Automated Basketball Object Detection and Tracking

1. Hetu Patel
Computer Science Student
Toronto Metropolitan University
Toronto, Canada
hetu.patel@torontomu.ca

2. Eric Mergelas
Computer Science Student
Toronto Metropolitan University
Toronto, Canada
eric.mergelas@torontomu.ca

3. Kushal Bhattad
Computer Science Student
Toronto Metropolitan University
Toronto, Canada
kushal.bhattad@torontomu.ca

4. Chris Grover
Computer Science Student
Toronto Metropolitan University
Toronto, Canada
c1grover@torontomu.ca

Abstract **- Automated analysis of basketball games requires robust object detection and tracking systems capable of handling the dynamic and complex nature of live sports footage. This paper addresses the challenges of tracking fast-moving players, occlusions, camera motions, and complex backgrounds in basketball videos. We present a system that utilizes supervision libraries and custom techniques to detect and annotate players, the ball, shooting actions, and the rim. Positional constraints and team classification methods are applied to enhance tracking accuracy. The proposed approach demonstrates improved performance in object detection and tracking, enabling quick detection and analysis of basketball games.**

Keywords - Basketball Analytics, Object Detection, Tracking, Team Classification.

I. INTRODUCTION

Automated systems for analyzing basketball games are essential for extracting insights into player performance and even reducing injuries [1]. Traditional methods struggle with the rapid movements of players, frequent occlusions, and varying camera motions, leading to inaccuracies in object detection and tracking [2].

II. BACKGROUND

Tracking objects in basketball videos is uniquely challenging due to the sport's complex nature [3]. Players exhibit rapid and unpredictable movements, often accelerating, changing direction, or jumping, which can cause motion blur and confuse detection models [2]. Frequent interactions between players, such as screens and blocks, increase physical proximity, making it harder to distinguish individuals, especially during occlusions when players or the ball are momentarily obscured [2].

Dynamic camera movements, including pans, zooms, and rotations, introduce additional difficulties, often causing motion blur or disrupting object detection [2]. Additionally, annotation requires contextual understanding to accurately label actions like shooting, passing, or dribbling, which often depend on a sequence of movements rather than a single frame [4].

Previous approaches have utilized deep learning models for object detection and action recognition in sports analytics [3]. However, these approaches have often lacked real-time processing

capabilities and struggle with the dynamic environment of live basketball games [3].

III. MATERIALS AND METHODS

A. System Overview

The system comprises modules for video preprocessing, object detection, ball tracking, team classification, and visualization. It leverages the Roboflow 3.0 object detection model and the Supervision framework for processing.

B. Video Preprocessing

The Supervision framework was utilized for frame extraction, offering a generator-based approach that ensures seamless handling of large video files [5]. This method iteratively loads frames into memory using "sv.get_video_frames_generator", reducing excessive memory usage and enabling compatibility with GPU-accelerated operations [5]. This process allows for granular analysis of each frame. Frame sampling was used at regular intervals to increase computational efficiency. By selecting a stride rate (e.g., every 30th frame), redundant computations can be minimized while reducing key motion details in the video. The stride rate can be adjusted as the requirements of analysis change.

To accelerate preprocessing tasks, ONNX Runtime with CUDA Execution was used. Python libraries tqdm and numpy are used for progress tracking and data manipulation. To streamline processing and analysis, only the first 10 seconds of the video were targeted for preprocessing. Metadata such as frame rate and total frame count are extracted using Supervision. The sv.VideoInfo function was used to retrieve frame rate and resolution. The total frame count is calculated using frame rate and duration. By shortening the segment of the video, we reduce computational requirements and allow for quicker iterations during development.

C. Model Training

The YOLOv8 model was trained to optimize basketball analytics using a dataset from Roboflow comprising 4,061 annotated images. The dataset included bounding boxes for players, the ball, and the rim, enabling precise object detection. Two training setups were conducted to refine the model's accuracy and performance.

In the first setup, YOLOv8 was trained for 10 epochs, focusing on balancing GPU memory usage with smaller batch sizes. The model achieved strong initial results, with an mAP50 of 0.934 and high precision (0.857) and recall (0.901). However, challenges arose with detecting overlapping objects and underrepresented classes like "shoot," leaving room for further improvements.

The second setup fine-tuned the model for 15 epochs, building on the weights from the first run and adjusting hyperparameters like learning rate and batch size. This resulted in an mAP50 of 0.942 and an improved mAP50-95 of 0.737, indicating enhanced accuracy at stricter IoU thresholds. Precision increased to 0.894, with recall remaining stable at 0.91, reflecting better detection of objects with fewer false positives.

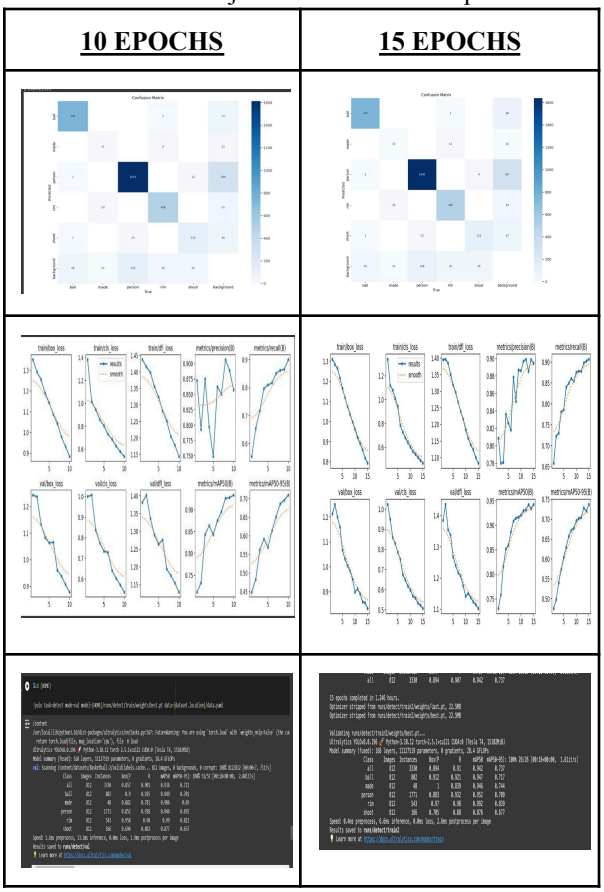

Fig. 1. Comparison of performance with models trained at epoch 10 and 15.

Performance analysis showed significant gains in detecting overlapping and occluded objects in the 15-epoch model compared to the 10-epoch setup. Confusion matrices revealed reduced false positives and higher true positive rates, particularly for detecting "ball," "rim," and "person." Precision and recall graphs

further validated the model's improvements in key metrics.

Training was conducted on Google Colab with GPU acceleration, achieving ~4 minutes per epoch for the 10-epoch setup and ~1.2 hours for the 15-epoch run. The optimized model demonstrated real-time inference capabilities with a runtime of ~9.6 ms per image, making it suitable for efficient basketball action detection.

D. Object Detection

Both our custom YOLOv8 model and an open source Roboflow 3.0 Object Detection (Fast) model were evaluated for basketball object detection. YOLOv8 offered high flexibility, while Roboflow 3.0 was pre-trained on a similar dataset and was optimized for ease of deployment. Roboflow 3.0 was ultimately chosen for its ease of use for development [9].

Player detection and tracking are managed using ByteTrack, which assigns a unique ID to each detected player, ensuring consistent identification across frames [5]. The process involves filtering detections by class ID to isolate players and applying Non-Maximum Suppression (NMS) to refine overlapping bounding boxes and eliminate duplicates, enabling smooth tracking throughout the video [2].

Rim tracking leverages its unique class ID for precise localization in each frame, which is critical for analyzing proximity to the ball and correlating with shooting actions. Similarly, shooting detection is integrated through a custom pipeline that identifies players in the act of shooting using a dedicated class ID, with actions highlighted through distinct annotations.

The ball is also tracked using ByteTrack. A filter_ball_by_distance function was introduced in testing which smooths the trajectory by validating ball positions within a distance threshold to eliminate erratic detections.

E. Team Classification

Player crops are generated by sampling one frame per second from the video, detecting the players in each frame, and isolating their images. The Siglip Vision Model, a pre-trained transformer model, is used to compute high-dimensional embedding for these crops [6]. To optimize computational efficiency, crops are processed in batches using the “create_batches” function. This minimizes memory usage and accelerates inference on the GPU. The batched images are fed into the Siglip model, where the outputs are aggregated to generate feature vectors.

The embeddings generated by the Siglip model are high-dimensional, which makes them computationally expensive to process for clustering. UMAP (Uniform Manifold Approximation and Projection) is used to reduce dimensionality of these embeddings into 3D spaces [7]. This allows for suitable visualization and clustering.

KMeans clustering is applied to the embeddings to group the players into three clusters: Team 1, Team 2, and Referees. The clustering process assigns labels to each crop based on the proximity in the feature space, splitting players into two different teams and the outliers being classified as referees. The TeamClassifier model is an adaptation of the original TeamClassifier from the Roboflow sports repository [8]. This version was modified to handle three clusters instead of two. Early attempts to create 4 clusters (Team 1, Team 2, Referees, and Bystanders) showed referees and bystanders being stored in the Referee cluster and some of the members of Team 2 being found in the Bystanders cluster. This approach was ultimately deemed unnecessary, since distinguishing referees from bystanders did not impact gameplay statistics. The final model has Referees and Bystanders stored in a single cluster, focusing resources on the more critical task of differentiating between Team 1 and Team 2.

F. Visualization

The Supervision library was used to annotate detected objects such as players, the ball, and the rim, creating clear visualizations of game dynamics. Each player is marked with a colored circle beneath their position, representing their team: blue circles for Team 1 and yellow circles for Team 2, while referees are distinguished with magenta annotations. To highlight the ball, a triangle annotation hovers above it, pointing down to provide tracking that remains visible even in crowded or fast-paced scenarios. The rim is annotated with a green ellipse, ensuring it is clearly highlighted even in cluttered scenes or during occlusions. Shooting actions are indicated with a red bounding box around the player, signaling the event clearly in real time.

Additionally, unique player IDs are displayed as labels below each player, simplifying individual tracking and identification throughout the game. See sample frame in Fig 2.

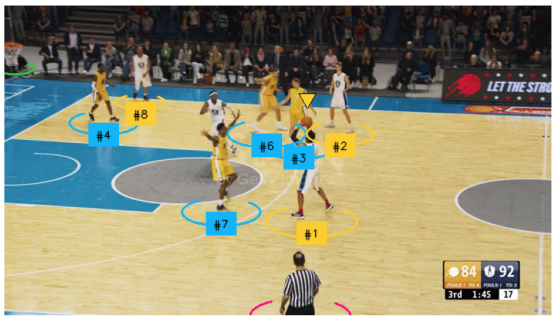

Fig. 2. Annotated Video Instance

IV. RESULTS

A. Model Performance

The YOLOv8 model delivered strong results. The 10-epoch model showed precision of 85.7%, recall of 90.1%, mAP50 of 93.4%, and mAP50-95 of 71.1%, excelling in detecting larger classes like ball, person, and rim. The 15-epoch model improved further with precision at 89.4%, mAP50 at 94.2%, and mAP50-95 at 73.7%, enhancing bounding box quality and stricter IoU performance. While smaller classes like made and shoot underperformed due to limited data, the model remains highly effective for real-time basketball analytics, with room for improvement via data augmentation and tuning.

B. Tracking Performance

When a player becomes obstructed, such as moving behind another player or temporarily leaving the field of view, they are not detected during these moments. However, upon re-entering the frame or becoming unobstructed, the system successfully reassigns them a detection, albeit with a new tracker ID. This discontinuity results in a player potentially being assigned multiple IDs during the game. The rim detection, by contrast, performed consistently well, with a green circular annotation effectively marking its position across all frames, even during occlusions or rapid camera movement. This provides a reliable reference point for analyzing shooting actions and player positioning relative to the basket.

Challenges arose when calculating the duration of ball possession by each team. The model only accounted for moments of direct contact between the ball and a player, rather than the full duration the ball was guided or controlled. Additionally, minor discontinuities were observed during highly dynamic sequences or significant occlusions, temporarily impacting tracking accuracy.

The shooting detection system demonstrated strong performance, accurately identifying shooting actions by highlighting players with a red bounding box. However, occasional false positives occurred, such as misclassifying non-shooting motions, like a player raising the ball overhead, or detecting referee gestures as shots.

C. Team Classification

The classification model assigned 175 samples to Team 1, 174 samples to team 2, and 114 samples to Referees. As shown in Fig 3, the balance in the number of samples assigned to Team 1 and Team 2 confirms the system's fairness in distributing clusters. The lower number of referees aligns with gameplay context as there are only 2 referees but 10 players on the court at all times. As shown in Fig 4, the players with the yellow jerseys were sorted into Team 1 and the players with white uniforms were sorted into Team 2. The referee cluster contains referees as well as detected bystanders. This overlap was deemed acceptable since differentiating referees from bystanders does not impact gameplay analytics.

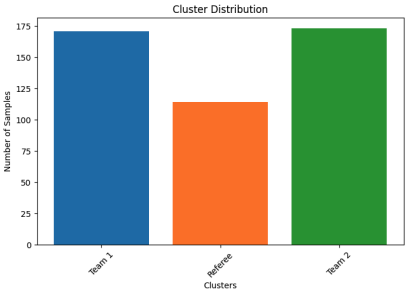

Fig. 3. Samples assigned to the Team

The system's use of UMAP for dimensionality reductions and KMeans for clustering proved effective. The 3 cluster approach provided sufficient granularity to separate gameplay entities while avoiding unnecessary complexity. This classification system lays the foundation for scaling the system for other sports and larger datasets.

Fig. 4. Team Distribution

V. DISCUSSION

A. Model selection

The model training process utilized YOLOv8 for real-time object detection, achieving high precision (mAP50: 94.2%) and efficient performance. Data was sourced from Roboflow, chosen for its robust annotation and preprocessing tools. Training included a 10-epoch initial run and a 15-epoch fine-tuned run, improving mAP50-95 to 0.737, reducing false positives, and enhancing accuracy for overlapping objects, though smaller classes like "shoot" remained challenging. See Fig 5.

Aspect	Epoch 10	Epoch 15
Key Features	Bounding boxes for players, ball, and rim.	Focused on improving underrepresented class detection.
mAP50	0.934	0.942
mAP50-95	0.711	0.737
Precision	0.857	0.894
Recall	0.901	0.91
Training Duration	~42 mins in total	~1.2 hours in total
Performance Strength	High precision and recall for larger classes (ball, person, rim)	Improved precision and recall across all classes, especially for “ball” and “rim”
Performance Weakness	Lower performance for smaller classes (shoot, made)	Still struggles with “shoot” class but precision improved

Fig. 5 Comparison of performance with models trained at epoch 10 and 15.

B. Tracking improvements

The tracking system showcased strengths and limitations across key areas. Player tracking faced challenges with temporary occlusions and players leaving the frame, leading to tracking discontinuities and multiple IDs, which could be addressed by integrating re-identification models. Rim detection was accurate, maintaining consistent tracking even during occlusions and rapid camera movements, providing a stable reference for analyzing player positioning and shooting accuracy. Ball tracking struggled with capturing the entire possession duration, focusing only on direct contact moments, while discontinuities during dynamic plays highlighted the need for advanced filtering and predictive tracking methods. Shooting detection effectively identified actions with bounding box visualizations but occasionally misclassified non-shooting gestures, showing the importance of refining classification criteria and adding contextual cues.

C. Team classification improvements

The team classification model demonstrated effective performance, with UMAP and KMeans clustering accurately separating teams and referees. Balanced assignments aligned well with gameplay dynamics, though the referee cluster occasionally included bystanders, which was acceptable for analytics purposes. The model's success highlights its scalability to other sports and larger datasets, offering a good foundation for future expansion.

VI. CONCLUSION

This paper presented a system for automated object detection and tracking in basketball videos. By addressing common challenges such as fast player movements and occlusions, the system demonstrated improved accuracy in detecting and annotating key

game elements. The application of positional constraints and team classification methods proved effective in enhancing tracking performance. applications and expanding its adaptability to other sports.

VII. FUTURE WORK

Future advancements in basketball analytics can focus on developing a model to detect keypoints on the court, such as the edges, center circle, free-throw line, and three-point arc. By accurately mapping these keypoints, the court can be geometrically reconstructed into a 2D plane, enabling enhanced tracking of players and the ball. Combined with detections of players and the ball, this mapped court can be used to render a 3D model of players positions on the court in real-time.

Using detected court keypoints, distances between specific locations can be calculated to derive the relative positions of players and the ball. This enables advanced metrics such as velocity and acceleration by continuously tracking player movements on the court. These metrics can reveal performance insights, such as sprint speeds and reaction times. Moreover, incorporating the ball's 3D position and trajectory allows for predictive analysis to determine whether a shot is likely to succeed, based on factors such as the ball's arc, velocity, and position relative to the rim. By associating these mapped distances with player and ball detections, the 3D model can offer real-time gameplay analysis that extends beyond what traditional systems provide.

Ball tracking can also be improved by leveraging court keypoints. For instance, the system can ensure the ball's position in subsequent frames lies within a plausible range based on its estimated velocity and direction, reducing false positives and maintaining tracking consistency, even during occlusions or fast-paced movements. Implementing this system will require high-quality datasets annotated with court key points and player positions, accurate algorithms to manage court distortions from dynamic camera angles, and integration of physics-based models to improve trajectory predictions.

VIII. ACKNOWLEDGMENTS

We would like to thank username "OwnProjects" for collecting the annotated images and training the pre-trained Roboflow 3.0 Object Detection (Fast) model used in this project [9]. Our project was also inspired by insights from a Roboflow Google Doc, which served as a resource for implementing object detection [10].

IX. REFERENCE

1. Xu, T., & Tang, L. (2021). Adoption of machine learning algorithm-based intelligent basketball training robot in athlete injury prevention. *Frontiers in Neurorobotics*, *14*, 620378. https://www.frontiersin.org/journals/neurorobotics/articles/10.3389/fnbot.2020.620378/full
2. Rahimian, P., & Toka, L. (2022). Optical tracking in team sports: A survey on player and ball tracking methods. Journal of Quantitative Analysis in Sports. https://www.degruyter.com/document/doi/10.1515/jqas-2020-0088/html
3. Lyu, T., Liu, J., Liu, X., & Qu, M. (2025). EITNet: An IoT-enhanced framework for real-time basketball action recognition. Alexandria Engineering Journal. https://www.sciencedirect.com/science/article/pii/S1110016824010706#b1
4. Arbues Sangüesa, A. (2021). A journey of computer vision in sports: From tracking to orientation-based metrics. https://arbues6.github.io/assets/pdf/compressed_%20Thesis_AdriaArbues.pdf
5. Roboflow. Supervision [Computer software]. https://github.com/roboflow/supervision
6. Hugging Face. SifLip [Computer software] https://huggingface.co/docs/transformers/en/model_doc/siglip
7. Imcinnes. Umap [Computer software] https://github.com/lmcinnes/umap
8. Roboflow. Sports [Computer software] https://github.com/roboflow/sports
9. Roboflow Universe. "Basketball Object Detection Dataset and Model." https://universe.roboflow.com/ownprojects/basketball-w2xcw
10. Roboflow. (2024, August 30). *football-ai.ipynb*. Google colab. https://colab.research.google.com/github/roboflow-ai/notebooks/blob/main/notebooks/football-ai.ipynb