video. Since the tennis ball is small and often moves at high speed, this class plays a critical role in ensuring accurate ball detection throughout the match. It also estimates or interpolates missing ball positions when the ball is temporarily lost due to motion blur, occlusion, or detection gaps. This makes the ball tracking process more reliable and useful for analyzing shot movement and ball trajectory.

4. CourtKeypointExtractor:

The CourtKeypointExtractor detects important court boundaries and structural reference points such as corners, service lines, and side lines. These keypoints provide a spatial understanding of the tennis court, allowing the system to interpret player and ball positions more meaningfully in relation to the actual playing area. This class helps the system understand where movement is occurring on the court and supports advanced analysis such as player positioning and tactical court coverage. It is an essential module for adding spatial intelligence to the system.

5. SpeedCalculator:

The SpeedCalculator is responsible for computing the movement speed of both tennis players and the tennis ball based on their tracked positions across consecutive video frames. By analyzing positional changes over time, it calculates useful motion-based metrics such as player running speed and ball shot speed. These values are important for understanding gameplay intensity, physical movement, and shot performance during the match. This class adds quantitative analytical value to the system and supports deeper performance evaluation.

6. PerformanceAnalyzer:

The PerformanceAnalyzer evaluates the data generated from tracking and speed calculation modules to produce meaningful insights about player and match performance. It interprets movement patterns, speed values, and other analytical information to generate performance related observations and summaries. This class plays a major role in transforming raw tracking data into useful evaluation outputs that can support coaching, training, and tactical decision making. It acts as the intelligence layer of the system by converting processed data into understandable analysis.

7. OutputHandler:

The OutputHandler manages the final output generated by the tennis analysis system after all processing and analysis tasks are completed. It is responsible for organizing the annotated video results, saving the final output file, and enabling playback for user viewing. This class ensures that the processed results are delivered in a clear and user friendly format, making them easy to review and interpret. It serves as the final stage of the system, connecting the analytical process with practical output presentation.

5.3.3 Sequence Diagram

The Sequence Diagram represents the dynamic interaction between different modules of the YOLO v5 based Tennis Analysis System in a time ordered manner as shown in Fig. 5.5. It shows how the user initiates the system by uploading a tennis video and how the internal components collaborate step by step to process the input and generate the final analytical output. Unlike structural diagrams that focus on system organization, the sequence diagram emphasizes the flow of messages and operations exchanged between modules during execution. This helps in understanding the chronological order of system behavior and clarifies how each module contributes to the overall tennis analysis process.

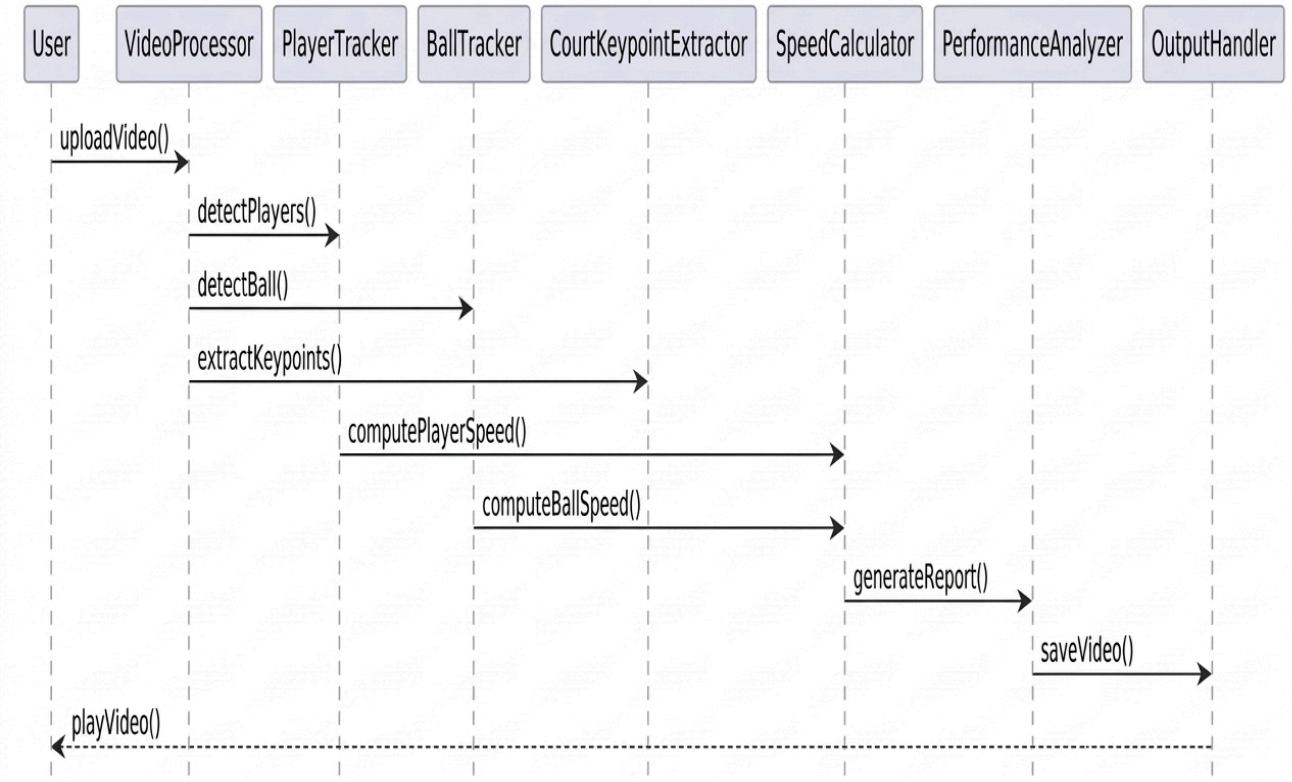

Fig. 5.5: Sequence Diagram of YOLO v5 System

The sequence begins with the User, who uploads a tennis video to the VideoProcessor module. This module is responsible for reading the input video, splitting it into frames, and preparing those frames for analysis. Once preprocessing is completed, the control is passed to the PlayerTracker and BallTracker modules, which identify and track the players and tennis ball across the video frames. At the same time, the CourtKeypointExtractor module detects the important court reference points such as court boundaries and key positions, which are essential for mapping movements and understanding player positioning. These modules work together to create a complete representation of the tennis environment.

After the tracking and court analysis stages are completed, the processed positional data is sent to the SpeedCalculator and PerformanceAnalyzer modules. The SpeedCalculator computes metrics such as ball speed, player running speed, and movement distance, while the PerformanceAnalyzer interprets this information to generate meaningful insights such as shot count, movement patterns, and overall player activity. Finally, the OutputHandler collects all processed results and creates the final annotated output video containing bounding boxes, trajectories, and statistical overlays. This sequence diagram clearly illustrates the logical communication flow of the system and helps in understanding how the proposed architecture transforms raw tennis video input into valuable performance analytics.

5.3.3.1 List of actions

- **User:** The user initiates the process by uploading a tennis match or training video into the system. After the analysis is completed, the user views the final processed output, which includes annotations, tracking information, and performance metrics for evaluation.

- **System:** The system receives the uploaded video and converts it into individual frames for analysis. These frames are then forwarded to the detection and tracking modules so that players, the tennis ball, and court related features can be identified and processed efficiently.

- **Model:** The model performs the core analysis by tracking the movement of players and the tennis ball across consecutive frames. Based on the tracked data, it calculates important metrics such as ball speed, player speed, movement trajectories, and shot count for performance assessment.

- **Evaluation:** The evaluation stage combines all processed information and generates the final annotated output video. This output includes visual overlays such as bounding boxes, trajectories, and performance statistics, which are then delivered back to the user for easy interpretation and analysis.

5.3.4 Activity Diagram

The Activity Diagram of the proposed Tennis Analysis System using YOLO v5 represents the complete workflow of the system in a step by step manner, showing how the application performs from the moment the user provides a tennis video until the final analytical output is generated as illustrated in Fig. 5.6. It is used to describe the dynamic behavior of the system and the sequence of activities carried out by various modules during execution.

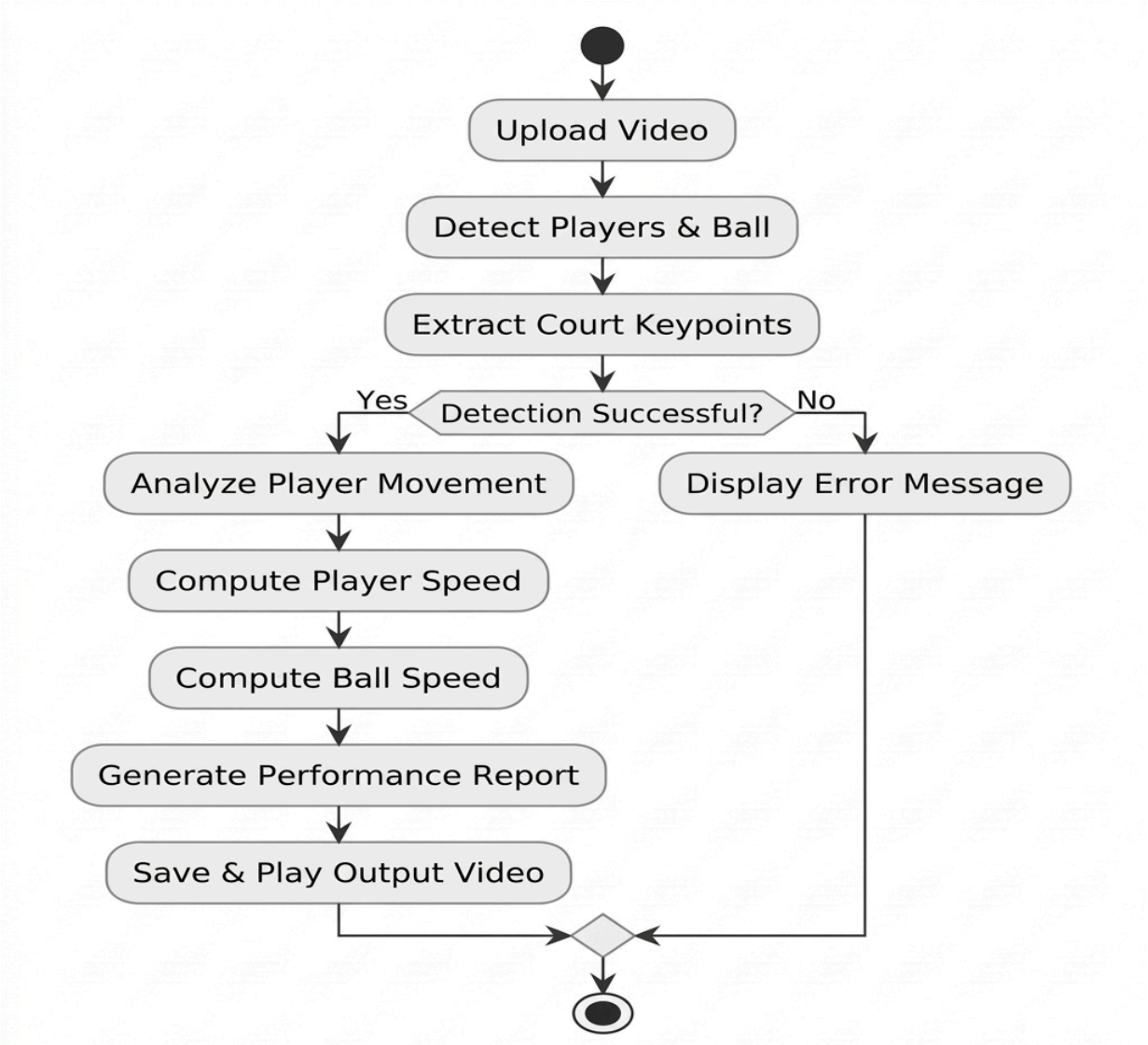

Fig. 5.6: Activity Diagram of YOLO v5 System

This diagram helps in understanding how data flows through the system and how each task is connected logically to the next. In the context of this project, the activity diagram highlights the internal operational flow of video input, preprocessing, object detection, tracking, speed calculation, visualization, and output generation in an organized and easy-to-understand form.

The workflow begins when the user uploads a tennis match or training video into the system. Once the video is received, the system starts processing it by extracting frames and preparing them for analysis. These frames are then passed to the YOLO v5 based detection module, which identifies key objects such as tennis players, the tennis ball, and relevant court elements. After the objects are detected, the system moves to the tracking stage, where the positions of players and the ball are continuously followed across multiple frames. At the same time, the court keypoint extraction module identifies important court boundaries and reference points, helping the system understand the spatial positions of players and ball movement within the actual tennis court layout.

After detection and tracking are completed, the system enters the analytical phase, where performance related metrics such as player speed, ball speed, movement trajectory, shot count, and positional behavior are calculated. These values are then passed to the annotation and visualization module, where the system overlays bounding boxes, labels, court markers, and performance statistics onto the video frames. Finally, the processed frames are combined and converted into a final annotated output video, which is then saved and displayed to the user. The activity diagram therefore gives a complete picture of how the proposed tennis analysis system functions internally, making it easier to understand the execution flow and interaction between major processing stages.

5.3.4.1 Activity Diagram Components

1. Start Node

The Start Node represents the beginning of the workflow in the proposed Tennis Analysis System using YOLO v5. It indicates the point at which the system is activated and the video analysis process is initiated. In the activity diagram, this node serves as the entry point from where all subsequent operations begin. It is important because it defines the starting state of the system before any input is provided or any processing takes place.

2. Upload Video

The Upload Video activity represents the step in which the user provides a tennis match or practice video to the system for analysis. This is the first major user driven action in the workflow and acts as the source of input data for the entire process. Once the video is uploaded, the system accepts it and prepares it for further operations such as frame extraction, detection, and tracking. This component is essential because the accuracy and usefulness of the final output depend heavily on the quality and format of the uploaded video.

3. Detect Players & Ball

The Detect Players & Ball activity represents the core object detection phase of the system, where the YOLO v5 model is used to identify important moving objects in the tennis video. In this stage, the system analyzes each extracted frame and detects the tennis players and the tennis ball by placing bounding boxes around them. This component is crucial because it forms the foundation for all later analysis, including tracking, speed calculation, and performance evaluation. Accurate detection at this stage directly affects the reliability of the entire system output.

4. Extract Court Keypoints

The Extract Court Keypoints activity is responsible for identifying important structural points of the tennis court, such as corners, boundaries, service lines, and center markings. These keypoints provide a spatial reference system that helps the system understand where the players and ball are positioned relative to the court layout. This stage adds contextual meaning to the detected movement data and allows for more advanced gameplay interpretation. Without court keypoint extraction, the system would only detect objects visually without understanding their tactical or positional relevance on the court.

5. Decision Node: Detection Successful?

The Decision Node labeled “Detection Successful?” is used to determine whether the player, ball, and court detection stages have been completed correctly. It acts as a control point in the workflow, allowing the system to check if the detection results are valid and sufficient for continuing to the next stage. If the detection is successful, the process moves forward toward tracking and analysis; otherwise, the system may reprocess or stop depending on the design. This component is important because it improves system reliability and ensures that incorrect or incomplete detections do not affect the final analytical output.

6. Tracking Process

The Tracking Process activity is responsible for following the movement of the detected tennis players and ball across multiple consecutive video frames. After initial detection, this stage ensures continuity by maintaining the identity and positional history of the objects throughout the match. It helps the system understand how players move on the court, how the ball travels during rallies, and how these movements evolve over time.

7. Performance Analysis

The Performance Analysis activity represents the stage where the system converts tracked motion data into useful performance related insights. At this point, the system calculates values such as player running speed, ball shot speed, trajectory, shot count, and positional behavior using the movement data collected across frames. This stage transforms raw object detection into meaningful sports analytics that can support coaching, player evaluation, and tactical review. It is one of the most important components in the activity diagram because it gives analytical value to the entire system.

8. Annotation / Visualization

The Annotation / Visualization activity is the stage where all detected and analyzed information is visually displayed on the video frames. In this component, the system overlays bounding boxes, labels, player IDs, ball IDs, court keypoints, trajectories, and performance metrics directly onto the original tennis video. This makes the output easier to interpret and more useful for users such as players, coaches, and analysts. The visualization process plays a significant role in making the system user-friendly by converting technical analysis into a clear and understandable visual form.

9. Output Generation

The Output Generation activity is responsible for producing the final processed result after all detection, tracking, analysis, and annotation tasks are completed. In this stage, the system combines all annotated frames into a continuous video output that contains both the original match footage and the added analytical overlays. This component ensures that the final output is presented in a structured and viewable format. It acts as the final technical stage before the processed video is delivered to the user.

10. Save / Display Result

The Save / Display Result activity represents the final user facing stage of the workflow, where the completed annotated tennis analysis video is either saved to the system or displayed for direct viewing. This allows the user to review the processed video and examine the analytical results generated by the system. It ensures that the final output is accessible, reviewable, and useful for future performance evaluation or documentation. This component is important because it completes the interaction between the system and the end user.

11. End Node

The End Node represents the completion of the entire activity flow in the proposed tennis analysis system. It indicates that all required processes: starting from video upload and ending with output generation have been successfully completed. This node marks the termination point of the workflow and confirms that the final result has been produced and delivered. In the activity diagram, it serves as the final state of the system after all operations have been executed in the correct sequence.

6. CODING AND IMPLEMENTATION

6.1 Source Code

court_line_detector.py

```
import torch
import torchvision.transforms as transforms
from torchvision.models import resnet50, ResNet50_Weights
import cv2
from torchvision import models
import numpy as np

class CourtLineDetector:
    def __init__(self, model_path):
        # self.model = models.resnet50(pretrained=True)
        self.model = models.resnet50(weights=ResNet50_Weights.IMAGENET1K_V1)
        self.model.fc = torch.nn.Linear(self.model.fc.in_features, 14*2)
        self.model.load_state_dict(torch.load(model_path, map_location='cpu'))
        self.transform = transforms.Compose([
            transforms.ToPILImage(),
            transforms.Resize((224, 224)),
            transforms.ToTensor(),
            transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
        ])

    def predict(self, image):
        image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        image_tensor = self.transform(image_rgb).unsqueeze(0)
        with torch.no_grad():
            outputs = self.model(image_tensor)
        keypoints = outputs.squeeze().cpu().numpy()
        original_h, original_w = image.shape[:2]
        keypoints[::2] *= original_w / 224.0
```

```
        keypoints[1::2] *= original_h / 224.0
        return keypoints

    def draw_keypoints(self, image, keypoints):
        # Plot keypoints on the image
        for i in range(0, len(keypoints), 2):
            x = int(keypoints[i])
            y = int(keypoints[i+1])
            cv2.putText(image, str(i//2), (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
            cv2.circle(image, (x, y), 5, (0, 0, 255), -1)
        return image

    def draw_keypoints_on_video(self, video_frames, keypoints):
        output_video_frames = []
        for frame in video_frames:
            frame = self.draw_keypoints(frame, keypoints)
            output_video_frames.append(frame)
        return output_video_frames
```

main.py

```
from utils import (read_video,
                   save_video,
                   measure_distance,
                   draw_player_stats,
                   convert_pixel_distance_to_meters
                   )

import constants
from trackers import PlayerTracker,BallTracker
from court_line_detector import CourtLineDetector
from mini_court import MiniCourt
import cv2
import pandas as pd
from copy import deepcopy
```

```
def main():
    # Read Video
    input_video_path = r"input_videos\input_video.mp4"
    video_frames = read_video(input_video_path)
    # Detect Players and Ball
    player_tracker = PlayerTracker(model_path='yolov8x')
    ball_tracker = BallTracker(model_path='models/yolo5_last.pt')

    player_detections = player_tracker.detect_frames(video_frames,read_from_stub=True,
                                                     stub_path="tracker_stubs/player_detections.pkl" )
    ball_detections = ball_tracker.detect_frames(video_frames, read_from_stub=True,
                                                      stub_path="tracker_stubs/ball_detections.pkl" )
    ball_detections = ball_tracker.interpolate_ball_positions(ball_detections)

    # Court Line Detector model
    court_model_path = "models/keypoints_model.pth"
    court_line_detector = CourtLineDetector(court_model_path)
    court_keypoints = court_line_detector.predict(video_frames[0])

    # Choose Players
    player_detections = player_tracker.choose_and_filter_players(court_keypoints, player_detections)

    # MiniCourt
    mini_court = MiniCourt(video_frames[0])

    # Detect ball shots
    ball_shot_frames= ball_tracker.get_ball_shot_frames(ball_detections)

# Convert positions to mini court positions
player_mini_court_detections,ball_mini_court_detections=mini_court.convert_bounding_boxes_to_
mini_court_coordinates(player_detections,ball_detections,court_keypoints)

    player_stats_data = [{
        'frame_num':0,
```

```
        'player_1_number_of_shots':0,
        'player_1_total_shot_speed':0,
        'player_1_last_shot_speed':0,
        'player_1_total_player_speed':0,
        'player_1_last_player_speed':0,

        'player_2_number_of_shots':0,
        'player_2_total_shot_speed':0,
        'player_2_last_shot_speed':0,
        'player_2_total_player_speed':0,
        'player_2_last_player_speed':0,
    } ]

    for ball_shot_ind in range(len(ball_shot_frames)-1):
        start_frame = ball_shot_frames[ball_shot_ind]
        end_frame = ball_shot_frames[ball_shot_ind+1]
        ball_shot_time_in_seconds = (end_frame-start_frame)/24 # 24fps
        # Get distance covered by the ball
   distance_covered_by_ball_pixels = measure_distance(ball_mini_court_detections[start_frame][1],
                                                      ball_mini_court_detections[end_frame][1])
distance_covered_by_ball_meters=convert_pixel_distance_to_meters(distance_covered_by_ball_pix
els, constants.DOUBLE_LINE_WIDTH, mini_court.get_width_of_mini_court())

        # Speed of the ball shot in km/h
        speed_of_ball_shot = distance_covered_by_ball_meters/ball_shot_time_in_seconds * 3.6

        # player who the ball
        player_positions = player_mini_court_detections[start_frame]
player_shot_ball=min(player_positions.keys(),key=lambdaplayer_id:
measure_distance(player_positions[player_id],ball_mini_court_detections[start_frame][1]))

        # opponent player speed
        opponent_player_id = 1 if player_shot_ball == 2 else 2
```

```
            distance_covered_by_opponent_pixels=    measure_distance(player_mini_court_detections
[start_frame][opponent_player_id], player_mini_court_detections[end_frame][opponent_player_id])

distance_covered_by_opponent_meters=convert_pixel_distance_to_meters(distance_covered_by_op
ponent_pixels, constants.DOUBLE_LINE_WIDTH, mini_court.get_width_of_mini_court())
      speed_of_opponent = distance_covered_by_opponent_meters/ball_shot_time_in_seconds * 3.6
       current_player_stats= deepcopy(player_stats_data[-1])
       current_player_stats['frame_num'] = start_frame
       current_player_stats[f'player_{player_shot_ball}_number_of_shots'] += 1
       current_player_stats[f'player_{player_shot_ball}_total_shot_speed'] += speed_of_ball_shot
       current_player_stats[f'player_{player_shot_ball}_last_shot_speed'] = speed_of_ball_shot
                              current_player_stats[f'player_{opponent_player_id}_total_player_speed']    +=
speed_of_opponent
       current_player_stats[f'player_{opponent_player_id}_last_player_speed'] = speed_of_opponent
       player_stats_data.append(current_player_stats)

   player_stats_data_df = pd.DataFrame(player_stats_data)
   frames_df = pd.DataFrame({'frame_num': list(range(len(video_frames)))})
   player_stats_data_df = pd.merge(frames_df, player_stats_data_df, on='frame_num', how='left')
   player_stats_data_df = player_stats_data_df.ffill()

player_stats_data_df['player_1_average_shot_speed']=player_stats_data_df['player_1_total_shot_spe
ed']/player_stats_data_df['player_1_number_of_shots']

player_stats_data_df['player_2_average_shot_speed']=player_stats_data_df['player_2_total_shot_spe
ed']/player_stats_data_df['player_2_number_of_shots']

player_stats_data_df['player_1_average_player_speed']=player_stats_data_df['player_1_total_player
_speed']/player_stats_data_df['player_2_number_of_shots']

player_stats_data_df['player_2_average_player_speed']=player_stats_data_df['player_2_total_player
_speed']/player_stats_data_df['player_1_number_of_shots']
```

```
# Draw output
## Draw Player Bounding Boxes
output_video_frames= player_tracker.draw_bboxes(video_frames, player_detections)
output_video_frames= ball_tracker.draw_bboxes(output_video_frames, ball_detections)
## Draw court Keypoints
    output_video_frames = court_line_detector.draw_keypoints_on_video(output_video_frames, court_keypoints)
# Draw Mini Court
output_video_frames = mini_court.draw_mini_court(output_video_frames)
output_video_frames=mini_court.draw_points_on_mini_court(output_video_frames,player_mini_court_detections)
output_video_frames=mini_court.draw_points_on_mini_court(output_video_frames,ball_mini_court_detections, color=(0,255,255))
# Draw Player Stats
output_video_frames = draw_player_stats(output_video_frames,player_stats_data_df)
## Draw frame number on top left corner
for i, frame in enumerate(output_video_frames):
    cv2.putText(frame, f"Frame: {i}",(10,10),cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
save_video(output_video_frames, "output_videos/output_video.avi")
# Define fixed width and height for video playback
FIXED_WIDTH = 1200  # Adjust as needed
FIXED_HEIGHT = 800  # Adjust as needed
# Play the saved output video
output_video_path = "output_videos/output_video.avi"
cap = cv2.VideoCapture(output_video_path)
if not cap.isOpened():
    print("Error: Unable to open the output video file.")
    return
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break  # Exit loop if video ends
```

```
        # Resize frame to the fixed width and height
frame_resized=cv2.resize(frame,(FIXED_WIDTH,FIXED_HEIGHT),
interpolation=cv2.INTER_AREA)
        # Display resized video frame
        cv2.imshow("Processed Tennis Analysis Video", frame_resized)
        # Press 'Q' key to exit the video playback and stop execution
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
if __name__ == "__main__":
    main()
```

mini_court.py

```
import cv2
import numpy as np
import sys
sys.path.append('../')
import constants
from utils import (
    convert_meters_to_pixel_distance,
    convert_pixel_distance_to_meters,
    get_foot_position,
    get_closest_keypoint_index,
    get_height_of_bbox,
    measure_xy_distance,
    get_center_of_bbox,
    measure_distance
)
class MiniCourt():
    def __init__(self,frame):
        self.drawing_rectangle_width = 250
```

```
        self.drawing_rectangle_height = 500
        self.buffer = 50
        self.padding_court=20
        self.set_canvas_background_box_position(frame)
        self.set_mini_court_position()
        self.set_court_drawing_key_points()
        self.set_court_lines()
    def convert_meters_to_pixels(self, meters):
        return convert_meters_to_pixel_distance(meters,
                                                constants.DOUBLE_LINE_WIDTH,
                                                self.court_drawing_width
                                            )
    def set_court_drawing_key_points(self):
        drawing_key_points = [0]*28
        # point 0
        drawing_key_points[0] , drawing_key_points[1] = int(self.court_start_x), int(self.court_start_y)
        # point 1
        drawing_key_points[2] , drawing_key_points[3] = int(self.court_end_x), int(self.court_start_y)
        # point 2
        drawing_key_points[4] = int(self.court_start_x)
drawing_key_points[5]=self.court_start_y+self.convert_meters_to_pixels(constants.HALF_COURT
_LINE_HEIGHT*2)
        # point 3
        drawing_key_points[6] = drawing_key_points[0] + self.court_drawing_width
        drawing_key_points[7] = drawing_key_points[5]
        # #point 4
drawing_key_points[8]=drawing_key_points[0]+self.convert_meters_to_pixels(constants.DOUBLE
_ALLY_DIFFERENCE)
        drawing_key_points[9] = drawing_key_points[1]
        # #point 5
drawing_key_points[10]=drawing_key_points[4]+self.convert_meters_to_pixels(constants.DOUBL
E_ALLY_DIFFERENCE)
        drawing_key_points[11] = drawing_key_points[5]
```

```
        # #point 6
drawing_key_points[12]=drawing_key_points[2]-self.convert_meters_to_pixels(constants.DOUBLE
_ALLY_DIFFERENCE)
drawing_key_points[13] = drawing_key_points[3]
        # #point 7
drawing_key_points[14]=drawing_key_points[6]-self.convert_meters_to_pixels(constants.DOUBLE
_ALLY_DIFFERENCE)
drawing_key_points[15] = drawing_key_points[7]
        # #point 8
        drawing_key_points[16] = drawing_key_points[8]
drawing_key_points[17]=drawing_key_points[9]+self.convert_meters_to_pixels(constants.NO_MA
NS_LAND_HEIGHT)
        # # #point 9
drawing_key_points[18]=drawing_key_points[16]+self.convert_meters_to_pixels(constants.SINGL
E_LINE_WIDTH)
        drawing_key_points[19] = drawing_key_points[17]
        # #point 10
        drawing_key_points[20] = drawing_key_points[10]
drawing_key_points[21]=drawing_key_points[11]-self.convert_meters_to_pixels(constants.NO_MA
NS_LAND_HEIGHT)
        # # #point 11
drawing_key_points[22]=drawing_key_points[20]+self.convert_meters_to_pixels(constants.SINGL
E_LINE_WIDTH)
        drawing_key_points[23] = drawing_key_points[21]
        # # #point 12
        drawing_key_points[24] = int((drawing_key_points[16] + drawing_key_points[18])/2)
        drawing_key_points[25] = drawing_key_points[17]
        # # #point 13
        drawing_key_points[26] = int((drawing_key_points[20] + drawing_key_points[22])/2)
        drawing_key_points[27] = drawing_key_points[21]
        self.drawing_key_points=drawing_key_points

    def set_court_lines(self):
        self.lines = [
```

```
        (0, 2),
        (4, 5),
        (6,7),
        (1,3),

        (0,1),
        (8,9),
        (10,11),
        (10,11),
        (2,3)]

def set_mini_court_position(self):
    self.court_start_x = self.start_x + self.padding_court
    self.court_start_y = self.start_y + self.padding_court
    self.court_end_x = self.end_x - self.padding_court
    self.court_end_y = self.end_y - self.padding_court
    self.court_drawing_width = self.court_end_x - self.court_start_x

def set_canvas_background_box_position(self,frame):
    frame= frame.copy()
    self.end_x = frame.shape[1] - self.buffer
    self.end_y = self.buffer + self.drawing_rectangle_height
    self.start_x = self.end_x - self.drawing_rectangle_width
    self.start_y = self.end_y - self.drawing_rectangle_height

def draw_court(self,frame):
    for i in range(0, len(self.drawing_key_points),2):
        x = int(self.drawing_key_points[i])
        y = int(self.drawing_key_points[i+1])
        cv2.circle(frame, (x,y),5, (0,0,255),-1)
    # draw Lines
    for line in self.lines:
start_point=(int(self.drawing_key_points[line[0]*2]), int(self.drawing_key_points[line[0]*2+1]))
end_point=(int(self.drawing_key_points[line[1]*2]), int(self.drawing_key_points[line[1]*2+1]))
```

```
            cv2.line(frame, start_point, end_point, (0, 0, 0), 2)
        # Draw net
net_start_point=(self.drawing_key_points[0],int((self.drawing_key_points[1]+self.drawing_key_poi
nts[5])/2))
net_end_point=(self.drawing_key_points[2],int((self.drawing_key_points[1]+self.drawing_key_poin
ts[5])/2))
        cv2.line(frame, net_start_point, net_end_point, (255, 0, 0), 2)
        return frame

    def draw_background_rectangle(self,frame):
        shapes = np.zeros_like(frame,np.uint8)
        # Draw the rectangle
            cv2.rectangle(shapes, (self.start_x, self.start_y), (self.end_x, self.end_y), (255, 255, 255),
cv2.FILLED)
        out = frame.copy()
        alpha=0.5
        mask = shapes.astype(bool)
        out[mask] = cv2.addWeighted(frame, alpha, shapes, 1 - alpha, 0)[mask]
        return out

    def draw_mini_court(self,frames):
        output_frames = []
        for frame in frames:
            frame = self.draw_background_rectangle(frame)
            frame = self.draw_court(frame)
            output_frames.append(frame)
        return output_frames

    def get_start_point_of_mini_court(self):
        return (self.court_start_x,self.court_start_y)
    def get_width_of_mini_court(self):
        return self.court_drawing_width
    def get_court_drawing_keypoints(self):
        return self.drawing_key_points
```

```
    def get_mini_court_coordinates(self,
                                   object_position,
                                   closest_key_point,
                                   closest_key_point_index,
                                   player_height_in_pixels,
                                   player_height_in_meters
                                   ):

distance_from_keypoint_x_pixels,distance_from_keypoint_y_pixels=measure_xy_distance(object_p
osition, closest_key_point)
        # Convert pixel distance to meters
distance_from_keypoint_x_meters=convert_pixel_distance_to_meters(distance_from_keypoint_x_p
ixels,player_height_in_meters,player_height_in_pixels)
distance_from_keypoint_y_meters=convert_pixel_distance_to_meters(distance_from_keypoint_y_p
ixels,player_height_in_meters, player_height_in_pixels)
        # Convert to mini court coordinates
mini_court_x_distance_pixels = self.convert_meters_to_pixels(distance_from_keypoint_x_meters)
mini_court_y_distance_pixels = self.convert_meters_to_pixels(distance_from_keypoint_y_meters)
      closest_mini_coourt_keypoint = ( self.drawing_key_points[closest_key_point_index*2],
                                       self.drawing_key_points[closest_key_point_index*2+1])
mini_court_player_position = (closest_mini_coourt_keypoint[0]+mini_court_x_distance_pixels,
                              closest_mini_coourt_keypoint[1]+mini_court_y_distance_pixels)
        return  mini_court_player_position


            def   convert_bounding_boxes_to_mini_court_coordinates(self,player_boxes,   ball_boxes,
original_court_key_points ):
        player_heights = {
           1: constants.PLAYER_1_HEIGHT_METERS,
           2: constants.PLAYER_2_HEIGHT_METERS
        }
        output_player_boxes= []
        output_ball_boxes= []
        for frame_num, player_bbox in enumerate(player_boxes):
           ball_box = ball_boxes[frame_num][1]
```

```
        ball_position = get_center_of_bbox(ball_box)
closest_player_id_to_ball=min(player_bbox.keys(),key=lambda  x:  measure_distance(ball_position,
get_center_of_bbox(player_bbox[x])))
        output_player_bboxes_dict = {}
        for player_id, bbox in player_bbox.items():
            foot_position = get_foot_position(bbox)
            # Get The closest keypoint in pixels
    closest_key_point_index = get_closest_keypoint_index(foot_position,original_court_key_points,
[0,2,12,13])
closest_key_point=(original_court_key_points[closest_key_point_index*2],original_court_key_poin
ts[closest_key_point_index*2+1])
            # Get Player height in pixels
            frame_index_min = max(0, frame_num-20)
            frame_index_max = min(len(player_boxes), frame_num+50)
            bboxes_heights_in_pixels = [get_height_of_bbox(player_boxes[i][player_id])
            for i in range    (frame_index_min,frame_index_max)]
            max_player_height_in_pixels = max(bboxes_heights_in_pixels)
            mini_court_player_position = self.get_mini_court_coordinates(foot_position,
                                                    closest_key_point,
                                                    closest_key_point_index,
                                                    max_player_height_in_pixels,
                                                    player_heights[player_id])
            output_player_bboxes_dict[player_id] = mini_court_player_position
            if closest_player_id_to_ball == player_id:
                # Get The closest keypoint in pixels
closest_key_point_index=get_closest_keypoint_index(ball_position,original_court_key_points,[0,2,
12,13])
                closest_key_point = (original_court_key_points[closest_key_point_index*2],
                            original_court_key_points[closest_key_point_index*2+1])
                mini_court_player_position = self.get_mini_court_coordinates(ball_position,
                                                    closest_key_point,
                                                    closest_key_point_index,
                                                    max_player_height_in_pixels,
                                                    player_heights[player_id])
```

```
                output_ball_boxes.append({1:mini_court_player_position})
            output_player_boxes.append(output_player_bboxes_dict)
        return output_player_boxes , output_ball_boxes
    def draw_points_on_mini_court(self,frames,positions, color=(0,255,0)):
        for frame_num, frame in enumerate(frames):
            for _, position in postions[frame_num].items():
                x,y = position
                x= int(x)
                y= int(y)
                cv2.circle(frame, (x,y), 5, color, -1)
        return frames
```

yolo_infer.py

```
import cv2
import torch
import time
import threading
from ultralytics import YOLO
# Load YOLOv8 model (optimized with GPU if available)
device = "cuda" if torch.cuda.is_available() else "cpu"
model = YOLO("yolov8x").to(device)
# Open the video file
video_path = "input_videos/ip-2.mp4"
cap = cv2.VideoCapture(video_path)
# Get video properties (width, height, FPS)
frame_width = int(cap.get(3))
frame_height = int(cap.get(4))
fps = int(cap.get(cv2.CAP_PROP_FPS))
# Define codec and create VideoWriter for saving the output
out = cv2.VideoWriter("output_video.mp4", cv2.VideoWriter_fourcc(*'mp4v'), fps, (frame_width,
frame_height))
# Global variable to hold the latest processed frame
latest_frame = None
latest_results = None
lock = threading.Lock()
```

```
def process_frames():
    """Background thread to process frames asynchronously for smooth performance."""
    global latest_frame, latest_results
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break
        with lock:
            latest_frame = frame.copy()

        # Run YOLO object tracking (faster than per-frame detection)
        results = model.track(frame, persist=True, conf=0.2, iou=0.4)
        with lock:
            latest_results = results
# Start the frame processing thread
processing_thread = threading.Thread(target=process_frames, daemon=True)
processing_thread.start()
prev_time = time.time()

while cap.isOpened():
    if latest_frame is None or latest_results is None:
        continue  # Wait until first frame is processed
    with lock:
        frame = latest_frame.copy()
        results = latest_results
    # Draw bounding boxes on the frame
    for r in results:
        for box in r.boxes:
            x1, y1, x2, y2 = map(int, box.xyxy[0])  # Get bounding box coordinates
            conf = box.conf[0]  # Confidence score
            cls = int(box.cls[0])  # Class ID
            # Draw the bounding box
            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)  # Green box
            label = f"{model.names[cls]} {conf:.2f}"
```

```
            # Put label text on the frame
            cv2.putText(frame, label, (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
    # Calculate FPS
    curr_time = time.time()
    fps_text = f"FPS: {1 / (curr_time - prev_time):.2f}"
    prev_time = curr_time
    # Display FPS on the frame
    cv2.putText(frame, fps_text, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2)
    # Show the frame
    cv2.imshow("Smooth YOLO Detection", frame)
    # Save the frame to output video
    out.write(frame)
    # Press 'q' to exit
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
# Release video and close windows
cap.release()
out.release()
cv2.destroyAllWindows()
```

yolo_inference.py

```
from ultralytics import YOLO
model = YOLO('yolov8x')
result = model.track('input_videos/input_video.mp4',conf=0.2, save=True)
# print(result)
# print("boxes:")
# for box in result[0].boxes:
# print(box)
```

6.2 Implementation

6.2.1 Front End Implementation:

The front end implementation of the proposed Tennis Analysis System using YOLO v5 is designed to provide a simple, interactive, and user friendly environment through which users can easily access the system's functionalities. Since the system is intended for users such as coaches, players, analysts, and researchers, the interface must be intuitive enough for both technical and non technical users. The front end serves as the visual communication layer between the user and the back-end processing modules, allowing smooth interaction with the video analysis system. It is responsible for presenting the input options, displaying processing results, and organizing analytical outputs in a clear and understandable format. A well designed front end improves the usability of the system and ensures that complex deep learning outputs are transformed into accessible visual information.

The interface is developed using standard web technologies such as HTML, CSS, JavaScript, and Flask, where Flask is used as the web framework to connect the user interface with the underlying Python based processing modules. Through this interface, users can upload tennis matches or training videos, initiate the analysis process, and observe the system's output without directly interacting with the internal implementation details. The uploaded video is displayed on the interface, and once processing begins, the front end visually presents the detection results generated by the YOLO v5 model. These results include bounding boxes around players and the tennis ball, as well as highlighted court boundaries or keypoints, enabling the user to visually understand how the system interprets the match environment frame by frame.

In addition to video visualization, the front end also supports the presentation of performance metrics and analytical insights in an organized manner. Important values such as player speed, ball speed, shot count, movement trajectory, and positional patterns can be displayed through labels, charts, tables, or dashboard panels. This makes the analysis more informative and easier to interpret for end users. The interface can also be designed to support responsive layouts so that the system remains accessible across desktop, laptop, and mobile devices. Overall, the front end implementation plays a crucial role in improving user experience by making the tennis analysis system not only technically functional but also visually effective, interactive, and suitable for practical real-world usage.

6.2.2 Backend Implementation:

The backend implementation acts as the computational core of the proposed Tennis Analysis System using YOLO v5, where all major processing, decision-making, and analytical operations are carried out. While the front-end provides interaction and visualization, the backend is responsible for managing the complete workflow that transforms a raw tennis video into a fully analyzed and annotated output. It handles the communication between user requests, video processing modules, object detection models, and performance analysis components. This layer ensures that uploaded videos are received properly, processed in the correct order, and returned with meaningful outputs. Because the system involves real-time or near real time analysis, the backend is designed to be efficient, scalable, and capable of handling computationally intensive tasks smoothly.

The backend is implemented primarily using Python, which is well-suited for artificial intelligence, computer vision, and data analysis applications. It integrates YOLO v5 as the main object detection model to identify players, tennis balls, and relevant court elements from the video frames. OpenCV is used for frame extraction, image preprocessing, and video manipulation, while NumPy and Pandas are used for numerical operations, coordinate handling, and performance related computations. After a user uploads a tennis video, the backend processes the input frame by frame and passes each frame to the detection model. The detected outputs are then forwarded to tracking and analytical modules, where the movements of players and the ball are monitored continuously. A court keypoint extraction algorithm is also integrated into the backend so that detected positions can be mapped accurately onto the court layout or a virtual mini court representation.

Once the detection and tracking processes are completed, the backend stores the extracted results in a structured format for further retrieval, analysis, and visualization. These results may include object coordinates, movement trajectories, speed calculations, shot count, and other performance metrics that are useful for tennis analysis. To enable communication between the user interface and the analytical engine, the backend uses Flask APIs, which handle requests such as video upload, model execution, and output delivery. The use of API based design also improves flexibility and supports future integration with web dashboards, mobile applications, or third party analytics tools. In addition, asynchronous or background processing can be used to improve response time and avoid blocking user interactions during long video analysis tasks.

6.2.3 Deployment and Reliability:

The complete Tennis Analysis System using YOLO v5 can be deployed in a cloud-based or local computing environment depending on the project requirements and available resources. Cloud platforms such as Google Colab, AWS EC2, or similar GPU enabled environments are highly suitable for this project because they provide the computational power needed to run deep learning models efficiently. These platforms support faster model inference, large scale video processing, and easy access to GPU acceleration, which is particularly important for real time or high resolution tennis video analysis. Model files, processed videos, analytical reports, and log data can also be stored securely in cloud storage, making the system more accessible, scalable, and manageable for future improvements or multi-user access.

Reliability is an important aspect of the system because accurate detection and stable execution are essential for meaningful performance analysis. To ensure dependable operation, the system can include exception handling mechanisms, input validation checks, and monitoring procedures to detect errors during video upload, model execution, or output generation. Performance consistency can be maintained through regular testing, validation on different types of tennis videos, and monitoring of model behavior under varying lighting, camera angles, and motion conditions. In addition, the modular architecture of the system improves maintainability and supports future upgrades, such as replacing YOLO v5 with newer models, improving court detection, or expanding the analytical dashboard. This makes the system not only reliable in its current form but also adaptable for long term development and deployment.

6.2.4 Conclusion:

The implementation successfully integrates deep learning, computer vision, and interactive visualization into a unified system for automated tennis action analysis. The combination of YOLO based detection, robust backend processing, and scalable cloud deployment ensures high accuracy, real time feedback, and operational reliability. This modular architecture lays a strong foundation for future enhancements like motion based skill assessment, AI assisted training recommendations, and integration with wearable sensor data for a more holistic performance analysis solution.

7. SYSTEM TESTING

System testing is one of the most important phases in the Software Development Life Cycle (SDLC), as it ensures that the complete and integrated software system functions correctly according to the specified requirements. It is generally performed after integration testing has been completed and before the system is delivered for user acceptance testing. At this stage, the application is treated as a finished product, and all its modules are tested together in a unified environment. The main purpose of system testing is to verify whether the developed software behaves as expected in real usage conditions and whether it can reliably perform all intended operations without failure. This phase plays a crucial role in identifying any functional gaps, performance issues, or unexpected behavior before the system is made available to end users.

In system testing, the software is evaluated as a complete system rather than testing its individual modules separately. This means that all major components such as: input handling, processing logic, output generation, and user interaction are tested together to check whether they work in coordination. The testing process focuses on verifying system level characteristics such as functionality, performance, reliability, compatibility, and security. Testers observe how the software behaves under normal, boundary, and exceptional conditions, ensuring that the interactions between different modules do not lead to errors or inconsistencies. By examining the application from an end to end perspective, system testing helps confirm that the software not only satisfies technical design specifications but also aligns with user expectations and operational requirements.

In the context of the Tennis Analysis System using YOLO v5, System testing is especially important because the project involves the integration of multiple technologies such as video processing, object detection, tracking, court analysis, performance computation, and output visualization. The system must be tested to ensure that uploaded tennis videos are correctly processed, players and balls are accurately detected, performance metrics are calculated properly, and the final annotated output is generated without errors. It is also necessary to evaluate how the system performs under different video qualities, lighting conditions, player movements, and camera angles. Through effective system testing, the reliability and stability of the proposed tennis analysis system can be improved, ensuring that it operates accurately and consistently in practical real world environments.

7.1 Types of System Testing

7.1.1 Unit Testing:

Unit testing focuses on testing the smallest individual parts or modules of a software application separately to ensure that each one performs its intended function correctly. In the proposed Tennis Analysis System using YOLO v5,this may include testing modules such as video frame extraction, player detection, ball detection, speed calculation, and court keypoint extraction independently. Each function or method is checked with predefined inputs to verify whether the expected output is produced. This type of testing is generally carried out during the early stages of development and is mainly performed by developers to identify coding errors before the modules are integrated.

The main purpose of unit testing is to ensure that every component behaves reliably in isolation and forms a strong base for the complete application. By testing individual units separately, developers can quickly detect logical errors, incorrect outputs, or failures in specific functions without interference from other modules. In a project like tennis analysis, where several independent modules work together, unit testing helps guarantee that each core function is accurate and stable. As a result, it improves code quality, simplifies debugging, and reduces the chances of larger system-level failures during later stages of testing.

7.1.2 Integration Testing:

Integration testing is performed after unit testing to verify whether multiple modules of the system work together correctly when combined. In the proposed tennis analysis system, this involves checking the interaction between components such as the video processing module, YOLO v5 detection model, tracking module, speed calculator, and output generator. Even if each unit functions properly on its own, errors may still occur when these modules exchange data or depend on one another. Integration testing ensures that the communication and coordination between these components are smooth and error-free.

This type of testing is especially important for systems that rely on a sequence of interconnected operations, as in the case of sports video analysis. For example, if the object detection module fails to send proper coordinates to the tracking module, the final analysis will become inaccurate. Integration testing helps identify issues such as data format mismatches, communication failures, incorrect parameter passing, or processing interruptions between modules.

7.1.3 Functional Testing:

Functional testing is carried out to ensure that the system performs according to the functional requirements defined in the project documentation. This testing focuses on validating whether each feature of the application behaves correctly when given specific inputs. In the tennis analysis system, functional testing may involve checking whether the system accepts video uploads, detects players and the tennis ball correctly, computes performance metrics, and generates annotated output videos as expected. It verifies that the system's features operate properly from the user's perspective and satisfy the intended project objectives.

During functional testing, both valid and invalid inputs are provided to examine how the system responds under different conditions. For example, testers may upload supported and unsupported video formats, provide low quality or high resolution videos, or test videos with different court angles and lighting conditions. The output is then compared with the expected behavior to confirm correctness. This type of testing is important because it ensures that the application delivers the required functions in a practical and user-oriented manner, making the system more dependable and effective for real-world usage.

7.1.4 System Testing :

System testing evaluates the complete and fully integrated software system to ensure that it satisfies the specified requirements as a whole. Unlike unit or integration testing, which focus on individual modules or module interactions, system testing checks the behavior of the entire application in a realistic environment. In the proposed tennis analysis system, this means verifying that all modules: from video upload and frame extraction to object detection, tracking, analytics, and output generation work together correctly without failures. It is an essential stage in confirming that the system is ready for practical use.

The purpose of system testing is to validate the overall functionality, performance, reliability, and behavior of the application under different operating conditions. Testers observe how the system behaves when processing various tennis videos and whether the output meets the expected results. This includes checking for smooth video processing, accurate detection results, correct speed calculations, and proper output generation. By testing the software as a complete system, this method helps ensure that the developed solution is stable, usable, and capable of handling real world scenarios effectively.

7.1.5 White Box Testing:

White box testing is a software testing technique in which the internal code structure, logic, and implementation of the application are examined in detail. In this type of testing, the tester has full knowledge of the source code and tests the software by analyzing control flow, conditions, loops, and internal algorithms. In the proposed tennis analysis system, white box testing can be applied to internal functions such as frame extraction logic, YOLO v5 prediction flow, speed calculation formulas, and data handling procedures. This type of testing is generally performed by developers or technically skilled testers who understand the internal working of the program.

The main goal of white box testing is to identify logical errors, unreachable code, improper conditions, and hidden defects that may not be visible through normal user level testing. It helps ensure that all possible execution paths are tested and that the internal logic of the software is correct and efficient. In a system involving computer vision and analytics, this testing is useful for verifying that algorithms process data correctly and that each function behaves as intended under different scenarios. White box testing therefore improves code quality, supports better debugging, and contributes to the overall reliability of the software.

7.1.6 Black Box testing:

Black box testing is a testing method in which the software is tested without any knowledge of its internal code or implementation details. The tester focuses only on the system's inputs and outputs to determine whether it behaves as expected. In the proposed tennis analysis system, black box testing may involve uploading a tennis video, checking whether the system detects players and the ball correctly, and verifying whether the final annotated output is generated accurately. This approach evaluates the software purely from the user's point of view.

The main purpose of black box testing is to ensure that the application meets its functional requirements and provides correct results under different conditions. Testers can supply a variety of inputs such as: different video resolutions, camera angles, or unsupported file types and observe how the system responds. If the outputs match the expected behavior, the system is considered functionally correct. This testing method is particularly useful because it reflects real world usage and confirms whether the application is practical, user friendly, and reliable from an end user perspective.

7.2 Testing Strategies

A structured testing strategy was followed throughout the development lifecycle of the proposed Tennis Analysis System using YOLO v5 to ensure that each component of the system functioned correctly and contributed effectively to the final output. The testing process was carried out in a phased and systematic manner, beginning with the validation of individual modules such as video preprocessing, frame extraction, object detection, and tracking. At this stage, each module was tested independently to confirm that it produced the expected results under different input conditions. For example, the preprocessing module was checked for proper frame extraction and normalization, while the detection module was tested to ensure accurate identification of players, tennis balls, and court-related features. This early-stage testing helped in identifying and correcting functional errors before the modules were integrated into the complete system.

After validating the individual modules, the testing strategy gradually progressed toward integration testing and full system evaluation, where all components were tested together as a unified pipeline. This phase focused on verifying whether the modules interacted smoothly and whether the complete system could process tennis videos accurately from input to output. Special attention was given to the reliability of tracking, the correctness of speed and performance metric calculations, and the quality of the final annotated output video. The system was also evaluated under different conditions such as varying video resolutions, lighting environments, player movements, and camera angles to ensure robustness and adaptability. By following this layered testing strategy, the project achieved better stability, reduced the risk of unexpected failures, and improved the overall confidence in the practical usability of the tennis analysis system.

7.2.1 Test Strategy and Approach:

The testing strategy for the proposed Tennis Analysis System using YOLO v5 was designed to ensure that the system performs accurately, efficiently, and reliably under different operating conditions. Both manual verification and automated evaluation techniques were used to validate the model's detection performance, video processing behavior, and overall system functionality. Different tennis match videos and sample image datasets were used to examine how effectively the system handled frame preprocessing, player and ball detection, and real time processing. The strategy mainly focused on confirming that the complete workflow from video input to YOLO v5 inference and output generation worked smoothly and produced correct analytical results.

The primary strategic objectives included:

- Verifying correct preprocessing of video frames before model inference.
- Ensuring accurate detection of tennis players and tennis balls using YOLO v5.
- Validating bounding box predictions and confidence scores.
- Ensuring smooth processing of video frames without delays .
- Verifying correct integration between the video input module and the YOLO v5 detection model.

7.2.2 Test Objectives:

The testing objectives were defined to ensure that all major functionalities of the proposed system worked correctly and met the expected performance requirements. These objectives focused on validating the correctness of video frame extraction, object detection accuracy, output visualization, and real time processing capability. Since the system is designed for practical tennis video analysis, testing was also aimed at checking whether the model performed consistently under different match conditions such as varying lighting, movement speed, and camera perspectives. These objectives helped guide the evaluation process and ensured that the system delivered useful and dependable results.

The following objectives guided the testing process:

- Ensure video frames are correctly extracted and processed.
- Verify that YOLO v5 accurately detects players and tennis balls in each frame.
- Confirm correct generation of bounding boxes around detected objects.
- Ensure the detection system works under different lighting and motion conditions.
- Validate that the system processes frames in real time without significant delay.
- Ensure detection results are correctly displayed on the output video frames.

7.2.3 Features Tested:

The testing phase covered the major functional features of the Tennis Analysis System using YOLO v5 to confirm that each component was working properly and contributing to the overall performance of the application. Features related to video preprocessing, object detection, bounding box generation, confidence scoring, and output rendering were carefully examined. The purpose of this testing was to verify that the system could accurately analyze tennis videos under realistic gameplay conditions and generate meaningful visual outputs.

The major system features examined during testing included:

- Correct video frame extraction and preprocessing.
- Accurate detection of tennis players and tennis balls using YOLO v5.
- Correct visualization of bounding boxes and confidence scores.
- Reliable processing of continuous video streams.
- Accurate detection under different player movements and ball speeds.
- Correct display of detection results on output video frames.

7.2.4 Integration Testing Strategy:

The integration testing strategy was used to verify whether all major modules of the tennis analysis system worked together effectively after individual testing was completed. Since the system includes multiple interconnected components such as the video input module, OpenCV preprocessing pipeline, YOLO v5 detection model, and output rendering mechanism, it was necessary to test their interaction and data flow. This strategy ensured that the communication between modules remained correct and stable throughout execution. It also helped identify any compatibility issues early, thereby improving the reliability of the complete application.

- Interaction between video input pipeline and YOLO v5 detection model.
- Synchronization of detection results with video frame rendering.
- Compatibility between OpenCV video processing and YOLO v5 inference.
- This approach helped detect integration issues early and ensured stable system performance during real-time analysis.

7.2.5 Acceptance Criteria:

The acceptance criteria were defined to determine whether the proposed tennis analysis system had successfully met its intended functional and performance goals. The system was considered acceptable only when it consistently produced accurate detections, smooth video processing, and clear visual outputs without major errors or instability. These criteria were based on the expected project requirements and practical usability of the application in tennis performance analysis scenarios. Meeting these conditions ensured that the developed system was ready for demonstration, evaluation, and future enhancement.

The tennis analysis system was accepted only when the following conditions were satisfied:

- Accurate detection of tennis players and tennis balls in video frames.
- Stable real-time processing of tennis match videos.
- Correct generation of bounding boxes with confidence scores.
- Smooth display of detection results without frame drops.
- Compliance with defined project requirements and functional goals.

7.2.6 Overall Test Results:

The Overall Test Results indicate that the proposed Tennis Analysis System using YOLO v5 performed successfully across the planned testing scenarios. All major test cases were executed and evaluated, and the system demonstrated stable and reliable behavior during video processing and object detection tasks. The YOLO v5 model showed good capability in detecting both tennis players and the tennis ball from match and training videos. The generated detections were visually represented through accurate bounding boxes and confidence based predictions, which were correctly displayed on the output frames. These results confirm that the detection module functioned effectively under normal operating conditions.

In addition to accurate object detection, the system also maintained satisfactory performance during more dynamic gameplay conditions such as fast player movements, rapid ball motion, and continuous frame processing. The integration of video preprocessing, YOLO v5 inference, and output visualization worked smoothly, producing consistent and stable annotated outputs throughout the testing process. The system was able to process input videos without major frame drops or processing interruptions, which demonstrates its suitability for practical sports video analysis applications. Overall, the testing results show that the developed system is capable of performing reliable tennis action analysis with acceptable accuracy and stability.

7.2.7 Conclusion:

The System testing phase confirmed that the developed Tennis Analysis System using YOLO v5 successfully meets the major functional and performance requirements defined for the project. Through structured testing strategies, module level verification, integration validation, and output analysis, the system proved its ability to process tennis videos accurately and generate meaningful analytical results. The model was able to detect tennis players and the ball effectively, while the surrounding modules ensured proper frame handling, visualization, and output generation. This confirms that the proposed system is technically sound and functionally complete for the intended application.

The final developed system demonstrates a good level of robustness, efficiency, and practical usability for sports analytics and tennis performance evaluation. Its ability to combine computer vision, deep learning, and video analysis into a single workflow makes it useful for applications such as player tracking, movement analysis, training evaluation, and automated tennis match analytics. The successful completion of testing also indicates that the system is ready for further improvement and future deployment. Therefore, the proposed project can be considered a reliable and scalable solution for intelligent tennis video analysis using modern deep learning techniques.

7.3 Sample Test Cases:

Table 7.1: Test Cases

S.No.	Test Case	Expected Result	Result	Remarks (if any)
1.	Video Upload	System successfully uploads and processes the tennis match video	Pass	Verify supported video formats such as MP4 and AVI
2.	Player Detection	YOLO v5 correctly detects tennis players and displays bounding boxes around them	Pass	Test with single-player and multiple-player scenarios
3.	Tennis Ball Detection	System accurately detects the tennis ball in video frames	Pass	Check detection during fast ball movement and occlusion
4.	Real-Time Frame Processing	System processes video frames continuously and displays detection results without delay	Pass	Ensure smooth processing with minimal frame drops
5.	Output Visualization	Detected players and tennis balls are highlighted with bounding boxes and confidence scores	Pass	Verify clarity of bounding boxes and correct object labeling

Test Case 1: Video Upload:

This test case verifies the system's ability to accept and process a tennis match video as input. The user uploads a video file through the system interface, which acts as the starting point of the entire analysis pipeline. The system validates the uploaded file by checking its format, size, and compatibility with supported video standards such as MP4 and AVI. This validation step ensures that only appropriate and processable video files are accepted, preventing errors during further stages of execution.

Once the video is successfully uploaded, the system begins preprocessing by extracting individual frames from the video. These frames serve as the input for the YOLO v5 based object detection model. The extraction process includes operations such as resizing, normalization, and format standardization to ensure consistency across all frames. This step is critical because accurate detection and tracking depend heavily on the quality and uniformity of the input data.

If the video upload process is successful, the system seamlessly transitions to the next stage of analysis. However, if the video file is corrupted, unsupported, or invalid, the system is designed to provide appropriate error messages and user guidance. This ensures robustness and user-friendliness, making the system reliable for real world applications where different types of video inputs may be encountered.

Test Case 2: Player Detection

This test case evaluates the accuracy of the YOLO v5 model in detecting tennis players within the video frames. As the video is processed, the system identifies players and draws bounding boxes around them, along with confidence scores that indicate detection accuracy. Each detected player is also assigned a unique identifier (Player ID), enabling consistent tracking across multiple frames and ensuring that the system can distinguish between different players.

The system must be capable of detecting players under various conditions, including different poses, movements, and positions on the court. It should work effectively in both single player training videos and multi player match scenarios. Additionally, the detection model should handle challenges such as occlusions, varying lighting conditions, and camera angles, which are common in real-world tennis videos.

Accurate player detection is essential for subsequent analysis tasks such as movement tracking, speed calculation, and performance evaluation. The bounding boxes generated during this stage serve as the foundation for tracking player trajectories and understanding gameplay dynamics. Successful execution of this test case confirms that the system can reliably identify players and maintain consistency throughout the analysis process.

Test Case 3: Tennis Ball Detection

This test case focuses on the system's ability to detect and track the tennis ball, which is one of the most challenging aspects due to its small size and high speed. The YOLO v5 model is used to identify the ball in each frame and generate bounding boxes around it. The system continuously updates the ball's position as it moves across frames, ensuring accurate tracking throughout the gameplay.The test also evaluates the system's performance under difficult conditions such as motion blur, rapid direction changes, and partial occlusions. Since the tennis ball often overlaps with players or blends into the background, the model must be robust enough to maintain detection accuracy in such scenarios. The system may also use interpolation techniques to estimate ball positions in frames where detection is temporarily lost.

Successful detection of the tennis ball is critical for analyzing shot trajectories, calculating ball speed, and understanding gameplay patterns. The accuracy of ball detection directly impacts the quality of performance metrics generated by the system. Therefore, this test case ensures that the model can handle real world challenges and provide reliable tracking of the tennis ball during matches.

Test Case 4: Real-Time Frame Processing

This test case verifies the system's capability to process video frames continuously and deliver results in real time. As the video is played, each frame is analyzed by the YOLO v5 model for detecting players and the tennis ball. The system must ensure that this processing happens efficiently without causing delays or interruptions in video playback.

The performance of the system is evaluated based on its ability to maintain smooth frame transitions and minimize latency. It should be capable of handling high resolution videos and fast-paced gameplay without significant frame drops. Efficient memory management and optimized model inference play a key role in achieving real time performance, especially when processing large volumes of data.

Real-time frame processing is essential for practical applications such as live match analysis and training sessions. A system that performs well in this test case can provide immediate feedback to players and coaches, enabling quick decision making and performance improvement. This ensures that the tennis analysis system is not only accurate but also responsive and efficient.

Test Case 5: Output Visualization

This test case ensures that the final output generated by the system is clear, informative, and easy to interpret. After processing the video, the system overlays bounding boxes around detected players and the tennis ball, along with confidence scores. These visual elements help users understand what the system has detected and how accurately it has performed.

In addition to object detection, the output may include trajectory paths, player IDs, and other annotations that provide deeper insights into gameplay. The system should ensure that these visual elements are properly aligned, clearly labeled, and do not obstruct the original video content. A well designed visualization enhances user experience and makes the analysis more accessible to non technical users.

8. RESULTS

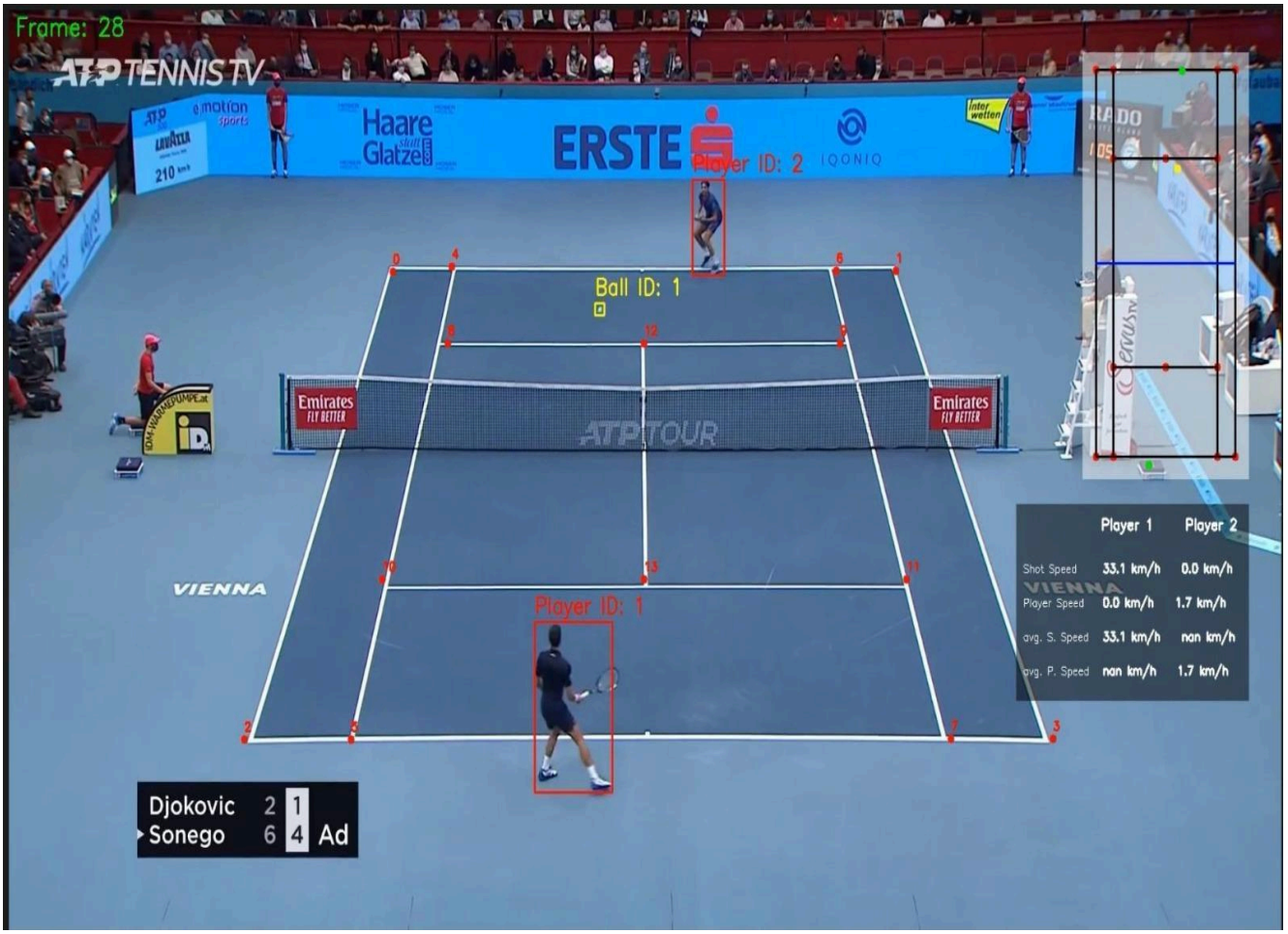

Fig. 8.1: YOLO v5 Based Tennis Detection Output

Description:

The final processed result of the proposed Tennis Analysis System using YOLO v5 is shown in Fig. 8.1, where the uploaded tennis match video has been transformed into an intelligent analytical display. This screen demonstrates how the system performs real-time object detection, tracking, spatial mapping, and performance analysis simultaneously within a single visual interface. The processed frame includes multiple layers of information such as player detection, tennis ball detection, court keypoint mapping, trajectory interpretation, and speed based analytics, all of which are overlaid directly onto the original match footage. As a result, the output is not only visually informative but also highly useful for performance evaluation, tactical analysis, and automated sports understanding. One of the most important features visible in the output is the automatic detection and tracking of tennis players and the tennis ball using the YOLO v5 object detection model.

Each player is enclosed in a clearly defined bounding box and assigned a unique Player ID, which allows the system to consistently identify and monitor the players across multiple frames. Similarly, the tennis ball is also detected and labeled with a Ball ID, despite its small size and fast movement. This is particularly important in tennis analysis because the ball often moves rapidly and can become difficult to track manually. The presence of the frame number at the top of the screen further indicates that the system is processing the video continuously on a frame by frame basis, which is essential for maintaining real-time analysis and smooth tracking performance.

Another major feature shown in the output is the court keypoint detection and court structure mapping, which plays a significant role in understanding gameplay context. Important court locations such as the baseline corners, service line intersections, center lines, and side boundaries are marked using numbered keypoints. These keypoints create a spatial reference system that allows the model to determine where each player and the ball are positioned relative to the actual court layout. This spatial awareness makes the analysis much more meaningful, as it enables the system to understand whether a player is near the baseline, approaching the net, or moving laterally across the court. It also improves the accuracy of ball placement interpretation and tactical movement evaluation during rallies.

The output image also includes a mini-court representation on the top-right side, which provides an additional abstract visualization of player and ball movement. This compact court model helps simplify the tracking data by projecting the real time positions of the players and the ball onto a structured top down view of the court. Such a representation is highly useful for analyzing movement patterns, positioning behavior, court coverage, and rally flow in a more understandable format. It allows coaches and analysts to quickly interpret how players are occupying space on the court and how the ball is traveling during points. This type of visualization transforms raw detection data into meaningful tactical insights, making the system valuable not only for technical analysis but also for strategic decision-making.

In addition to visual detection and court mapping, the system also generates a performance analytics panel on the right side of the output screen. This panel displays important metrics such as shot speed, current player speed, average speed, and other motion related values for both players. These statistics are computed using positional changes across consecutive frames and provide quantitative insight into the intensity and quality of gameplay. Such information can help identify player movement efficiency, speed of reaction, and shot execution characteristics during the match.

The output of the proposed Tennis Analysis System identifies and classifies player actions into four key tennis positions: Serve, Forehand, Backhand, and Volley. These positions represent the most fundamental strokes in tennis and are essential for understanding player performance, movement patterns, and gameplay dynamics.

1. Serve Position

The serve position in the output screen is illustrated in Fig. 8.2, where the system captures the player in a vertically extended posture preparing to strike the ball. Using the YOLO v5 based detection module, the player is identified and enclosed within a bounding box, while the tennis ball is simultaneously tracked during the toss phase. The system processes this action frame-by-frame, ensuring that even rapid upward motion and ball movement are accurately detected. The presence of court keypoints further enhances the contextual understanding of the serve by determining the player's exact location relative to the baseline and service area.

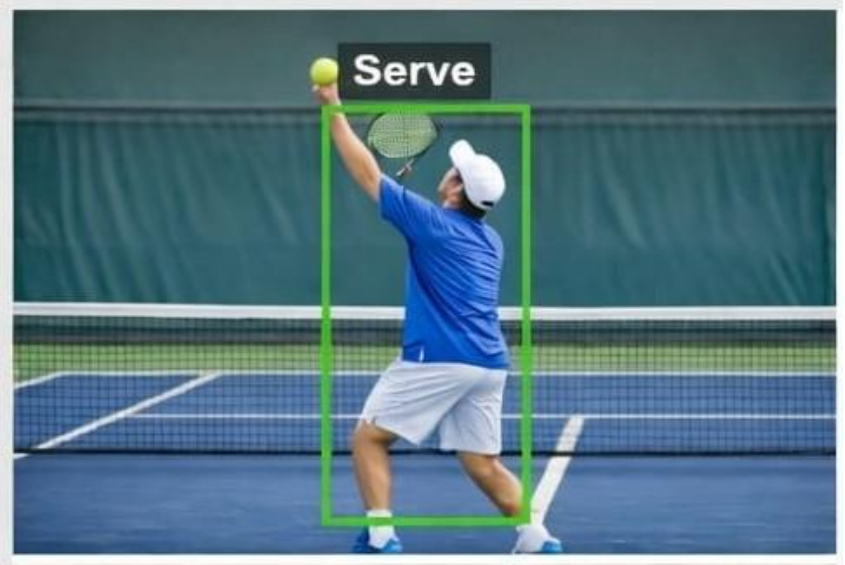

Fig. 8.2: Player Position-1

From a performance analysis perspective, the serve position plays a crucial role in generating key metrics such as ball speed, player motion during the serve, and shot initiation timing. By tracking the ball trajectory from the toss to the point of impact, the system calculates serve speed and direction, which are essential indicators of player performance. Additionally, the mini-court visualization maps the serve action onto a top down court model, enabling better interpretation of ball placement and player positioning. The analytics panel further provides numerical insights such as speed values and shot counts, helping coaches and players evaluate serve effectiveness, consistency, and tactical execution.

2. Forehand Position

The forehand position is depicted in Fig. 8.3, representing one of the most dominant offensive strokes in tennis, where the player executes a forward swing to return the ball. In this position, the YOLO v5 detection model identifies the player's body orientation, racket movement, and ball position to accurately classify the action. The player is enclosed in a bounding box with a unique ID, ensuring consistent tracking across frames, while the tennis ball is simultaneously detected despite its high speed. The court keypoint mapping provides additional spatial context by identifying whether the player is positioned near the baseline, mid court, or moving laterally, which is critical for understanding the dynamics of the forehand stroke.

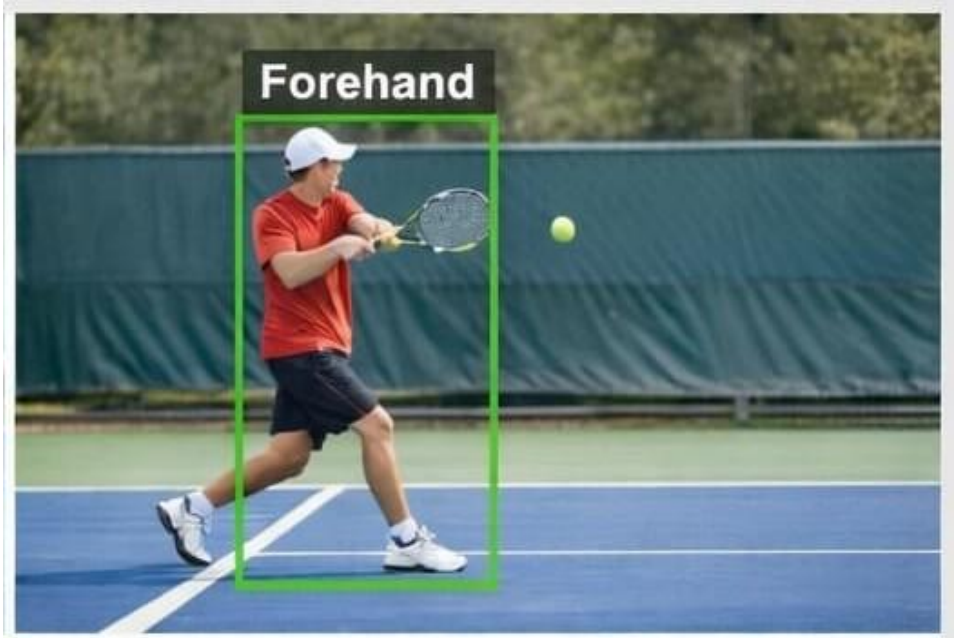

Fig. 8.3: Player Position-2

In terms of analytics, the forehand position contributes significantly to performance evaluation by enabling the calculation of shot speed, player movement speed, and trajectory analysis. The system tracks the ball's path before and after impact, allowing it to measure the effectiveness and direction of the shot. The mini court representation visualizes this movement in a simplified format, helping users understand positioning and shot placement more clearly. Furthermore, the performance analytics panel displays real time metrics such as average speed and shot intensity, providing valuable insights into the player's offensive capabilities. This comprehensive analysis helps in improving stroke technique, timing, and overall gameplay efficiency.

3. Backhand Position

The backhand position is illustrated in Fig. 8.4, reflecting the player's response to shots directed towards the non dominant side, requiring precise coordination and balance. The system detects this action by analyzing the player's body alignment, racket positioning, and ball trajectory using YOLO v5. The player is tracked continuously with a bounding box and unique ID, ensuring accurate identification even during rapid movements or partial occlusions. The tennis ball is also detected and followed across frames, enabling the system to capture the complete backhand motion. Court keypoints play an important role in determining the player's spatial position, helping to understand whether the shot is executed near the baseline or during lateral movement.

Fig. 8.4: Player Position-3

From a performance standpoint, the backhand position is essential for evaluating defensive strength and adaptability. The system calculates metrics such as reaction time, shot speed, and movement efficiency by analyzing frame by frame positional changes. The mini court visualization provides a clear representation of how the player moves across the court while executing the backhand, offering insights into positioning and coverage. Additionally, the analytics panel displays relevant statistics that help in assessing consistency and control during backhand shots. This detailed analysis enables players and coaches to identify weaknesses, improve technique, and enhance overall gameplay performance in challenging situations.

4. Volley Position

The volley position is shown in Fig. 8.5, representing close-range gameplay near the net, where quick reflexes and precise timing are critical. In the output screen, the system detects the player in a forward stance with minimal backswing, indicating a volley action. The YOLO v5 model accurately identifies both the player and the tennis ball, even during rapid exchanges near the net. The bounding boxes and object IDs ensure consistent tracking, while the court keypoint mapping provides spatial context by highlighting the player's proximity to the net. This allows the system to distinguish volley actions from other strokes and analyze them effectively within the gameplay sequence.

Fig. 8.5: Player Position-4

From an analytical perspective, the volley position is used to measure reaction speed, positioning accuracy, and shot redirection efficiency. The system tracks the ball's quick movement and calculates metrics such as response time and ball speed after contact. The mini court representation simplifies this data by showing player positioning and ball movement in a top-down view, making it easier to interpret net play strategies. Additionally, the performance analytics panel provides real-time statistics that help evaluate the player's effectiveness in close range situations. This information is valuable for improving reflexes, decision making, and finishing skills, which are crucial for gaining an advantage during net play.

9. CONCLUSION AND FUTURE SCOPE

9.1 Conclusion

The proposed YOLO v5 based Tennis Action Analysis System represents a significant advancement in the field of sports analytics, particularly in the area of intelligent tennis performance evaluation. Traditional tennis analysis methods often rely on manual observation, pose based systems, or skeletal tracking techniques such as Graph Convolutional Network (GCN) based approaches, which may require specialized sensors, depth cameras, and highly controlled environments. In contrast, the developed system provides a more practical and scalable solution by directly analyzing ordinary tennis video footage using deep learning based object detection. This makes the system easier to deploy, more adaptable to real world conditions, and more useful for a wider range of users including players, coaches, and analysts.

One of the major strengths of the proposed system is its ability to perform accurate player and tennis ball detection in challenging gameplay conditions. By using YOLO v5, the system is capable of identifying players and the ball quickly and reliably, even during high speed rallies, dynamic movements, and changing court positions. This is particularly important in tennis, where the ball often moves at very high speed and player actions change rapidly within fractions of a second. The integration of tracking mechanisms further improves the continuity of analysis by maintaining object movement across frames, enabling the system to generate a consistent and stable understanding of gameplay events throughout the match.

Another important contribution of the system is its ability to generate meaningful performance metrics that go beyond simple visual detection. The system calculates valuable insights such as player running speed, ball shot speed, shot count, movement trajectory, and positional behavior, which are extremely useful for performance evaluation and coaching analysis. These metrics help convert raw video footage into structured and measurable information, making it easier to study a player’s movement efficiency, tactical response, and shot execution. Such analytical outputs can assist coaches in identifying strengths and weaknesses, planning corrective strategies, and monitoring improvement over time in a more objective and data driven manner.

The inclusion of court keypoint detection and spatial mapping further enhances the intelligence of the proposed system by allowing it to understand the position of players and the ball relative to the actual tennis court layout.

This feature adds tactical context to the analysis by enabling the system to determine where players are moving, how they are occupying the court, and where the ball is being placed during rallies. As a result, the system is not only able to detect movement but also interpret gameplay more meaningfully in terms of court coverage, spatial awareness, and tactical positioning. This makes the solution more powerful and relevant for real world tennis training and competitive match analysis. Although the current implementation already demonstrates strong detection accuracy, analytical capability, and practical usability, the project also establishes a strong foundation for future development in AI driven sports intelligence.

Several advanced enhancements can be integrated in future versions, such as real-time live match analysis, 3D pose estimation, automated shot classification, rally segmentation, stroke recognition, and AI based coaching recommendations. These improvements would make the system even more useful as a digital coaching assistant and tactical evaluation tool. In addition, the modular design of the architecture allows new technologies and upgraded models to be incorporated without requiring major changes to the overall framework.

In conclusion, the proposed Tennis Action Analysis System using YOLO v5 can be considered a highly effective and future ready solution for automated tennis performance analysis. It successfully combines computer vision, object detection, tracking, court understanding, and performance analytics into a single intelligent system capable of supporting both technical and tactical evaluation. With future support for mobile devices, cloud platforms, and live streaming environments, the system has the potential to become widely accessible to professional athletes, amateur players, sports academies, and research communities. Overall, this project represents a meaningful step toward fully automated, AI powered tennis analytics and has strong potential to transform how tennis training and performance evaluation are conducted in the modern era.

9.2 Future Scope

The proposed Tennis Analysis System using YOLO v5 provides a strong base for intelligent tennis performance evaluation, but it can be further improved with several advanced features in the future. One important enhancement is the integration of real time live video analysis, which would allow the system to provide instant feedback during practice sessions and matches. This would help players and coaches make immediate corrections and improve performance more effectively. In addition, the system can be optimized for mobile devices, lightweight platforms, and cloud-based environments, making it more accessible and practical for regular use in different training conditions.

Another valuable area for future development is the addition of advanced analytics and motion understanding. Features such as improved ball tracking, 3D pose estimation, shot classification, and tactical movement analysis can provide deeper insights into player technique and gameplay strategy. The system can also be extended with heatmaps, rally segmentation, AI based coaching suggestions, and comparative performance analysis to make the platform more intelligent and useful. With these enhancements, the project can evolve into a complete AI driven tennis coaching and sports analytics system.

- Real-time live video processing can be added to provide instant match and practice feedback.
- The model can be optimized for mobile devices, edge systems, and lightweight deployment.
- Advanced tracking methods can improve the accuracy of player and tennis ball movement analysis.
- 3D pose estimation can be introduced to study player posture and stroke mechanics.
- Automatic shot classification can identify serves, forehands, backhands, volleys, and smashes.
- Heatmaps and tactical analysis can help visualize player movement and court coverage patterns.
- AI-based coaching recommendations can be included for personalized player improvement.
- Cloud and mobile support can make the system more accessible and easier to use in real environments.

REFERENCES

1. Y. Ding, Z. Fan, and Y. Zhao, "YOLO-Ball: Real-time Tennis Ball Detection under Occlusion and Motion Blur," Journal of Sports Engineering and Technology, 2026.

2. M. D. Rafeeq and M. Mallikarjun, “Assessment of Real-Time Object Identification Using Convolutional Neural Networks with YOLO V4,” in *Smart Computing Paradigms: Sustainable Computing*, LNNS, vol. 1318, pp. 453–463, 2025.

3. X. Zhang and B. Li, “Tennis ball detection based on YOLOv5 with TensorRT,” *Scientific Reports*, vol. 15, pp. 21011–21023, 2025.

4. V. M. Desu and S. F. Ali, “Automated tennis player and ball tracking with court keypoints detection (Hawk-Eye simulation),” *arXiv preprint* arXiv:2511.04126, 2025.

5. S. Geetha et al., “High-speed and tiny object tracking in racquet sports using deep learning,” *Procedia Computer Science*, 2025.

6. A. Banoth, N. Hashmi, and R. Bokde, “A review of object detection empowering sports: Key technologies and future outlook,” *AI in Sports*, 2025.

7. K. Fujii, “Computer vision and deep learning for professional sports analytics,” in *CV-Based Sports Analysis Handbook*, Springer, 2025.

8. R. Singh and M. Rao, “YOLOv5-based tennis ball and player tracking for performance analysis,” *IJMRSET*, vol. 13, no. 2, pp. 88–96, 2025.

9. J. W. Chen, “Automated video-based analytics framework for tennis doubles,” *NUS Dissertation*, 2025.

10. S. C. Vanga, R. Vangari, and S. Vasre, “Lane Line and Object Detection Using YOLO v3,” *International Journal of Innovative Science and Research (IJISR)*, vol. 9, no. 6, p. 2371, 2024.

11. J. Wang et al., “Application of tennis motion detection, tracking, and trajectory prediction using deep learning,” *Journal of Sports Engineering*, 2024.

12. T. Zou et al., “Fast sports object tracking using detection plus discrimination for table tennis,” *Scientific Reports*, vol. 14, pp. 1–15, 2024.

13. M. Haq et al., "Player Detection and Posture Recognition for Badminton Using YOLO and Tracking," JOIV, vol. 7, no. 4, pp. 512–520, 2023.

14. Y. Yang, “Ball tracking and trajectory prediction for tennis robots,” *Journal of Computer & Design Engineering*, vol. 10, no. 3, pp. 1176–1191, 2023.

15. D. Gao, Y. Zhang, and H. Qiu, “Automatic detection method of small target in tennis game video based on deep learning,” Journal of Intelligent & Fuzzy Systems, 2023.

16. J. Patel and M. Khan, “Tiny object detection challenges in high-speed sports videos,” *Pattern Recognition Letters*, vol. 165, pp. 22–35, 2023.

17. I. Huang, C. Liao, and C. Chen, “TrackNet: A deep network for tracking high-speed and tiny objects in sports,” *arXiv preprint* arXiv:2007.03698, 2022.

18. B. T. Naik et al., “A comprehensive review of computer vision in sports analytics,” *Applied Sciences*, vol. 12, no. 9, pp. 4429–4452, 2022.

19. P. Korrai and R. Singh, “Trajectory-aware event detection in racket sports,” *Multimedia Tools and Applications*, vol. 81, pp. 31899–31920, 2022.

20. S. Chaudhury and A. Gupta, “Unsupervised temporal aggregation for sports event detection,” *IEEE Transactions on Multimedia*, vol. 23, pp. 3025–3037, 2021.

21. M. Rubio and O. James, “Transformer-based multi-label action categorization in sports,” *Pattern Recognition Letters*, vol. 152, pp. 89–104, 2021.

22. A. Rahman and M. Silva, “Enhanced ball trajectory estimation using optical flow and Kalman filters,” *Applied Soft Computing*, vol. 95, pp. 106–120, 2020.