Vision-based spatiotemporal analysis of football matches

1st Francesco De Lucchini
dept. of Information Engineering
Università di Pisa
La Spezia, Italy
f.delucchini@studenti.unipi.it

2nd Salvatore Laporta
dept. of Information Engineering
Università di Pisa
Pisa, Italy
s.laporta5@studenti.unipi.it

***Abstract*—The increasing availability of high-resolution sports footage and the advancements in the field of computer vision have opened new opportunities in the domain of sports intelligence. Among these opportunities is the automated analysis of football matches, both during during live gameplay, which offers substantial benefits such as improved broadcasting experience and enhanced fan interaction, and offline, which offers support for coaching staff in tactical evaluations.**

In this paper, we present a machine learning pipeline built upon the YOLOv11 object detection model, capable of identifying players, goalkeepers, referees, and the ball with a mean Average Precision at IoU threshold 0.5 (mAP@50) of 0.827.

Additionally, we introduce an unsupervised learning approach utilizing K-means clustering to effectively group players into two distinct teams without prior labeling.

The proposed architecture operates at approximately 30 frames per second on a consumer-grade NVIDIA RTX 2060 Super GPU with 8GB of VRAM, when processing 1920×1080 resolution video streams. This performance demonstrates the feasibility of real-time, vision-based football analysis on commodity hardware. Such a system is particularly suited for applications in cost-sensitive environments, such as local sports facilities aiming to offer enhanced video services and basic AI-driven analytics without the need for enterprise-level computational resources.

Overall, this work provides a practical foundation for real-time sports analytics and paves the way for the development of more advanced vision-based systems in accessible, real-world settings.

***Index Terms*—YOLO, Object detection, Sports analytics, Computer vision, Football**

I. Introduction

The field of sports analytics has witnessed rapid evolution with the integration of computer vision and deep learning techniques [1] [2].

These technologies are particularly useful in football, a sport characterized by constant motion, complex interactions, and a large number of visual elements on the field.

One of the key challenges in football video analysis is the real-time detection and classification of entities such as players, referees, goalkeepers, and the ball. In addition to object identification, it is often crucial to determine the team affiliation of each player, which is not always explicitly available in raw footage and can be complicated by similar jersey designs or lighting conditions.

Identifying these elements accurately is a prerequisite for many downstream applications, including automatic highlight generation, extraction of the ball's trajectory, on-demand 3D reconstructions and generation of virtual views, tactical evaluation, and statistics such as possession time percentage and total number of passes per team [3].

II. Related Work

Football match analysis initially relied on manual observations and physical measurements to estimate player movements and strategies [4]. The early 2000s saw the emergence of computer vision techniques using multi-camera setups and background subtraction to track players [5]. With the deep learning revolution, especially the rise of convolutional neural networks (CNNs), robust detectors for players and the ball were developed, significantly enhancing the accuracy of tracking and event recognition [6], [7]. These advances laid the groundwork for referee assistance systems, enabling near-real-time foul detection and automated offside evaluation through frame-wise spatial analysis [8]. In recent years, benchmark datasets like SoccerNet have unified evaluation protocols and facilitated progress across tasks such as camera calibration, player re-identification, action spotting, and tactical understanding [9]. Together, these contributions represent the shift from handcrafted, offline annotations to fully automated, scalable football video analytics.

III. Backgrounds

Object detection models are generally classified into two main categories: two-stage and one-stage detectors.

Two-stage detectors, such as Faster R-CNN [10], first generate region proposals and then classify and refine bounding boxes. They are known for high accuracy but often suffer from slower inference speeds, making them less suitable for real-time applications, such as sports analytics.

In contrast, one-stage detectors perform detection in a single forward pass, directly predicting class probabilities and bounding boxes over a dense grid. These models, such as SSD [11] and the YOLO family (You Only Look Once) [12], offer superior speed and are commonly used in real-time systems.

As an example architecture for the YOLO family, we briefly illustrate the architecture of the YOLOv5 model [13]. YOLOv5 is an object detection model that uses anchor boxes, which are predefined bounding boxes of different sizes and

aspect ratios, to identify objects in an image at various scales and orientations. The image is first passed through an input layer and then processed by the backbone of the network, which extracts features at multiple levels. These features are then passed to the neck of the architecture, where they are combined and refined to produce three feature maps. Each map is responsible for detecting objects of a specific size: 80×80 for small objects, 40×40 for medium objects, and 20×20 for large objects. The feature maps are then sent to the head of the model, which performs the classification and localization tasks. The detection results are stored in a multidimensional array that contains the predicted class, confidence score, bounding box coordinates, and the width and height of each detected object.

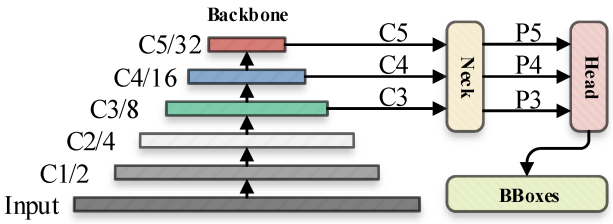

Fig. 1. YOLOv5 architecture [14]

IV. Overview

In this work, we present a prototype system for real-time detection of players, goalkeepers, referees, and the ball within football videos. The proposed solution consists of two main components: a fine-tuned YOLOv11-based object detection module and an unsupervised team classification stage based on K-means clustering.

First, we describe the dataset employed for fine-tuning, followed by a discussion of its limitations and the corrective measures implemented. Subsequently, we detail the data splitting methodology, augmentation techniques, training strategy, and computational resources utilized. Finally, we present and analyze the results obtained.

A. Dataset

The original dataset[1] selected for fine-tuning YOLOv11 comprised 372 images, each with a resolution of 640×640 pixels, annotated with four object classes: `ball`, `goalkeepers`, `players`, and `referees`. However, the dataset exhibited several quality issues, including inconsistent annotations such as the presence of multiple balls, the absence of referees, and an unrealistic number of moving players (i.e., over 20) in a single frame.

To address these deficiencies, a comprehensive manual review of all 372 images was conducted. Each image was meticulously inspected, and annotations were corrected or refined as necessary to ensure accuracy and consistency across all object classes.

[1]https://universe.roboflow.com/roboflow-jvuqo/football-players-detection-3zvbc/dataset/14

In order to enhance the dataset's diversity and robustness, 28 additional images were incorporated, resulting in a final set of 400 labeled images. The augmented dataset captures a broader range of game scenarios, camera perspectives, and lighting conditions, thereby aiming to improve the model's generalization capabilities.

Moreover, YOLO's training pipeline applies several data augmentation techniques by default to increase even more robustness across varying visual conditions. These include random variations in the levels of saturation, brightness, scale (zoom), and random effects such as translations, horizontal flips, and mosaics (stitch four training images together into one, enriching the context and improving small-object detection).

B. Data splitting

A conventional approach to model development and evaluation involves partitioning the dataset into three subsets:

- Training set: used to fit the model.
- Validation set: used for hyperparameter tuning and model selection.
- Test set: used to assess the generalization performance of the final selected model on previously unseen data.

However, given the relatively small size of our dataset (400 labeled instances), a fixed partitioning can result in performance estimates that are sensitive to the specific composition of the splits. In particular, each subset may, by chance, contain samples that are disproportionately easy or difficult to predict, leading to biased or unstable evaluation metrics.

To mitigate this issue, we employ, as shown in Fig. 2, **K-fold cross-validation**:

1) The dataset is first divided into a training set (80%) and a test set (20%).
2) We then perform 5-fold cross-validation on the training portion to evaluate and select the best-performing model based on the average performance across the validation folds.
3) The selected model is retrained on the full training set (i.e., the entire 80%) and evaluated on the held-out test set, which remains untouched throughout training and model selection.

Ideally, in scenarios involving limited data, a nested cross-validation strategy would provide more reliable performance estimates by repeating the entire training and evaluation process across multiple test splits. Alternatively, repeating the above procedure multiple times with different random seeds offers a computationally feasible approximation. Nonetheless, due to resource limitations, we opted for a single test split in this study.

V. Training strategy

All model training and evaluations were conducted using an RTX 2060 Super GPU with 8 GB of VRAM. This is a low-tier consumer-grade GPU, which aligns with the practical and resource-conscious objectives of this study.

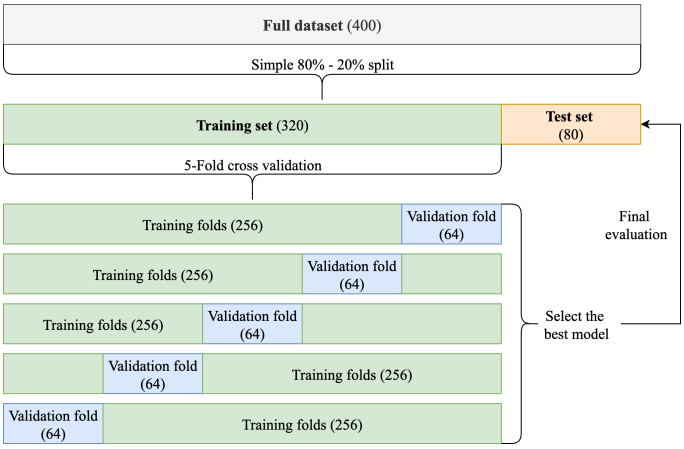

Fig. 2. Diagram illustrating the splitting of the dataset

We fine-tuned all available YOLOv11 model variants (`n`, `s`, `m`, `l`, `x`) on the curated dataset described above. Each model was trained for 100 epochs using the default data augmentation techniques provided by the Ultralytics YOLO framework, with one exception: the `scale` parameter was reduced from 0.5 to 0.2, thereby limiting the random image scaling range to a maximum of $\pm 20\%$.

When it comes to the evaluation metrics, we opted for the built-in validation and benchmarking tools provided by the Ultralytics YOLO framework:

1) Average Precision at IoU = 0.5 (AP@50): the average precision scores across all classes, computed at an Intersection over Union (IoU) threshold of 0.5.
2) Confusion Matrix: useful to identify common misclassifications and to understand where the model is going wrong.
3) Confidence, Precision and Recall curves: useful to evaluate whether the model is struggling more with recalling objects or with assigning the correct labels

During experimentation, all model variants were compared based on their average mAP@50 values computed over the validation folds. The best-performing model was then retrained on the entire training set and subjected to a final evaluation using the previously held-out test set. This final evaluation employed all three validation metrics listed above. The model selected through this process is the one adopted for deployment in the production environment.

VI. Results

A. Selection of the Best Model

Following just over 5 hours of training, the evaluation results were obtained. Fig. 3 presents the mean Average Precision at IoU = 0.5 (AP@50) along with the standard deviation for each object class across the candidate models.

Based on this quantitative comparison, the `m` model was selected as the most suitable choice, offering a favorable balance between detection accuracy and computational efficiency. While a more rigorous statistical approach, such as increasing the sample size and computing confidence intervals, or conducting other statistical tests, would provide more robust insights, the observed trends in the chart are sufficient to indicate that the *m* variant achieves the best trade-off for the intended application.

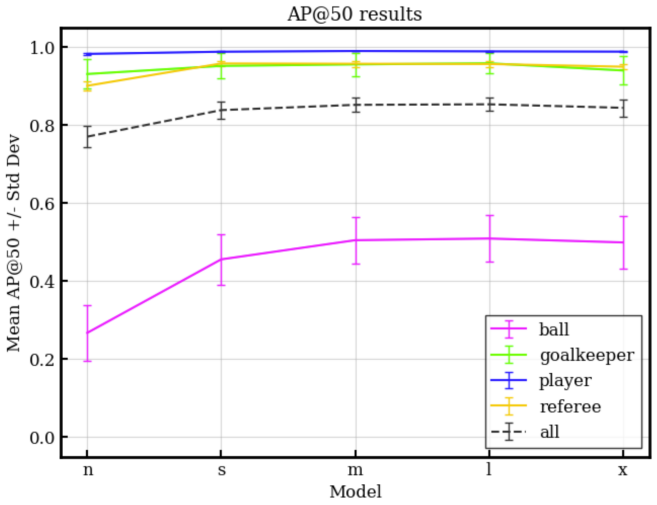

Fig. 3. Mean $\pm$ standard deviation of AP@50 per class, for each candidate model

B. Evaluation of the Final Model

The selected model, `m`, was retrained on the full training set and evaluated on the previously held-out test set, employing the three evaluation metrics outlined in Section V.

Fig. 5 displays the AP@50 scores per class on the test set. The model exhibits strong detection performance for `players`, `goalkeepers`, and `referees`. However, detection of the `ball` remains significantly weaker, with an average AP@50 of approximately 0.4.

The under-performance on ball detection can be attributed to several factors:

- The small physical size of the ball relative to other objects in the frame.
- The visual similarity between the ball and other round or white elements (e.g., penalty spots, players' heads).
- The low frequency of ball occurrences (typically only one per image).

The normalized confusion matrix in Fig. 6 provides additional insight into class-specific prediction behavior. The following observations can be made:

- As previously noted, `balls` are frequently not detected at all, contributing to their low recall and AP scores.
- Distinguishing between `players`, `goalkeepers`, and `referees` proves to be somewhat challenging due to their visually similar appearances, often differentiated only by subtle cues such as uniform color.
- The model tends to *hallucinate* `players`, likely due to ambiguous image compositions (e.g., a player lying on the ground or multiple players closely contesting the ball), leading to false positives.

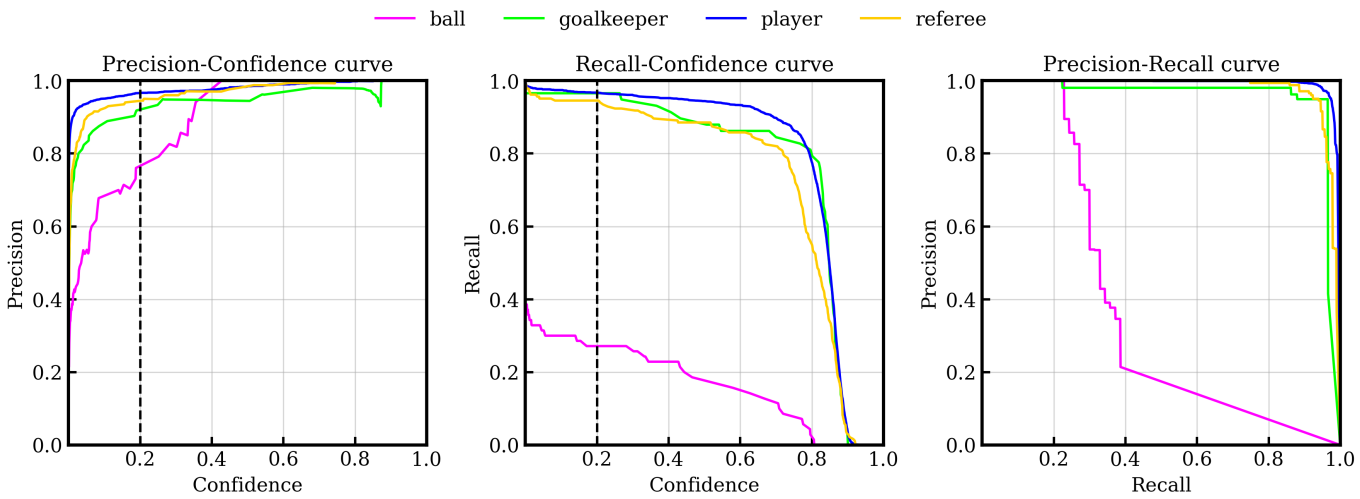

Fig. 4. Precision–Recall curves for the selected model `m`. A confidence threshold near 0.2 offers a good trade-off between precision and recall

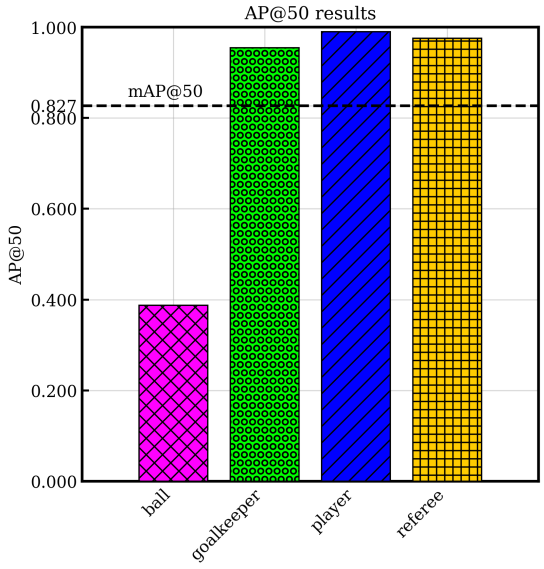

Fig. 5. AP@50 per class for the best model `m`, evaluated on the held-out test set. Ball detection shows notably lower performance

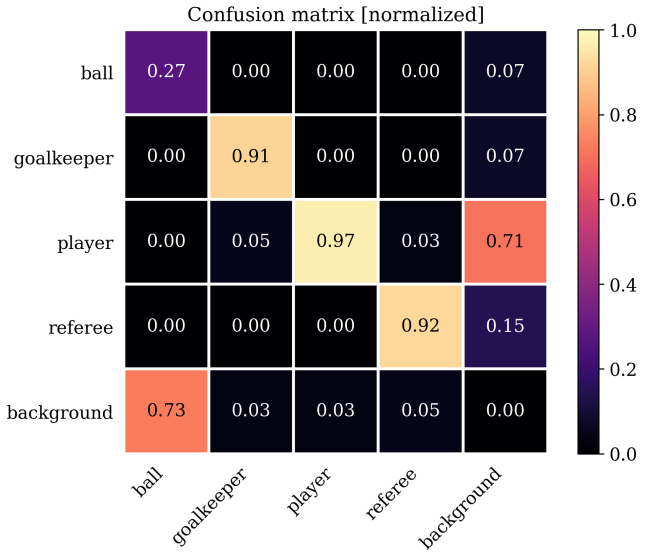

Fig. 6. Normalized confusion matrix for the final model `m`, indicating class-wise accuracy and common misclassifications

VII. Team Assignment

As team affiliation labels are not available in the dataset, an unsupervised approach is adopted to assign detected players to their respective teams. Due to the lack of ground truth for team membership, no standard performance metrics (e.g., accuracy or precision) can be reported. Nonetheless, the process yields visually coherent and meaningful groupings, as illustrated in the following pipeline.

Following bounding box detection, the system extracts jersey color features from each detected player. Specifically, a small rectangular patch is cropped from the bounding box at approximately two-thirds of the player's height, typically capturing the torso area, which best represents the team uniform. A set of sample patches from a single frame is shown in Fig. 7.

From each patch, the dominant color is computed by averaging the pixels values. To improve separability, only the *Hue* and *Saturation* components are retained, as these channels better capture perceptual color differences than raw RGB values. The resulting color vectors are then passed to a K-means clustering algorithm with the number of clusters set to $K = 2$.

Goalkeepers and referees are excluded from this process, as their uniforms are typically distinct and would introduce noise into the clustering of moving players. After clustering, each player is assigned to one of two teams based on proximity to the cluster centroids. An example of the resulting team

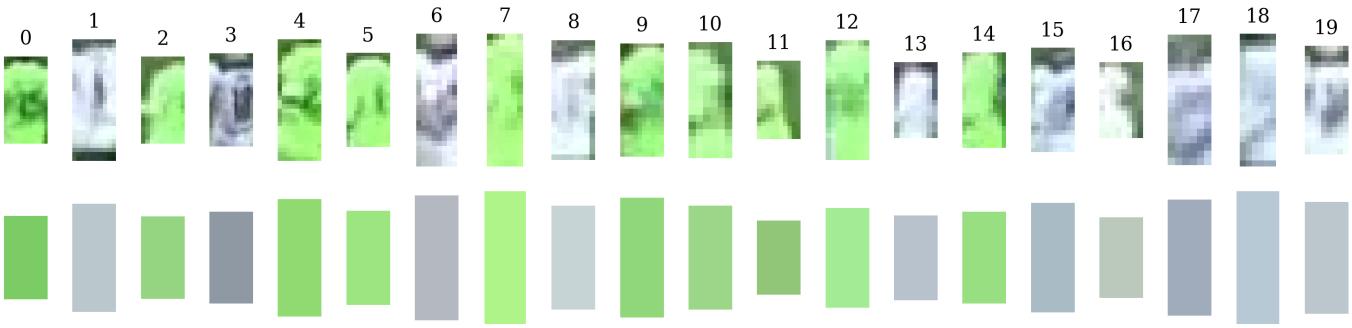

Fig. 7. Example torso patches extracted from detected player bounding boxes. These samples, taken from a single frame, represent the visual features used for unsupervised team clustering

assignments is visualized in Fig. 8, where players in the same cluster are marked with a shared color label.

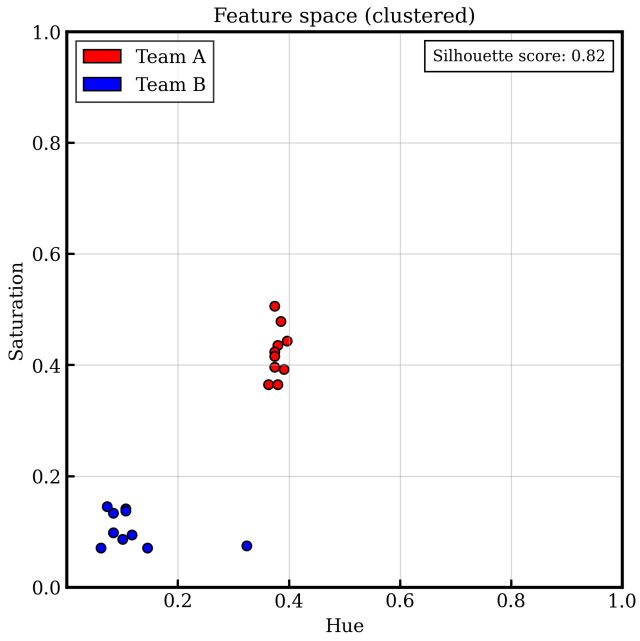

Fig. 8. Result of K-means clustering on torso color features

This unsupervised method enables generalization across matches and teams, without requiring prior knowledge of jersey designs or team identities. On a sample demonstration video, the clustering produced an average silhouette score of 0.8, indicating well-separated and coherent clusters.

VIII. Conclusions

Overall, the quantitative evaluation confirms that our prototype performs reliably on the major object classes (`players`, `goalkeepers`, and `referees`), and offers a practical precision-recall trade-off suitable for real-time inference applications.

However, the challenges observed in `ball` detection highlight a significant area for future enhancement. In particular, the use of relatively low-resolution 640×640 images exacerbates the problem; compression artifacts and motion blur strongly influences the visibility of small, fast-moving objects like the ball, which often spans only a few pixels.

To address these issues, future work should consider expanding the dataset with a larger number of high-quality, high-resolution images and accurate annotations under diverse conditions. The use of native HD video frames could reduce compression-related artifacts and preserve fine visual details critical for ball detection.

Moreover, given that the system operates on video streams, temporal information can be exploited. For instance, simple motion-based heuristics or interpolation techniques could be used to correct inconsistencies (e.g., detecting if the ball ”jumps” implausibly between frames, suggesting a missed or incorrect detection).

Once a robust object detection backbone is in place, additional modules can be incorporated to extend the system’s functionality, including, for example,:

- Field keypoint detection: identifying fixed landmarks such as penalty boxes, sidelines, or the center circle to allow for spatial context analysis and assist in determining whether a player or ball is inside or outside the field of play.
- Event detection: combining spatial and temporal data to identify high-level game events such as passes, shots, tackles, or offsides.

Integrating these capabilities would support the development of a more comprehensive, automated analysis framework for football videos, enabling applications in broadcasting, coaching, and sports analytics.

References

[1] K. Fujii, “Computer vision for sports analytics,” in *Machine Learning in Sports*, ser. SpringerBriefs in Computer Science. Singapore: Springer, 2025. [Online]. Available: https://doi.org/10.1007/978-981-96-1445-5_2

[2] A. B. Rashid and M. A. K. Kausik, “Ai revolutionizing industries worldwide: A comprehensive overview of its diverse applications,” *Hybrid Advances*, vol. 7, 2024.

[3] B. T. Naik, M. F. Hashmi, and N. D. Bokde, “A comprehensive review of computer vision in sports: Open issues, future trends and research directions,” *Applied Sciences*, vol. 12, no. 9, 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/9/4429

[4] P. J. Figueroa, N. J. Leite, and R. M. L. Barros, "Tracking soccer players aiming their kinematical motion analysis," *Computer Vision and Image Understanding*, vol. 101, no. 2, pp. 122–135, 2006.

[5] J. Gudmundsson and T. Wolle, "Football analysis using spatio-temporal tools," in *Proceedings of the 20th international conference on advances in geographic information systems*, 2012, pp. 566–569.

[6] J. Eichner, J. Nowak, B. Grzelak, T. Górecki, T. Piłka, and K. Dyczkowski, "Advanced vision techniques in soccer match analysis: From detection to classification," in *VISIGRAPP (VisAPP) 2025*, vol. 3, 2025, pp. 808–815.

[7] P. Andrews, N. Borch, and M. Fjeld, "Footyvision: Multi-object tracking, localisation, and augmentation of players and ball in football video," in *Proceedings of ICMIP*, 2024.

[8] Q. Zhang, L. Yu, and W. Yan, "Ai-driven image recognition system for automated offside and foul detection in football matches using computer vision," *International Journal of Advanced Computer Science and Applications*, vol. 16, no. 1, 2025.

[9] S. Giancola, M. Amine, T. Dghaily, and B. Ghanem, "Soccernet: A scalable dataset for action spotting in soccer videos," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 1711–1721.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *NeurIPS*, 2015.

[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," *ECCV*, 2016.

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *CVPR*, 2016.

[13] G. Jocher *et al.*, "Yolov5 by ultralytics," 2023, https://github.com/ultralytics/yolov5.

[14] H. Liu, F. Sun, J. Gu, and L. Deng, "Sf-yolov5: A lightweight small object detection algorithm based on improved feature fusion mode," *Sensors*, vol. 22, no. 15, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/15/5817