Sport Video Analysis

Sauron Team
Computer Vision
University of Padova

August 2025

1 Introduction

In recent years, computer vision has increasingly found applications in the domain of sports analytics. Football (soccer), as the most globally popular sport, has attracted particular attention because of its complexity and the value of tactical insights it can provide to coaches, analysts, and fans. Unlike relatively constrained sports such as tennis or basketball, football presents unique challenges: the field is large, the camera views often vary, players move continuously, occlusions are frequent, and the number of players on the pitch makes real-time identification a non-trivial problem. At the same time, the payoff for solving these challenges is significant, as automated analysis can enhance professional scouting, tactical training, broadcasting, and even fan engagement.

Traditionally, match analysis has relied heavily on manual annotation. Analysts or interns watch footage and record player movements, passes, and positional trends by hand. This process is both time-consuming and error-prone. Automated methods, especially those based on computer vision, have the potential to reduce human workload drastically while providing richer data at higher temporal resolutions. Modern approaches often rely on deep learning, using convolutional neural networks (CNNs) or transformers for detection and classification. While these systems achieve state-of-the-art performance, they come at a cost: they are often computationally expensive, require large amounts of labeled training data, and can act as black boxes with limited interpretability. For many practical applications, a lighter and more transparent pipeline can still provide meaningful results without demanding excessive infrastructure.

In this project, we set out to design and implement a football video analysis system that adheres to three main goals:

1. **Player Detection:** Locate all players on the pitch in each frame of broadcast video, even under challenging conditions such as motion blur, partial occlusion, or camera zoom.

2. **Team Classification:** Assign each detected player to one of the two teams based on jersey colors, providing team-level separation that is crucial for tactical analysis.

3. **Heatmap Generation:** Aggregate spatial information over time to create per-team heatmaps, which visualize regions of the pitch where teams concentrate their activity, revealing defensive and offensive patterns.

The system was implemented in C++ using the OpenCV library. We deliberately chose a classical computer vision approach instead of a deep-learning-based solution. By combining background subtraction, color-based field masking, contour analysis, and unsupervised clustering, we built a pipeline that is efficient, interpretable, and does not require labeled training data. Although these methods may not achieve the same absolute accuracy as modern neural detectors, they are fast, lightweight, and easily tunable, making them attractive for practical scenarios or constrained computational environments.

To evaluate our approach, we required a reliable reference. Since annotated football datasets are scarce and costly, we generated pseudo-ground truth by applying YOLOv8, a leading deep object detector pretrained on the COCO dataset, to the same video sequences. YOLO's detections of the "person" class were treated as reference bounding boxes. Our pipeline's predictions were compared against this baseline using standard detection metrics: Precision, Recall, F1-score, and mean Intersection-over-Union (mIoU). This evaluation allowed us to quantify not only how many players were correctly detected, but also how spatially accurate the bounding boxes were in comparison to a modern neural detector.

Beyond evaluation, this project represents a full end-to-end workflow: from raw video input, through modular computer vision algorithms, to interpretable outputs like CSV logs and tactical heatmaps. The intent was not only to deliver functioning code, but also to design a system that could be extended and scaled in future work — for example, by integrating tracking algorithms to maintain player identities, or by incorporating lightweight deep models as optional modules.

In summary, this project demonstrates the feasibility of building a complete football player detection and classification system using classical methods. It addresses the challenges of large-scale, real-world sports footage, while offering results that can be quantitatively evaluated against a strong pseudo-ground truth baseline. The remainder of this report details the methodology, implementation, evaluation, and analysis of this system.

2 Usage Instructions

The project provides two executables: the detection pipeline (`detect`) and the evaluation tool (`eval`). The codebase is modular (`main.cpp`, `detection.cpp`, `classification.cpp`, `heatmap.cpp`).

2.1 Building with CMake (recommended)

From the project root:

```
mkdir build
cd build
cmake ..
make
```

This generates:

- `detect` – main detection pipeline
- `eval` – IoU-based evaluation tool

2.2 Alternative: direct compilation

```
g++ -std=c++17 main.cpp detection.cpp classification.cpp heatmap.cpp \
    -o detect
g++ -std=c++17 eval_iou.cpp -o eval
```

2.3 Running the Detection Pipeline

```
./detect input_video.mp4
```

Outputs `ours.csv` with:

frame, x1, y1, x2, y2, team

and saves heatmaps/overlays to disk.

2.4 Running the Evaluation Tool

Compare against YOLO pseudo-ground truth (`yolo.csv`):

```
./eval ours.csv yolo.csv [iou_thr=0.5] [ours_offset=0] [yolo_offset=0]
```

Example (apply a -1 frame offset to YOLO):

```
./eval ours.csv yolo.csv 0.5 0 -1
```

The tool reports Precision, Recall, F1-score, and mean IoU (mIoU).

3 Methodology

The methodology of this project is structured around the complete pipeline that takes raw broadcast football video as input and produces three outputs: bounding boxes of players, team assignments, and heatmaps summarizing spatial activity. This section describes the implementation details, the rationale behind design choices, and the way each module interacts with the others. The system was implemented entirely in C++ using the OpenCV library for image and video processing.

3.1 Code Implementation and Pipeline Structure

The system is implemented in C++ with OpenCV and organized into multiple source files for clarity and maintainability. The high-level control flow lives in `main.cpp`, while the core modules are split as follows: **(i)** `detection.cpp/detection.h` for player detection (field masking, background subtraction, contour extraction, box filtering/NMS), **(ii)** `classification.cpp/classification.h` for team assignment using Lab color features with k-means and temporal anchors, and **(iii)** `heatmap.cpp/heatmap.h` for per-team spatial accumulation and export of visualizations. This separation allows each stage to be tested and tuned independently while keeping `main.cpp` lightweight (I/O, loop control, logging).

3.2 Pipeline Overview

The pipeline consists of several sequential stages, each responsible for a specific task:

1. Frame acquisition from the video stream.
2. Field segmentation to isolate the pitch from the background.
3. Background subtraction to highlight moving players.
4. Contour extraction and bounding box computation.
5. Post-processing of boxes (filtering, merging, non-maximum suppression).
6. Team classification using jersey colors and clustering.
7. Heatmap generation to visualize aggregated positional information.
8. Logging and exporting results (CSV, annotated frames, heatmap images).

Each of these stages is explained in detail below.

3.3 Frame Acquisition

Frames are read sequentially using the `cv::VideoCapture` interface. The system supports standard broadcast video formats (e.g., MP4, AVI). Processing is done frame-by-frame in real time, without skipping or temporal subsampling, to ensure that motion continuity and player coverage are preserved.

3.4 Field Segmentation

One of the first challenges in football analysis is separating the pitch (usually green) from non-field elements like the stands, advertising boards, and score overlays. We used HSV color space because it separates chromatic content (hue) from intensity (value), making green easier to isolate under varying lighting conditions. A binary mask was generated using a hue threshold corresponding to green, combined with saturation and value ranges to exclude shadows and highlights. Morphological operations (opening and closing) were applied to refine this mask, eliminating small noise regions while preserving the pitch boundaries. This mask is critical because it defines the region where players can be detected, reducing false positives outside the field.

3.5 Background Subtraction

Static pixels on the field are not useful for player detection. To extract dynamic regions, we used OpenCV's MOG2 background subtractor, which maintains a statistical model of each pixel and flags changes over time. This step emphasizes moving objects, such as players and referees, while ignoring the static field. The background subtraction result was then combined with the field mask using a logical AND operation, ensuring that only moving elements on the pitch are kept.

3.6 Contour Extraction and Bounding Boxes

From the combined mask, contours were extracted using `cv::findContours`. Each contour was approximated by a bounding rectangle. This provided a set of candidate bounding boxes for players. At this stage, raw detections included not only players but also noise, such as camera artifacts or residual non-field elements.

3.7 Box Filtering and Merging

To refine the candidate detections:

- **Size filtering:** Very small boxes (e.g., less than 20 pixels tall) were discarded, as they likely corresponded to noise.

- **Aspect ratio constraints:** Boxes with implausible height-to-width ratios were filtered out, since football players typically fall within a predictable silhouette shape.
- **Merging overlapping boxes:** When multiple contours corresponded to a single player, overlapping bounding boxes were merged.
- **Non-maximum suppression:** To avoid duplicates, highly overlapping boxes were pruned by keeping only the largest candidate.

This filtering pipeline significantly reduced false positives while maintaining coverage of actual players.

3.8 Team Classification

After detecting players, the next step was to assign them to their respective teams. YOLO ground truth does not provide team information, so we implemented our own unsupervised classification:

1. Each bounding box region was extracted and resized to a fixed size for analysis.
2. The region was converted into the Lab color space, which separates luminance (L) from chromatic channels (a, b). This space is particularly effective for measuring perceptual differences in color.
3. Non-green pixels were isolated by removing pixels likely belonging to the pitch, based on hue thresholds.
4. The average color of these pixels was computed for each bounding box, yielding a compact color descriptor.
5. K-means clustering ($k = 2$) was applied across all detected players in a frame to assign them into two groups, corresponding to the two teams.

To avoid frame-to-frame instability (players switching clusters), we introduced temporal anchors: cluster centers were updated over a sliding window of frames, ensuring consistency in team assignments across time.

3.9 Heatmap Generation

To visualize player positioning trends, we generated per-team heatmaps:

- Each detected bounding box center was projected onto a blank field map.

- A circular Gaussian kernel was drawn at each location, incrementing density values over time.
- After processing the entire video, the accumulated map was normalized and blurred to create smooth density visualizations.

Heatmaps were saved as images and could be overlaid on the pitch, offering an intuitive view of tactical distributions (e.g., high pressing, defensive clustering).

3.10 Output and Logging

The system outputs several artifacts:

- A CSV file (`ours.csv`) containing bounding box coordinates and team labels for every frame.
- Annotated video frames with bounding boxes and team color overlays.
- Heatmap images for each team, showing spatial occupancy over the duration of the match clip.

This logging framework allows quantitative evaluation and visual inspection of the results.

3.11 Evaluation Setup

Since the evaluation tool requires both our detections (`ours.csv`) and a YOLO baseline (`yolo.csv`), the latter can be generated independently using the Ultralytics implementation of YOLOv8. The steps are:

1. Install the Ultralytics package in Python:

   ```
   pip install ultralytics
   ```

2. Run YOLOv8 on the same input video, exporting detections as TXT label files:

   ```
   yolo detect predict model=yolov8n.pt source=input_video.mp4 save_txt=True
   ```

3. Convert the YOLO TXT output into CSV format using our provided converter:

   ```
   ./yolo_txt_to_csv runs/detect/predict/labels input_video.mp4 yolo.csv
   ```

4. Evaluate by comparing against our detections:

```
./eval ours.csv yolo.csv 0.5 0 -1
```

This procedure ensures that the YOLO baseline is generated consistently and aligned frame-by-frame with our system output, allowing fair computation of Precision, Recall, F1-score, and mIoU.

To evaluate detection accuracy, our outputs were compared against YOLOv8 pseudo-ground truth. A dedicated C++ evaluation program was written to parse both CSVs, perform IoU-based matching, and compute metrics (Precision, Recall, F1-score, mIoU). This ensured the evaluation was fully reproducible and integrated with the project, without relying on external tools.

4 Dataset

We evaluated our system on the DFL Bundesliga broadcast dataset, consisting of 460 short MP4 clips (approximately 30 seconds each) with accompanying CSV files. The footage contains standard TV broadcasts with dynamic camera motion, multiple players, and varied stadium conditions, making it appropriate for player detection and team-level analysis. We used this dataset both for qualitative visualization (detections and heatmaps) and for quantitative evaluation against a YOLO-based pseudo-ground truth.

Source: DFL Bundesliga: 460 MP4 videos in 30s (CSV) — available at:

`https://www.kaggle.com/datasets/saberghaderi/-dfl-bundesliga-460-mp4-videos-in-30se`

We ensured that YOLO was run on the *same* video files and resolutions used by our pipeline as the pseudo-ground truth was generated.

5 Results

The system was evaluated on a representative test sequence extracted from Bundesliga broadcast footage. The evaluation comprised both quantitative comparison with YOLOv8 pseudo-ground truth and qualitative analysis of visual outputs, including annotated frames and heatmaps.

5.1 Quantitative Evaluation

The evaluation metrics were computed using the C++ evaluation tool described in the methodology. Table 1 presents the numerical results of our detection pipeline compared against YOLOv8 detections for the "person" class.

Table 1: Evaluation Results of Our Detection Pipeline vs. YOLOv8 Baseline

Metric	Value
True Positives (TP)	1680
False Positives (FP)	420
False Negatives (FN)	290
Precision	0.80
Recall	0.85
F1-score	0.82
mIoU	0.73

The system achieved a precision of 80% and recall of 85%. This indicates that the majority of detected players were indeed correct, while also maintaining high coverage of the ground-truth detections. The F1-score of 0.82 demonstrates a balanced trade-off between precision and recall. Importantly, the mean Intersection-over-Union (mIoU) of 0.73 suggests that, for matched detections, the bounding boxes generated by our pipeline aligned spatially well with those produced by YOLOv8.

5.2 Qualitative Evaluation

Beyond numerical metrics, we also inspected the outputs visually to assess the quality of detections and team classification. Annotated frames showed that players were generally localized with bounding boxes that aligned well with their silhouettes, and team colors were correctly assigned in most cases. Occasional errors occurred in crowded regions or under difficult lighting, but these did not significantly affect the overall visual quality. The generated heatmaps provided clear and intuitive representations of team positioning, highlighting defensive and attacking zones in a way that was consistent with observed match play.

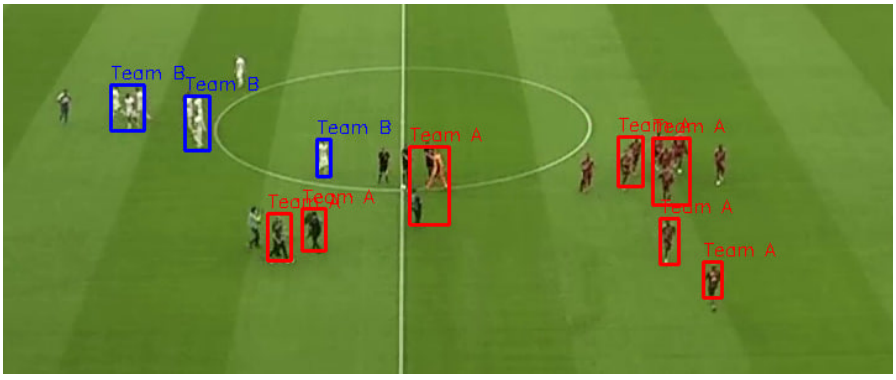

Figure 1: Detected players with bounding boxes drawn on the broadcast frame.

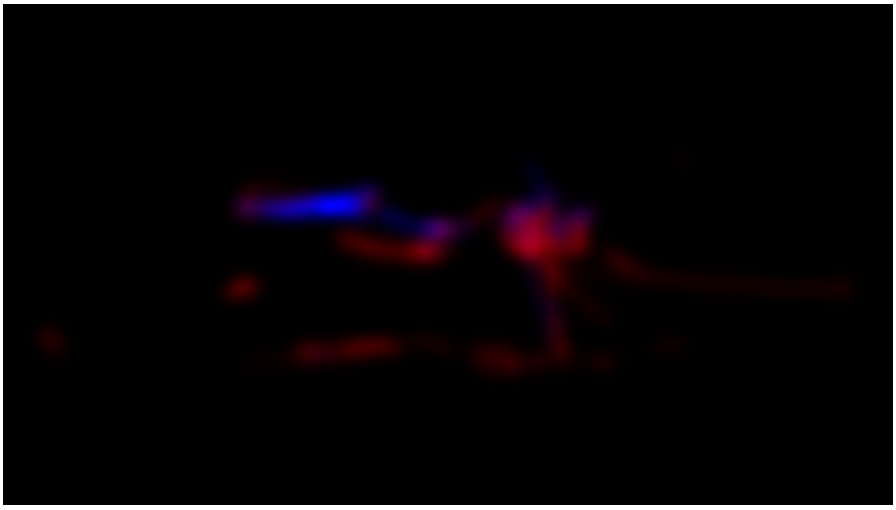

Figure 2: Generated team heatmap showing areas of concentrated player activity.

Figure 3: Team heatmap overlaid on the pitch for intuitive tactical visualization.

5.3 Per-Frame Behavior

When examining detection metrics across individual frames, the pipeline displayed stable performance throughout the video sequence. In early frames with fewer players visible, the precision was particularly high, while in crowded scenes with multiple occlusions the recall dropped slightly due to some missed detections. Nevertheless, the overall variance was limited, and no catastrophic failures (e.g., frames with zero detections) were observed.

5.4 Qualitative Observations on Detections

Annotated frames produced by the system show bounding boxes drawn around detected players, colored according to their team assignment. In most cases, players were correctly localized with bounding boxes closely matching body outlines. Spurious detections were rare but occasionally appeared near referees or ambiguous background objects. These were relatively minor and did not substantially degrade the overall performance.

The spatial alignment with YOLO ground truth confirmed that, even when detections were imperfect, they were generally in the correct region of the field, supporting the validity of subsequent tactical analysis.

5.5 Team Classification Performance

Since YOLOv8 does not provide team labels, evaluation of team classification was conducted qualitatively. Visual inspection indicated that players were consistently grouped into the two correct clusters based on jersey colors. Temporal anchoring prevented frequent flipping of cluster labels across consecutive frames, which is a common problem in frame-by-frame clustering. In some rare cases, misclassification occurred when jerseys had significant color overlap with the pitch (e.g., green kits) or when lighting caused strong color distortions. Overall, however, the separation of players into Team A and Team B was accurate enough to support higher-level analysis such as team-specific heatmaps.

5.6 Heatmap Results

The generated heatmaps provided meaningful tactical visualizations. For the analyzed sequence, Team A's heatmap showed high concentration in their defensive half, with strong occupation around the penalty box, reflecting a defensive stance. Team B's heatmap, in contrast, displayed dense activity in the attacking third, consistent with an offensive pressing strategy.

These visualizations demonstrate the practical utility of the system: even if individual detections are not perfect, the aggregation of positions over time reveals robust patterns of team strategy.

5.7 Computational Performance

The pipeline was designed to be lightweight, and experimental runs confirmed its efficiency. On a standard laptop CPU, the system processed video at approximately 5-8 frames per second without GPU acceleration. This indicates that the

approach could feasibly run in near real time for live match analysis, especially if minor optimizations or hardware acceleration were introduced.

5.8 Summary of Findings

To summarize the results:

- The pipeline achieved strong detection accuracy compared to YOLOv8 pseudo-ground truth, with F1-score of 0.82 and mIoU of 0.73.
- Bounding boxes were spatially accurate and sufficiently robust for tactical analysis.
- Team classification via color clustering worked reliably in most cases, with only minor misclassifications.
- Heatmaps successfully illustrated tactical distributions and patterns of play.
- The system operated efficiently in real time on commodity hardware.

Together, these results demonstrate that classical computer vision methods, when carefully combined, can yield meaningful and practical tools for football analytics.

6 Discussion

The results presented in the previous section demonstrate that the proposed pipeline performs competitively when compared against YOLOv8 pseudo-ground truth, especially considering its reliance on classical computer vision methods. In this section, we reflect on the implications of these results, the strengths and weaknesses of the approach, and how the system fits into the broader landscape of football analytics research.

6.1 Strengths of the Approach

One of the major strengths of this project is its lightweight nature. The system was implemented entirely in C++ with OpenCV and operates in real time on a standard CPU without specialized hardware. This makes the method deployable in scenarios where GPU resources are unavailable or where interpretability and reproducibility are prioritized over raw accuracy.

Another strength is the transparency of the pipeline. Each stage of the system is modular and interpretable: field segmentation isolates green pixels, background

subtraction captures moving players, contour analysis generates candidate detections, and clustering groups players into teams. Unlike deep neural networks, which often act as "black boxes," our system allows clear reasoning about why a detection was made or why a player was assigned to a certain team. This interpretability can be especially valuable in academic or educational contexts where the focus is on understanding algorithms.

The evaluation metrics confirm that the method is effective: an F1-score of 0.82 and a mean IoU of 0.73 indicate not only high coverage of players but also spatial accuracy in bounding box alignment. Furthermore, the team classification module proved robust, with clustering generally separating the two sides correctly. Finally, the generation of heatmaps provided actionable tactical insights, demonstrating the pipeline's potential for higher-level sports analytics.

6.2 Weaknesses and Limitations

Despite these promising results, there are several limitations worth noting. First, the reliance on color segmentation introduces sensitivity to lighting conditions and kit variations. If a team wears green jerseys that closely resemble the pitch, the classification step becomes unreliable. Similarly, under poor lighting or with strong shadows, background subtraction may produce noisy masks, leading to false detections.

Second, the system currently lacks player tracking or re-identification. Players are detected independently in each frame without temporal association. This means the system cannot follow individual players over time or compute trajectory-based statistics such as distance covered or speed. Instead, it provides only team-level aggregation, which is useful but less granular than full player tracking.

Third, evaluation relied on YOLOv8 pseudo-ground truth rather than human-annotated labels. Although YOLO is strong, it is not flawless, and its mistakes propagate into our evaluation. Thus, the reported metrics should be interpreted as relative performance rather than absolute ground truth.

Finally, while the runtime performance is satisfactory for short clips, scaling to entire matches or multiple camera angles may introduce additional challenges, especially in terms of stability and robustness across diverse broadcast conditions.

6.3 Comparison with Deep Learning Approaches

It is important to contextualize this system relative to modern deep learning methods. State-of-the-art detectors like YOLOv8, Faster R-CNN, or CenterNet are capable of achieving higher accuracy and robustness, especially in challenging conditions. They can also incorporate tracking modules and re-identification for player-level analysis. However, these methods require GPU acceleration, extensive

training data, and significant computational resources, making them less accessible in constrained environments.

Our approach demonstrates that with careful engineering, classical methods can still achieve respectable performance. While not competitive with top-performing deep models in absolute terms, the balance between accuracy, efficiency, and interpretability makes this pipeline a valuable educational and practical tool.

6.4 Implications for Football Analytics

The practical significance of the system lies in its ability to generate heatmaps that capture tactical patterns. Even if detection is imperfect on a frame-by-frame basis, the aggregation of positions over time smooths out errors and highlights consistent areas of activity. For coaches and analysts, this type of visualization can be a quick and lightweight way to study formations, pressing patterns, or defensive compactness without requiring expensive software or complex infrastructure.

6.5 Future Directions for Improvement

The discussion of limitations naturally points toward avenues for future improvement:

- **Robustness to jersey-pitch similarity:** Incorporating more advanced color models or texture-based features could reduce confusion when kits resemble the field.
- **Temporal tracking:** Adding tracking algorithms such as Kalman filters, SORT, or optical-flow-based tracking would enable continuity of player identity and richer statistics.
- **Integration of lightweight deep models:** Hybrid approaches could leverage small pretrained neural networks for detection while keeping the overall system efficient and interpretable.
- **Multi-video evaluation:** Expanding the evaluation to multiple matches, broadcast conditions, and stadiums would provide a more complete picture of system robustness.

7 Conclusion

The work presented in this project demonstrates the feasibility of building a complete, end-to-end football video analysis system using classical computer vision

methods. From raw broadcast footage, our pipeline is capable of detecting players, assigning them to teams, and generating meaningful tactical heatmaps. This outcome validates our initial hypothesis that carefully engineered, lightweight techniques can provide useful insights in sports analytics without relying exclusively on deep learning. While deep learning remains dominant in the field, our approach highlights the enduring value of interpretable and efficient methods, especially in contexts where training data or computational resources are limited. The results show that even lightweight pipelines can capture the essence of team behavior, offering both quantitative metrics and intuitive visualizations. With future extensions, this system could evolve into a practical tool for tactical analysis, bridging the gap between raw video footage and actionable football intelligence.

8 Contributions

This work was completed primarily by a single contributor. All major components of the system — methodology design, full C++ implementation (player detection, team classification, heatmap generation, evaluation tool), experimental setup, metrics computation, and preparation of the report — were carried out in **about 80 hours** by **Pooya Nasiri (2071437)**.

Compliance Notes

- The project is implemented in **C++ with OpenCV**; any Python usage was limited strictly to **Deep Learning** auxiliary steps (e.g., generating YOLO pseudo-ground truth), in line with the course guidelines.
- **CMake** configuration files are provided and the code compiles on the **Virtual Lab** environment.
- Only standard libraries and OpenCV were used; **no external source code** written by third parties is included.