Modular pipeline for football performance analysis: detection, tracking and valuation of actions

Author : Jean Grifnée

Promoters : Marc Vandrogenbroeck & Anthony Cioppa

Academic year 2024-2025

Contents

List of Tables

List of Figures

1 Introduction

Well before being a subject of study, football is first and foremost one of my passions. And where there's passion, there's also emotion, irrationality and endless discussion where bad faith often vies with chauvinism. Contrary to the caricatures that its detractors want to make of it, football is also the sum of a multitude of dynamic and contextual variables, influencing each game and even each action. All of a sudden, what may seem clear-cut to the casual football fan reveals all its complexity, and we discover that an analysis based on simple indicators such as goals or assists - traditional statistics if ever there was one - proves inadequate to offer anything other than a partial - and therefore misleading - view of the real contribution of each player to the team's performance.

As I said earlier, football is first and foremost a highly emotional sport, capable of arousing deep passion in players, coaches and even more so in supporters. I'd like to invite anyone who claims otherwise to the refreshment bar for a "total immersion" session!

To what extent can statistics provide analytical rigour to complement the necessarily subjective experience of watching a football match? We say complement, not replace, because, as you will have understood, in our view, enjoyment must remain the driving force behind sport. Our wager is that this pleasure can be enhanced by appropriate statistics, providing a basis for rational reflection that can overcome emotional bias.

And then the complexity catches up with us. An analysis based solely on raw statistics is not enough to grasp all the richness and subtlety of the game. A pertinent analysis of performance must combine quantitative analysis with qualitative observation of the match. Statistics make it possible to objectify certain intuitions and detect trends that may escape the human eye. They reveal, for example, recurring patterns in a player's movements or passing choices, thereby enriching tactical analysis. But they are partial; they don't tell us everything.

I discovered football at a time when the only statistics available could be described as rustic. Under the astonished eye of my father, who considered them sacrilegious in that they interrupted his perception of the fluidity of the game, the football statistics of those years were limited to simple indicators such as the number of passes, shots or tackles. These measurements provided basic information about individual actions, but were insufficient to measure their real impact on the outcome of the match. The arrival of advanced probabilistic metrics, such as Expected Goals (xG), has, on the one hand, lost my father for good and, on the other, profoundly changed the landscape of performance analysis. The xG, for example, estimates the probability of a shot resulting in a goal by taking into account a number of contextual factors: position, angle, distance, type of previous pass, etc.

These new statistics have a number of benefits:

- More accurate evaluation: they take into account the quality of the opportunities created

or suffered, and not just the final result. the role of chance: they better distinguish real performances from random or accidental events Helping with strategic decision-making: they enable coaches to identify effective or vulnerable areas and adjust their game plan. objective comparisons between players: they provide reliable criteria for recruitment, training and performance analysis.

Despite its qualities, the xG has several important limitations:

- Concentration exclusively on shooting: it ignores all other key actions such as passing or defensive interventions. of defensive context: it does not take into account actions that prevent opponents from scoring.
- Failure to consider movement without the ball: it neglects the impact of intelligent movements that create space or disorganise the defence.

In order to free ourselves from our dependence on specialist data providers, but also to broaden our field of action, it is essential to automate the analysis of actions and positions, so as to enable a detailed, contextualised assessment of performance, whatever the level of competition. It is therefore necessary to develop a tool capable of analysing matches from simple videos, and therefore to mobilise computer vision.

Unlike some advanced metrics that require proprietary data, computer vision allows any match to be processed as soon as it is filmed. This paves the way for universal analysis, even in lower divisions or less professional contexts.

As well as detecting actions (passes, shots, etc.), computer vision continuously tracks players' movements. This provides a better understanding of tactical phases, outnumbered situations and defensive pressure. Once the detection and tracking algorithms have been configured, the analysis can be fully automated. This saves a considerable amount of time, standardises the assessment and reduces human bias.

What's more, while access to advanced match data (GPS positions, detailed events) is often costly and reserved for professional clubs, computer vision, by processing standard videos directly, makes it possible to free oneself from this dependence while maintaining a high quality of analysis. élevée.

This thesis is part of this approach. The aim is to design and evaluate a complete analysis chain capable of automatically extracting the significant actions of a match from a simple broadcast video, and then estimating their tactical and contextual value. To do this, the system developed combines several modules: event detection, player tracking, re-identification, camera calibration and spatial projection. The ambition is to build a robust, reproducible tool capable of analysing the game without relying on proprietary data, and so pave the way for a finer, freer and fairer valuation of football.

May the football gods forgive my boldness. But let them also be reassured: no data will ever

eclipse a Courtois save; no statistic will replace a cutting pass from De Bruyne; no analysis will surpass a Lukaku goal; and finally, no Excel table will break the magic of a penalty shoot-out, provided it is ultimately won by Argentina against France in the World Cup Final.

2 Previous Work

2.1 Introducion

This chapter reviews the main components of video-based football analysis. We first examine methods for action spotting, then focus on player tracking, and finally discuss approaches to action valuation. Each section highlights key challenges and recent advances in the field.

2.2 Ball Action Spotting

2.2.1 Definition

Event spotting (or action spotting) refers to the task of identifying specific actions within a long, unedited video and then locating them precisely in time using a single timestamp [15]. In other words, it involves automatically detecting the exact moments when actions of interest (goals, cards, etc.) occur in a filmed match. This fine-grained detection is crucial for many sports applications, such as statistical game analysis, video analyst assistance, and automatic highlight generation for fans [15]. Before 2018, there were no large-scale datasets available to train and evaluate this type of system, which motivated the creation of the SoccerNet database dedicated to this problem [14].

The concept of Precise Event Spotting (PES). In the context of sports, the events to be spotted are often instantaneous and must be detected with very high temporal accuracy. Recently, Hong *et al.* introduced the notion of Precise Event Spotting (PES) to emphasise this requirement for absolute precision in the localisation of actions [17]. PES aims to predict the event exactly to the frame where it occurs, clearly distinguishing the decisive moment of an action. For example, a goal in football must be signalled at the precise moment when the ball crosses the goal line. This concept is accompanied by stricter evaluation metrics (tolerance of only a few frames), in order to test the ability of models to stick to the exact moment of the action rather than a broad time window [17]. The introduction of PES highlighted that many previous action detection models (from the literature on classical temporal action detection) were not accurate enough for this level of requirement, hence the need to develop approaches specific to ultra-precise spotting.

2.2.2 Historical approaches to action spotting.

Several deep learning approaches have emerged since the creation of SoccerNet to solve the problem of action spotting. Historically, several families of methods have marked the state of the art:

CNN and sliding window approaches. The first methods formulated spotting as a classification problem on sliding time segments. Typically, visual descriptors were extracted using a convolutional neural network (CNN) pre-trained on images (e.g., a ResNet or I3D), and these descriptors were then summarised over a short time window to predict the presence of a target action in the centre of the window [14]. For example, the basic SoccerNet-v1 method used a NetVLAD aggregation layer to compress the features of a sequence of a few seconds, followed by a classifier indicating whether an action (goal, foul, etc.) occurred in that time frame [14]. This purely sliding window approach laid the foundations for the field, demonstrating that it was possible to identify key actions with pre-computed visual features and segment-by-segment classification.

RNN/GRU sequential models. To go beyond the local window and incorporate temporal context, the following work introduced RNN-type sequential models. By inserting, for example, a Gated Recurrent Unit (GRU) or an LSTM above the features extracted by the CNN, the model can accumulate information over long sequences and better take into account the sequence of events before and after the key moment. Cioppa et al. proposed a 'contextual' cost function that rewards correct detection at the annotated time while temporally penalising false detections that are too far away, implicitly exploiting the context around each event [6]. Adding context via RNNs or similar mechanisms has proven beneficial, significantly improving Average-mAP compared to purely local CNNs [6]. Nevertheless, these recurrent models can be difficult to optimise on very long videos and remain limited by gradient propagation over time.

Multi-scale temporal convolutions (TCNs). In parallel, temporal convolution approaches have been explored to capture dependencies across multiple time scales. Farha et al. introduced the MS-TCN (Multi-Stage Temporal Convolutional Network) model for action segmentation [12], and this idea of sliding 1D convolutions has been adapted to event spotting. A TCN can scan the video sequence with temporal filters of different lengths, detecting dynamic patterns corresponding to the target actions. Multi-scale and multi-stage variants have gradually refined temporal predictions. For example, work has integrated multi-layer TCNs to aggregate game information over several seconds while maintaining the temporal resolution necessary for accurate localisation [12]. These purely convolutional models often offer a good compromise between long-term modelling capability and ease of training (no recursive dependency), making them a popular component of many spotting pipelines around 2019–2021.

Anchor-based methods. More recently, some methods have approached spotting from the perspective of object detection, transposing the notion of anchors to the temporal dimension. These anchor-based approaches generate multiple proposals for time intervals (or directly candidate timestamps) around the match timeline, then learn to classify these proposals and refine their location. For example, Soares and Shah proposed a dense anchor system that regularly places anchors in the video and predicts for each one an action and a possible time shift to apply to match the event [31]. This type of model typically has two heads: a classification head (which action category) and a temporal regression head (to precisely adjust the predicted time) [31]. The anchor approach has the advantage of explicitly traversing the temporal dimension in search

of events, similar to object detection in space, which has improved fine-grained action detection in some recent competitions. We have also seen the emergence of transformer architectures (e.g., SpotFormer) combining this idea of temporal anchors with global attention to further improve accuracy [4].

2.2.3 Difficulties of spotting in football videos.

Spotting actions in real football matches presents several major challenges. First, actions of interest are infrequent and unevenly distributed over the 90 minutes of play: this creates a significant class imbalance (a very small number of positive frames for a huge number of negative ones) as well as a very uneven frequency distribution between types of actions (many touches or shots, but very few penalties or red cards) [30]. Second, the annotations refer to a single moment in time with no time interval: this means that two consecutive images that are visually very similar can be labelled differently (just before the action vs. the moment of the action) [6]. This subtle feature makes the task difficult, as the model must distinguish between minute differences in order not to miss the exact moment. In addition, sports broadcast videos contain production cuts (replays, camera changes) that can result in actions not being visible on screen. For example, a foul may be called while a replay is in progress, making the event invisible in the main video stream [9]. Finally, there is some annotation noise and variability in how the exact timestamp of an event is defined (e.g., the moment of a goal may be annotated at the decisive touch of the ball or when it crosses the line, depending on the annotators). This variability complicates supervised learning. Recent work incorporates mechanisms to manage these uncertainties, for example by learning a probability distribution on the location of the action rather than a deterministic point [40]. In summary, the complexity of spotting in football lies as much in the rarity and brevity of the events to be detected as in the vagaries of the video content itself (camera angles, occlusions, etc.).

2.2.4 Spotting (and PES) in SoccerNet.

The SoccerNet platform has played a central role in the development and evaluation of action spotting methods. The first version of the dataset, SoccerNet-v1 (2018), included more than 500 complete matches annotated with key actions (goals, cards, penalties, substitutions, etc.), and established Action Spotting as an official benchmark [14]. Performance was measured using the mAP (mean average precision) metric, computed within a temporal tolerance window around the ground truth timestamp (e.g. ± 5 seconds) to account for acceptable slight delays [14]. The baseline published with SoccerNet-v1 used features extracted by a CNN (pre-trained on ImageNet and then fine-tuned on Kinetics) aggregated by NetVLAD, and achieved an initial mAP of approximately 49% [14]. In 2021, SoccerNet-v2 significantly enriched the framework: the number of action classes increased from 3 to 17, including more detailed events such as 'shot on target', 'goalkeeper save', 'throw-in', etc. [9]. In addition, additional annotations were provided for shot changes (cutting) and replay segments, making it possible to identify when a real action occurred off-screen [9]. These additions made it possible to evaluate not only the ability to detect actions, but also the robustness of the models in the face of the particularities of TV production. The annual SoccerNet challenges (CVPR Challenges, etc.) have encouraged the

community to propose new methods and have gradually introduced stricter evaluation criteria. For example, recent editions distinguish between 'loose' Average-mAP (wide tolerance) and 'tight' Average-mAP (reduced tolerance, close to PES) [40]. This pushes participants to optimise the temporal accuracy of their predictions. Thanks to SoccerNet and its competitions, more than 60 different methods were developed between 2018 and 2023, advancing the mAP rate from 50% to over 80% for the detection of major events [15]. SoccerNet now serves as the gold standard for comparing action spotting approaches, and its protocol (dataset, classes, PES metrics) has been adopted in other work, including on other sports for spotting specific events.

2.2.5 Feature-based vs. end-to-end approaches.

Another important area of development concerns how input data is processed. Historically, most spotting methods have adopted a two-step approach: first, visual features are extracted (by a network trained on a large database of generic videos), then these features are used for temporal classification. This feature-based approach has the advantage of reducing the computational load and allowing the use of prior knowledge (e.g., a CNN trained on Kinetics to capture general movements) [15]. For example, many participants in the early SoccerNet challenges used ResNet or I3D features extracted every half second, on which they trained their temporal model (RNN, TCN, etc.). However, relying on a frozen backbone can limit performance, as the features are not optimised to finely distinguish football actions.

Recently, there has been a transition towards end-to-end trained models on the spotting task itself. Hong et al. were among the first to propose a fully end-to-end model, called E2E-Spot, which integrates both feature extraction and action prediction in a single network optimised directly on precise spotting [17]. They show that by choosing a compact and suitable architecture (including a lightweight 2D CNN backbone with temporal fusion modules and a global temporal module of the transformer or GRU type), the entire model can be trained on long sequences while remaining manageable on GPUs [17]. The end-to-end model thus learns visual representations specific to football and its events, giving it an advantage over fixed-feature approaches.

Indeed, a jointly optimised network can detect subtle cues (camera movements announcing an action, player posture, etc.) that generic features would not have distinguished. Furthermore, recent work demonstrates that well-designed end-to-end training on SoccerNet can outperform state-of-the-art methods based on pre-computed features [30]. Examples include the arrival of spatio-temporal transformers trained from pixels, and the use of self-supervised learning strategies specific to sports videos to refine the backbone directly on SoccerNet data [10].

In this context, we adopt **T-DEED**[39], a recent end-to-end approach tailored for dense event spotting in football videos. T-DEED builds on the strengths of spatio-temporal modeling by combining a lightweight image encoder with a temporal architecture designed to process long untrimmed sequences. Its modular design allows for effective learning of football-specific representations while maintaining scalability on full matches. Unlike traditional pipelines that rely on detached feature extraction and classification steps, T-DEED is optimised globally to minimise spotting error, making it especially well-suited for fine-grained temporal localisation tasks. A detailed presentation of the model is given in Section **??**.

This evolution towards end-to-end learning comes with challenges (risk of overfitting, greater

need for annotated data), but it paves the way for more accurate spotting models that are more specialised in understanding the game in detail. Ultimately, we can expect most action spotting systems to adopt this integrated approach, taking full advantage of available data to maximise performance in Precise Event Spotting.

2.3 Tracking Players

2.3.1 Definition

Game State Reconstruction (GSR) refers to the task of reconstructing, from video sequences, the complete state of a match in real time in the form of a map of the field displaying the position and identity of each player (players, goalkeepers, referees) [32]. In other words, it involves extracting essential information about the dynamics of the game from sports videos: the 2D location of all athletes on the field (world coordinates on a top-down view), their role (player, goalkeeper, referee, etc.), their team affiliation, and their jersey number [32]. This data can then be visualised on a mini-map (*minimap*) of the field, providing a concise representation of the game in progress [32]. The goal is to provide a high-value overview for sports analysis, allowing, for example, the study of positioning, team tactics, or individual performance without having to equip players with sensors [16]. It should be noted that while football is the main sport studied in the GSR literature, this concept naturally extends to other team sports (basketball, hockey, etc.), with comparable principles applied to their specific playing fields [32].

2.3.2 Main challenges

GSR is a complex task that aggregates and extends several computer vision problems. Three major sub-problems must be solved simultaneously: (1) multi-player tracking from a monocular camera, (2) camera calibration to project trajectories onto the field coordinate system, and (3) accurate identification of each player (role, team, number).

First, single-camera tracking of players is made difficult by the particularities of match videos filmed in motion: limited fields of view (not all players are visible at all times), frequent camera movements, overlaps and occlusions between players, and high speeds of movement [16]. Unlike scenes from traditional video surveillance, players on the same team wear similar uniforms and constantly enter and exit the camera's field of view, "which complicates the association of trajectories over time and often results in trajectory fragmentation or identity switches, especially when using traditional multi-object tracking approaches [16]. Work has shown that tracking-by-detection algorithms (e.g., YOLO detectors coupled with DeepSORT) reach their limits in this context, due to the difficult-to-predict non-linear trajectories and the difficulty of re-identification (ReID) in a sport where players on the same team look very similar [16].

Secondly, automatic camera calibration (estimation of homography between the image and the field) is essential for expressing game positions in a uniform metric reference frame. This problem is difficult with a single view: the system must identify elements of the field (lines, circles, characteristic intersection points) in the image despite often partial or obstructed portions of the field [16]. Conventional methods either detect key points on the terrain (e.g., line intersections) to adjust a homography via RANSAC, or segment the lines on the terrain and then match them

to a known terrain model [16]. However, in TV broadcasting, it is not guaranteed that enough terrain landmarks will be visible at all times, which can lead to unstable calibration. More recent approaches use specialised neural networks that directly regress the camera parameters or refine the homography from images where few lines are visible [16].

Third, player identification is a challenge in itself: it is not only a matter of distinguishing between the two teams, but also of recognising each player individually by their shirt number, and possibly determining their role (goalkeeper, referee) based on their appearance or position. Team recognition can be based on the dominant colour of the kit: recent methods extract appearance descriptors (colour histograms, multi-image ReID embeddings, etc.) and then group players into clusters corresponding to the two teams and the referees [25]. For example, a classic method consists of calculating histograms or colour vectors for each detected player, then applying an unsupervised clustering algorithm (such as DBSCAN or k-means) to automatically separate the uniforms into two groups (home team vs. away team), and possibly a separate group for referees if necessary [16]. Other approaches avoid explicit clustering by directly training a model to produce discriminative team embeddings (associative embedding) [16].

Finally, automatic reading of jersey numbers is particularly difficult when players are far away, moving, and partially turned away. Adapted OCR techniques have been tested, combining text detection in the player's area followed by recognition by a character reading network [16]. For example, Balaji et al. propose detecting the digits visible on the players and then applying a Show, Attend and Read recognition model to read the number [2]. However, these generic approaches show their limitations in difficult conditions (low resolution, blurring, side view angle) [2]. Specialised solutions incorporating knowledge of the sports domain achieve better results: some models trained on sports data are able to directly classify the player's number in a single step (without going through traditional OCR) [2], while others exploit the player's pose (for example, by only attempting to read the number on images where the player's back is visible) coupled with temporal post-processing to increase reliability [2]. Despite these advances, real-time individual identification remains a bottleneck: a single error in reading a number or team cluster can cause the entire sequence to fail.

2.3.3 Recent work

As the GSR problem is relatively new (formalised in 2024 [32]), few studies have addressed it as a whole to date. Nevertheless, several recent works (2021–2024) have addressed components of GSR, mainly in the context of football, foreshadowing the convergence of these subtasks.

For example, Theiner et al. proposed a pipeline to project the position of each football player onto a top-down view from a monocular broadcast video, using camera calibration and a player detector [36]. This work demonstrated the feasibility of locating actors on a field minimap, but it did not take into account the temporal pairing of players (no multi-image tracking) or their individual identification [36].

Similarly, Maglo et al. developed a robust method for tracking players on football videos, including a field registration step by refining the detection model, which allowed them to place the estimated trajectories in the field reference frame [24]. Although paving the way for GSR, their approach was limited to tracking and spatial projection of players, without jersey identification,

due to the lack of annotated data to validate this component [24].

In addition, several teams focused on identification itself: for example, Mansourian et al. proposed a jointly trained multi-task learning network for player re-identification, team classification (jersey colour) and role classification (player vs goalkeeper vs referee) [25], showing that joint optimisation of these criteria improves overall tracking performance compared to models that treat these aspects separately. This type of approach illustrates the community's effort to go beyond the scope of classic Multi-Object Tracking (MOT) and tackle the semantic identity of players in the scene.

In terms of data resources, the SoccerNet initiative has greatly contributed to recent advances. After proposing SoccerNet-v2 and SoccerNet-Tracking, datasets comprising thousands of match sequences annotated with player detection and key actions [7], the authors of SoccerNet introduced a challenge dedicated to camera calibration from match videos in 2022 [7]. These efforts led to the publication of SoccerNet-GSR, the first open dataset annotated specifically for game state reconstruction [32].

SoccerNet-GSR provides synchronised, high-quality annotations for 200 30-second video clips: pixel trajectories of field lines (over 9 million annotated points) to facilitate calibration, 2D positions of all players and referees at each frame (over 2.3 million positions) with team IDs, role, and jersey number [32]. The introduction of this dataset in 2024, accompanied by a new evaluation metric (GS-HOTA) tailored to GSR, has set a first official benchmark for comparing approaches [32].

Somers et al. subsequently proposed an end-to-end baseline pipeline exploiting the best available methods for each subtask (YOLO detection, multi-object tracking algorithm combined with player re-identification, optimised OCR jersey number recognition, team colour classification, and automatic pitch calibration) [32]. This baseline, although perfectible, paved the way by obtaining the first quantitative results on the complete GSR task.

Since then, several research teams have tackled the SoccerNet-GSR challenge. For example, Golovkin et al. presented an optimised pipeline that combines a finely trained player detector (modified YOLOv5), a SegFormer-type homography estimator for continuous camera calibration, and a DeepSORT tracking module enhanced with improved re-identification and player orientation prediction, coupled with a jersey number reader [16]. Thanks to refinements such as intelligent trajectory fragment fusion (post-processing to rejoin short broken tracks) and improved temporal consistency, their method achieved first place in the GSR 2024 challenge with a GS-HOTA score of 63.8, far surpassing the other solutions submitted [16].

This competition highlights the importance of a modular and integrated approach to solving GSR, with each component (detection, tracking, calibration, identification) contributing crucially to the overall success.

Finally, it should be noted that similar work is emerging in other sports, demonstrating the cross-disciplinary interest in GSR. For example, for ice hockey, Prakash et al. (2024) proposed using a homographic projection of players onto a template of the rink seen from above, in order to better distinguish players during occlusions and thus improve tracking stability [28]. This idea of combining tracking and localisation on a standard map (in this case, the ice rink) is in line with the spirit of GSR and has shown significant gains in tracking hockey players in television

broadcasts [28]. This type of contribution, applied to a different sport, confirms the relevance of the game state reconstruction concept and suggests its future extension to many sporting contexts.

In conclusion, Game State Reconstruction is emerging as a new unifying challenge in sports vision, the complete resolution of which will make it possible to 'map' the entire game from simple videos, opening the door to analyses and applications at a level never seen before.

2.4 Valuing Action

General introduction

Recent work in football analysis has developed **models for the quantitative evaluation of playing actions** aimed at objectively measuring the contribution of each action to the evolution of a match's score. In other words, these approaches assign a numerical value to each event in a match, reflecting its impact on the chances of a team scoring or conceding a goal [34]. The main models of this type include **Expected Threat** (written xT), **VAEP** (*Valuing Actions by Estimating Probabilities*) and **On-Ball Value** (OBV). We present each of these models below, explaining their logic, advantages and limitations, before offering a brief summary comparison.

Expected Threat (xT)

The Expected Threat (xT) was originally proposed in 2018 by Karun Singh [34]. Its principle is to divide the pitch into zones and assign each one a value corresponding to the probability that a possession will result in a goal from that zone. This defines a"-valued surface on the pitch, where each location has an intrinsic xT estimated from historical data[34]. An action that moves the ball from one area to another that is more dangerous (e.g. a forward pass) is assigned a value equal to the increase in the probability of scoring induced, i.e. the difference in xT between the area where the ball arrives and the area where it leaves. Conversely, if the ball is moved to a less favourable area (e.g. a backward pass) or lost, the xT model assigns no positive value to this action. Thanks to this construction, xT makes it possible to quantify the offensive contribution of passes and ball movement in a simple and intuitive way, over and above decisive passes alone. In practice, xT has become one of the best-known action evaluation indicators in the world of football : 'What_is_Expected_Threat_xT_Possession_Value_models_explained'. However, the simplicity of xT comes with several limitations. Firstly, this model only values actions that move the ball from one area to another and ignores other types of action: neither shots nor defensive interventions (tackles, interceptions, etc.) are included in the calculation of xT [34].On the other hand, xT typically relies on relatively summary event data (positions and outcomes of passes, etc.) and *neglects certain contextual factors* such as the defensive pressure exerted on the ball carrier [34]. To sum up, xT provides an accessible measure of the offensive threat created by passes and transmissions, but it offers a partial view of the total contribution of players because it does not take into account the impact of defensive actions or the risk of losing the ball.

VAEP

The **VAEP** model (*Valuing Actions by Estimating Probabilities*), developed by Decroos *et al.* [8] and presented in 2019, extends the approach by evaluating *all types of actions* carried out by players, whether they are offensive (passes, dribbles, shots...) or defensive (tackles, interceptions, clearances...), by considering their effect on both the chances of scoring and the chances of conceding a goal. For each action performed, VAEP seeks to estimate two quantities: (1) how much this action increases or decreases the probability that the player's team will score a goal in the near future, and (2) how much it increases or decreases the probability that his team will concede a goal in the same time frame [8]. The methodology consists of training a supervised classification model on a large dataset of match events, in order to predict these future goal probabilities (for each team) from the context of the action (type of action, location, minute, score, etc.) [8]. The VAEP value of an action is then defined as the sum of its *offensive value* (the increase in the probability of scoring thanks to the action) and its *defensive value* (the reduction in the probability of conceding thanks to the action) [8]. Formally, if an action a_i changes the state of the game from a state S_{i-1} to a new state S_i, we can write $V(a_i) = [P(\text{score}|S_i)P(\text{score}|S_{i1})] - [P(\text{concede}|S_i)P(\text{concede}|S_{i1})]]$ for the team of the player performing the action (this formula corresponds to the definition proposed by Decroos *et al.* [8]). Intuitively, an action that is beneficial for the team (for example, a pass that creates a goal-scoring opportunity) will see $P(\text{goal})$ increase and therefore $V(a_i)$ positive, whereas an action that is dangerous for the team (for example, a missed pass that leads to an opposing counter-attack) will see $P(\text{collect})$ increase and therefore $V(a_i)$ negative. We thus obtain an objective measure of the contribution of each action to the potential evolution of the score [8]. Compared to xT, VAEP has the major advantage of covering all actions and situations in the game. For example, it explicitly credits successful defensive interventions (which reduce the probability of conceding) and can penalise high-risk actions that result in a dangerous loss of the ball [34]. VAEP therefore offers a more complete view of a player's contribution, taking into account both the defensive and offensive sides of his impact on the match. By aggregating the VAEP values of all a player's actions, we can quantify his total contribution on both sides of the pitch [8]. However, the VAEP model also has certain limitations. On the one hand, it is based on traditional event data (from suppliers such as Wyscout/Opta) which, at the time of the study, did not contain detailed information such as the presence of defensive pressure on the action or the exact positions of all the players. As a result, the context taken into account by VAEP remains incomplete, being limited to the few variables available (type of action, coordinates, minute, score, etc.) and to a *history of previous actions* to enrich the immediate context [8]. In particular, the authors of VAEP included as features the unfolding of the three actions preceding the action under consideration, in order to partially capture the dynamics of possession [8]. However, subsequent analyses showed that the inclusion of these memory "variables could introduce a systematic bias in the evaluation of players [34]. This is because dominant teams tend to have longer sequences of possession; thus, a player from a very strong team will more often benefit from a favourable possession context (several successive passes from his team-mates before his action) than a player from a weak team. The VAEP model, by learning from these sequences, then runs the risk of over-valuing certain actions simply because they are part of the well-constructed possession of a strong team, rather

than because of their intrinsic merits. This collective strength bias was highlighted in VAEP [34]. Finally, like all models based purely on event data, VAEP cannot directly value contributions *hors-ball* (placements, pressings without interceptions, ball calls not followed by passes, etc.), which leaves part of a player's qualitative performance outside its numerical evaluation.

On-Ball Value (OBV)

The **On-Ball Value (OBV)** is a model introduced by the Hudl/StatsBomb company in 2021 [33], which follows on from VAEP while making improvements designed to correct its identified limitations. Like VAEP, OBV evaluates each action in terms of its impact on the chances of scoring (*Goals For*) and conceding (*Goals Against*) a goal for the team in question [33]. However, instead of training the model directly on goals scored or conceded (rare events leading to high variance), OBV is trained on the basis of the *Expected Goals (xG)* accumulated over the course of possessions [33]. In other words, we use the sum of the goal probabilities (xG) generated during a possession as a continuous estimate of its offensive value, which serves as a signal for learning the contribution of intermediate actions. This choice makes it possible to *stabilise learning* by having many more positive examples (each shot brings an xG value, instead of only considering goals) and to attenuate the random effect of rare goals [33]. Secondly, OBV distinguishes itself by learning *two separate models*: one to assess the impact of the action on the chances of scoring (offensive value), and another to estimate its impact on the chances of conceding (defensive value) [33]. Unlike VAEP, which produces a single value combining these two aspects, this dual-component design allows OBV to better discriminate between the offensive and defensive contributions of a given action. For example, a successful defensive intervention in the box will greatly increase the OBV defensive score of the action (by reducing the immediate danger to the opponent) without affecting the offensive score, whereas a forward pass will have a mainly positive offensive value and a neutral defensive component. A player's final evaluation can thus distinguish his offensive contribution from his defensive contribution by summing up his offensive and defensive OBVs separately over the season. In addition, OBV has made modelling choices designed to eliminate the team context bias observed with VAEP. In particular, OBV **does not include variables describing possession history** (sequence of previous actions) in its features, precisely to avoid indirectly favouring actions performed by players from dominant teams [33]. Each action is evaluated intrinsically on the basis of its own characteristics and the state of the immediate game, but without explicit consideration of the length of possession or the number of passes that preceded it. This approach aims to ensure that the value assigned comes from the action itself and its situational context, and not from the overall level of the team. In addition, OBV exploits the enriched data provided by StatsBomb: for each action, the model uses detailed features such as the exact position on the pitch (x, y coordinates, distance and angle in relation to the goal), the type of action and its context (open play, set-piece, etc.), the defensive pressure experienced during the action (indicated in the StatsBomb dataset) or the body part used by the player [33]. The inclusion of this granular information means that the conditions under which the action is carried out can be better taken into account (for example, a pass made under heavy pressure from the opposition does not have the same difficulty or the same meaning as a pass that is unmarked, which the model can incorporate via the features). Finally, a notable

difference in OBV concerns the treatment of passes: the model *does not explicitly credit the receiver of a pass* for the receiving action itself [33]. Indeed, from the point of view of event data, receiving a pass does not in itself create progress towards the goal (it is the subsequent action that will potentially create value). OBV therefore considers that the successful reception of a pass adds no immediate value, and that its value resides entirely in the subsequent action enabled by this reception [33]. Thus, a player who positions himself intelligently to receive the ball will be rewarded in OBV not at the moment of reception, but through the success of the action he carries out immediately afterwards (for example a progressive pass, a shot, etc.), or penalised if he loses the ball immediately afterwards. This methodological choice highlights the difficulty that purely event-based models have in quantifying non-balloon contributions. Travelling without a balloon to make oneself available, for example, is not directly measured, and its effect is only captured by OBV indirectly via the success of a follow-up action. In summary, OBV represents the state of the art in action evaluation models: it *covers all actions* like VAEP, by quantifying both the offensive and defensive contribution of each gesture, but it provides *better robustness* thanks to the use of the xG as a learning target and the separation of the offensive/defensive components, and it *corrects to a large extent the bias* against weak teams by not using long possession descriptors [33]. In addition, the use of more detailed data gives it greater sensitivity to the immediate context than a model such as VAEP. Its limitations remain those inherent in event data: like VAEP, OBV can only explicitly evaluate the actions recorded on the ball, and not all tactical work without the ball. However, it is one of the most advanced models publicly available (via StatsBomb) for estimating the value of each football action.

Comparison of approaches

xThreat, VAEP and OBV all belong to the family of action valuation models, but with different scopes, methods and objectives. **xThreat** is distinguished by its simplicity: it provides an intuitive measure of the danger created by ball transmissions, as a function of progress towards favourable zones. It is lightweight and easy to interpret, but it ignores the majority of other types of action (shots, defensive interventions, ball losses) and neglects an important part of the context.

VAEP, proposed in an open academic framework, has introduced a more comprehensive view of valuation, by modelling the impact of each action - offensive or defensive - on the chances of scoring or conceding a goal. Its probabilistic framework provides a general basis for evaluating players' overall contributions. However, VAEP can suffer from a certain sensitivity to context bias (particularly in the modelling of possession dynamics) and depends on the limitations inherent in standard event data.

OBV, designed more recently by StatsBomb, takes up several of VAEP's ideas and seeks to correct its limitations, notably by using Expected Goals as a learning signal and by explicitly separating offensive and defensive components. OBV also takes advantage of enriched data (pressure, body used, etc.) to improve contextual sensitivity. This makes it perform well on StatsBomb data, but its proprietary nature and the lack of detailed publication of its methodology limit its transparency and reproducibility in an open research framework.

In summary, each approach has specific strengths and limitations. VAEP provides a robust

and repeatable foundation for customised pipelines, while OBV represents a practical optimisation reserved for proprietary environments with enriched and controlled data.

Methodological choices for our approach

In our work, we have chosen to use the VAEP model [8] as the basis for valuing shares.

This choice is motivated by several considerations. Firstly, although OBV [33] represents an interesting development in a specific industrial context, access to it is limited: the exact methodology is not entirely public, the model relies on enriched proprietary data (measured pressures, precise context), and its concrete implementation remains a black box for open research. Secondly, thanks to our tracking and calibration pipeline (see section 3.8.2), we have complete positional information on the players and the ball. This allows us to enrich the VAEP framework by injecting reconstructed tactical contextual elements (e.g. local pressure, distance to goal, defensive density), while retaining control over the entire value chain.

Finally, VAEP is structurally flexible enough for our experimental environment: its probabilistic modelling allows us to adjust the time horizon, explore different spatial granularities and fine-tune the balance between offensive and defensive contributions, depending on the objectives of our study.

The details of the adaptation and extension of the VAEP model in our pipeline are presented in section 3.9.

3 Method

3.1 Presentation of the general approach

The entire methodology developed in this work is based on a modular architecture built around two distinct but complementary pipelines: a player tracking pipeline and a ball action spotting pipeline. These two pipelines share the same input source - raw video extracts (30-second clips) from football matches - and have been designed so that they can be run independently and potentially in parallel, allowing flexibility of use and adaptation to different analysis scenarios.

Common input: video frames At the heart of our system are video clips extracted from matches, which are broken down frame by frame (frame extraction). These frames are the only entry point for the entire pipeline and are shared by the two main modules: tracking and spotting. This unified approach ensures temporal consistency between position and event information.

Pipeline 1 — Player tracking The first component of the pipeline aims to reconstruct the trajectories of players on the field from raw frames. This involves several sub-steps: player detection, multi-frame re-identification, temporal association, and camera calibration to project positions onto the field in metric coordinates. The final result of this pipeline is a structured set of time-synchronised player positions, represented in tabular form.

Pipeline 2 — Ball Action Spotting In parallel, the second pipeline uses the same video clips to detect game actions related to the ball. This task relies on a pre-trained spotting model which,

from a set of consecutive frames, produces a temporal sequence of probable actions (passes, shots, controls, etc.). The model returns ranked predictions for each time window, accompanied by a confidence score and a corresponding timestamp.

Merging and post-processing of results Once the two pipelines have been executed, their respective outputs are post-processed for merging. The detected actions are aligned temporally with the players' trajectories, allowing each event to be contextualised in its spatial environment. This process results in the creation of a consolidated tabular file, in which each row represents an action, enriched with information on the positions of the players at the time of the action.

Classification and valuation of action This consolidated tabular file is then used as input for a classifier whose task is to assign a value to each share. This classification model is based on a set of tactical variables (position, type of action, defensive configuration, etc.) to estimate the potential impact of each event on the probabilities of scoring or conceding a goal. The final result of the pipeline is a sequence of valued actions that can be interpreted and used for tactical analysis or individual performance measurement.

Conclusion on the structure This organisation into separate pipelines, combined with a centralised merging logic, maintains a clear, scalable and easily adaptable architecture. Each component can be improved or replaced independently, while ensuring full compatibility at the final output level.

3.2 Environment

Before presenting the various modules of our pipeline, it is essential to clarify the context in which the experiments were conducted. Unlike much research work based on high-performance servers equipped with professional GPUs (such as NVIDIA A100, V100 or RTX 6000), all of our experiments were conducted locally on a personal computer.

More specifically, inferences and processing were performed on a MacBook equipped with an Apple M4 chip, with 16 CPU cores, 34 GPU cores, and 32 GB of unified memory. This choice of infrastructure has several practical advantages (autonomy, reproducibility, control of the environment), but also implies hardware and software constraints that influence certain methodological choices.

3.2.1 Compatibility with Metal and absence of CUDA

The main impact is the lack of native compatibility with CUDA libraries, which are at the heart of the NVIDIA ecosystem used in most deep learning frameworks. As a result, some classic features such as 'torch.cuda.amp.autocast', used for training or inference in mixed precision (FP16), are not yet fully supported under Metal Performance Shaders (MPS), the backend used on macOS.

This has two main consequences:

- **Disabling AMP optimisations**: calls to 'autocast' had to be removed or replaced with neutral contexts, which may increase inference times and slightly increase memory footprint.

- **Incompatibility of certain dependencies**: certain external libraries pre-compiled for CUDA cannot be used directly. CPU versions or generic alternatives have been preferred.

3.2.2 Limitations related to extensions compiled in C/C++/CUDA

Many libraries in the field of computer vision or sequential processing (MMDetection, MMOCR, MMCV, etc.) rely on modules compiled in C++ and/or CUDA for performance reasons. However, these modules are generally compiled for a CUDA-compatible x86 architecture and are not immediately compatible with the ARM64 architectures of Apple chips or with the Metal backend.

In practice, this means that:

- some native extensions fail to compile locally (undefined symbol or unsupported architecture errors),
- incompatibilities arise with PyTorch's dynamic loaders (especially for `.so` or `.dylib` files),
- it is sometimes necessary to rewrite or disable certain components in order for these libraries to work on macOS.

This problem has been encountered in particular with the MMOCR library, some of whose features (feature extraction, sequential alignment, custom encoding modules) require specific CUDA compilations that cannot be used on macOS/MPS. Adjustments were therefore necessary to work around these dependencies.

3.2.3 Impact on measured performance

Any direct comparison with results from the literature should therefore be viewed with caution. Locally measured inference times (for the T-DEED model or other spotting modules) are not directly comparable to benchmarks performed on optimised clusters. Similarly, some optimisations potentially available on CUDA GPUs have not been reproduced here.

That said, the choice of a consistent and controlled local environment is fully justified as it guarantees complete reproducibility of the results, while illustrating the pipeline's ability to run on consumer hardware, which may be of practical interest in real-world application contexts.

This execution context therefore influences several technical decisions made in the following sections, whether in terms of memory management, data format, or the choice of base models used.

3.3 Dataset

All of the experiments described in this thesis are based on two main data sources, which are distinct in nature and structure: a video dataset 3.3.1 and a tabular dataset 3.3.2. These datasets were used for complementary purposes in the various stages of pipeline design, inference, and evaluation.

3.3.1 Video clips from SoccerNet

The core of the pipeline is based on video clips from the **SoccerNet** benchmark (SoccerNet-v2 version). The data used here consists exclusively of *raw 30-second excerpts*, taken directly from the videos in the dataset, without any additional transformation or segmentation. No re-encoding, manual cutting or event extraction was performed beforehand.

General characteristics

- **Duration per clip**: exactly 30 seconds, without overlap,
- **Format**: MP4 video files, converted to individual images (frames) at 25 FPS,
- **Single camera**: each clip is captured from the main broadcast camera,
- **Content**: the clips cover different phases of play indiscriminately, with no guarantee that an action will be present.

Organisation and structure Each frame is associated with a unique identifier encoded in the file name, according to the following convention:

`SCCCXXXFFF`

where:

- `S`: set identifier (1 = train, 2 = valid, 3 = test, 4 = challenge),
- `CCC`: clip identifier (3 digits),
- `FFF`: frame index (from 001 to 750 for 30s at 25 FPS).

This nomenclature allows for easy reconstruction of clips in the pre- or post-processing stages, enabling local operations to be performed on individual clips in certain situations in order to improve the temporal consistency of predictions, for example.

Specificity of the format used Unlike standard SoccerNet configurations that use multiple cameras and rich annotations throughout the match, our single-camera configuration with short clips results in a significant loss of context. This directly impacts the performance of spotting models trained on long sequences, particularly for contextual actions (fouls, passes, etc.) whose beginning or end may be truncated.

3.3.2 Tabular data: StatsBomb 360 Frames

To enable the actions detected in the raw video clips to be evaluated, a second dataset is used to train a model for predicting the score of an action. The central idea is to be able to estimate the tactical value of an action solely from a video sequence, which requires prior supervised learning on annotated examples.

The **StatsBomb 360** dataset, published as open data, provides this type of information: each match event (pass, shot, duel, etc.) is accompanied by a partial capture of the spatial state of the players on the field at time t.

General data structure Each entry in the dataset corresponds to an event annotated according to the JSON format developed by StatsBomb:

- `event_uuid`: unique identifier of the event linked to the capture,
- `visible_area`: polygon describing the area visible on the screen (broadcast camera),
- `freeze_frame`: list of visible players with their coordinates $[x, y]$ on the field, accompanied by Boolean indicators (`teammate`, `actor`, `keeper`).

Use in our pipeline This tabular data is used to train a supervised action valuation model. This model learns to associate the spatial configuration of a situation with a target tactical value (e.g., probability of scoring in the next few seconds). Once trained, it can be applied to situations automatically detected from raw clips: the nature of the action (e.g., pass or shot) and the position of the players are then predicted by the other video modules in the pipeline.

The StatsBomb dataset is therefore only used during the pre-training and evaluation phases of the valuation model, but not in inference.

Known limitations The 360 data has some specific characteristics that need to be taken into account:

- Not all frames include all 22 visible players.
- The visible field is not always complete (masked or blurred areas),
- Some captures are missing a designated player (no information on the ball carrier),
- The camera angle may vary between matches, which affects the readability of tactical configurations.

These two types of data form the input base for our pipeline: video clips as the main inference medium, and tabular data as the learning base for the valuation model. The next section will provide a detailed exploratory analysis of this data, both quantitatively and structurally.

3.3.3 Pre-processing and normalisation of StatsBomb data

Before any modelling or valuation, it is crucial to ensure the homogeneity and structural consistency of the data. However, StatsBomb 360° data uses a coordinate format (x, y) expressed on a field with dimensions 120×80, which differs from the conventions adopted by other providers or standard analytical tools such as `socceraction`. Figure 1 illustrates this diversity of spatial coordinates according to the main data providers.

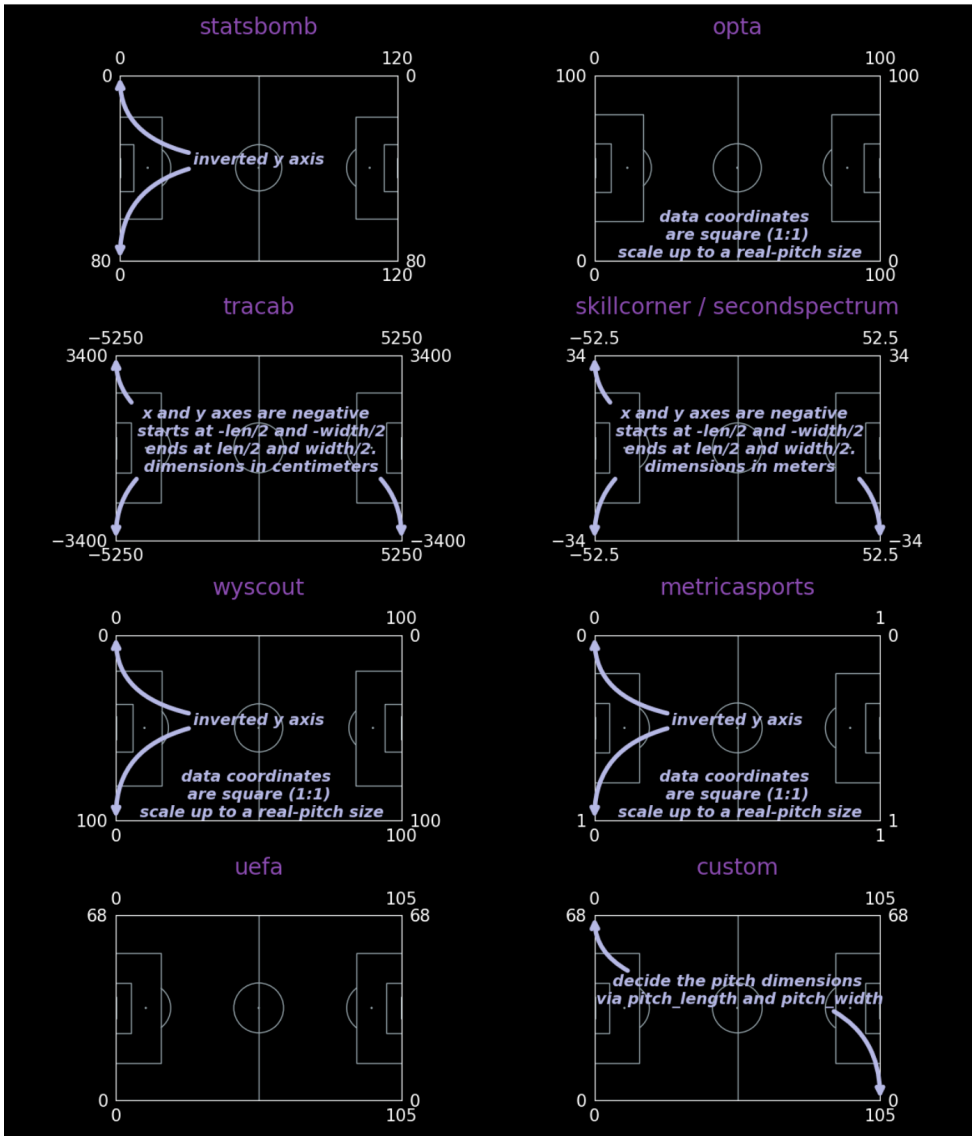


Figure 1: Comparison of spatial coordinates used by different data providers. Source: mplsoccer documentation

To ensure compatibility with our analysis tools and the video modules of the pipeline, all coordinates have been converted to a common 105×68 reference frame, corresponding to the SPADL (*Soccer Player Action Description Language*) standard. This transformation has been applied uniformly to the coordinates of action points (`start_x`, `end_x`), player positions (`freeze_frame`) and visible field polygons (`visible_area`).

Spatial standardisation ensures consistency in calculations, avoids confusion related to units or initial dimensions, and facilitates comparison between different types of entities (players, ball, visible surface). In addition, directional standardisation has been applied: all actions have been transformed to be represented in a single direction of play (from left to right) in order to maintain consistent reading of sequences, regardless of the status of the team (home or away).

Finally, each event is encoded in a structured tabular format via SPADL. This format facilitates downstream processing by describing each action using standardised columns: type, positions, player, team, result, timestamp. It also allows different data sources to be aligned

within a common, unified framework, ready for exploration and supervised learning stages.

This standardisation process is an essential prerequisite for ensuring the quality and robustness of the analyses. It provides a stable basis for exploratory analysis, spatial feature generation and model training, while limiting biases associated with heterogeneous raw formats.

3.4 EDA (Exploration Data Analysis)

3.4.1 Exploratory analysis of the SoccerNet dataset

The SoccerNet dataset is the main basis for our pipeline. For each frame, it includes image-by-image annotations of dynamic objects (players, ball, referees) as well as visible lines on the pitch. This corpus serves as a common foundation for all the modules developed in this work: event detection, multi-player tracking, re-identification, role or number recognition, camera calibration, etc.

The objective of this section is to examine the structure, completeness and limitations of these annotations in order to identify the frames that can actually be used and to anticipate the technical constraints that arise from them. The analysis focuses on both the presence and quality of dynamic objects and the coverage of the lines of the pitch visible in the image.

This study will serve as a basis for defining the filtering, training and evaluation strategies implemented in the subsequent stages of the pipeline.

Annotation structure Each 30-second clip (750 frames at 25 FPS) is accompanied by a `Labels-GameState.json` file containing frame-by-frame annotations. This file contains two main types of annotations, differentiated by the `supercategory` field:

- supercategory entities `object`, which describe the dynamic objects present in the image: players, referees, ball;
- supercategory entities `pitch`, which describe the lines visible on the pitch in each frame.

Each `object` supercategory annotation contains:

- a unique identifier `id` and a frame identifier `image_id` (in the form SCCCXXXFFF);
- a `track_id` local to the clip, allowing the object to be tracked over time;
- a category (`category_id`: 1 for player, 2 for referee,3 for goalkeeper, 4 for ball);
- attributes describing the role (`player`, `referee`, etc.), the team (`left`, `right`) and sometimes the jersey number (`jersey`) ;
- an enclosing box in the image $\texttt{bbox}_{image}$, $defined by the absolute coordinates \texttt{x}, \texttt{y}, the width \texttt{w} and the height \texttt{h}$;
- a version `bbox_pitch_raw` derived from the raw prediction before filtering.

Verification of annotation completeness First, we verified that all annotated clips contain 750 frames, in accordance with the expected format (30 seconds at 25 FPS). This check was performed directly from the `image_id` present in the annotation files, without going through the image files.

The result is consistent: all clips in the `train` and `valid` splits have exactly 750 unique frame IDs, with no duplicates or missing frames.

However, a more detailed analysis identified a recurring phenomenon: some frames, although present in the annotation files, do not contain any `object` type objects with valid `bbox_image` and `bbox_pitch` fields. These frames are therefore present in the JSON structure, but do not contain any content that can be used by modules requiring image-terrain matching, such as tracking or calibration.

This phenomenon remains marginal : out of the 86,250 frames analysed (115 clips $\times$ 750 frames), only 1,080 frames (1.25%) are affected. Their distribution is irregular and does not seem to be linked to a particular position in the clips (beginning, end, specific game phase). These cases will simply be ignored in subsequent steps without any significant impact on the overall coverage of the dataset.

To confirm the origin of these absences, several of these frames were inspected manually. Although the visual content clearly shows the presence of players in the image, no corresponding annotation is entered in the `Labels-GameState.json` file. This suggests that this is an annotation error rather than empty or unusable content. Two typical examples are shown in Figure 2.

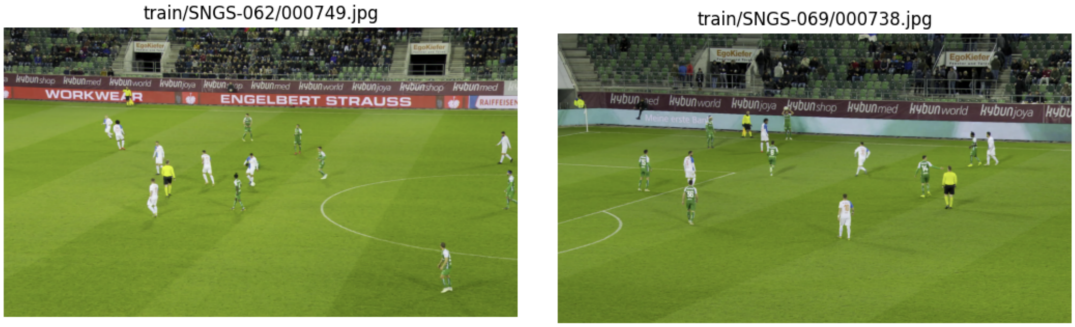


Figure 2: Examples of annotated frames with no objects detected despite the visible presence of several players in the image.

Analysis of visible lines An exploratory analysis of visible lines was conducted to quantify the geometric richness of the frames. As illustrated in Figure 3, the majority of annotated frames contain between 3 and 13 different types of lines, with a peak around 12 lines, which indicates good spatial coverage over a large portion of the field. Some poorer frames (fewer than 3 lines) are also present, particularly during camera zooms or off-centre phases of play.

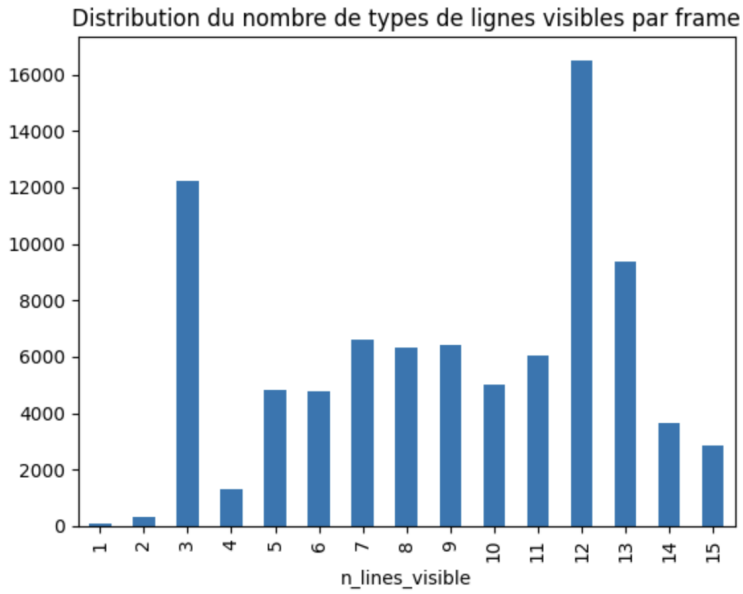


Figure 3: Distribution of the number of visible line types per frame in the `pitch` annotations.

Figure 4 shows the frequency of occurrence of different types of lines in the dataset. Touch lines (`Side line top`, `Side line bottom`) and the central circle (`Circle central`) are by far the most frequent, reflecting a dominant side view framing. Conversely, lines related to penalty areas or goalposts are much rarer. This distribution will have a direct impact on calibration or spatial reconstruction modules, which will make greater use of the central areas of the pitch, which are better covered by annotations.

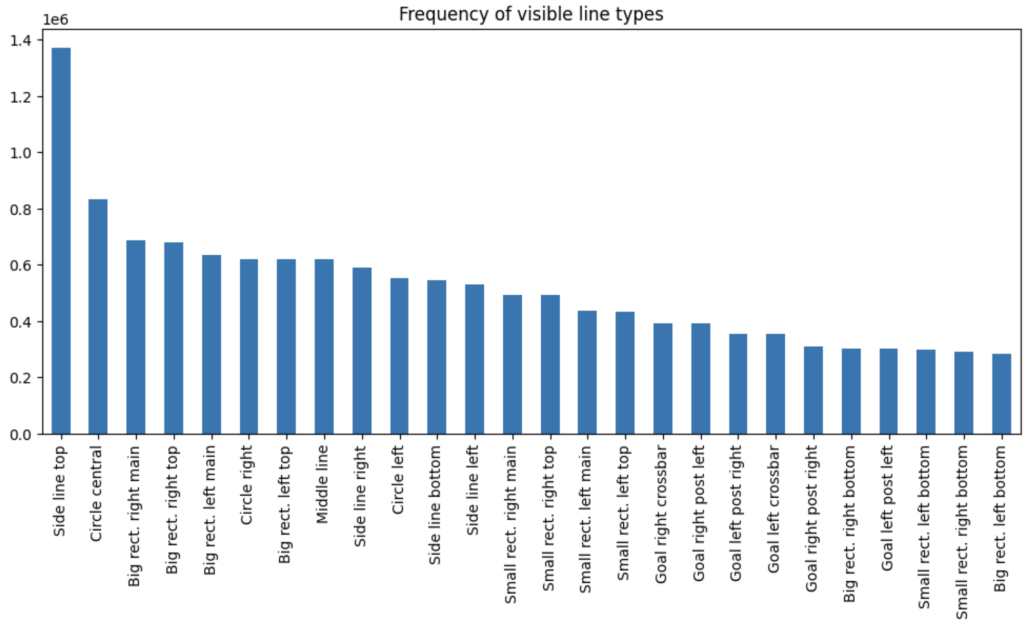


Figure 4: Frequency of occurrence of different types of lines in the entire dataset.

Size of dynamic objects Figure 5 shows the distribution of the heights of bounding boxes annotated for dynamic objects in the game, grouped by category. We observe that field players (`player`) and goalkeepers (`goalkeeper`) are relatively similar in size, although goalkeepers are

slightly larger, as they are often filmed closer during goal sequences.

In contrast, the ball has a very low average height, generally less than 20 pixels, with extreme values close to 5 pixels. This observation highlights the major challenge posed by ball detection in this type of content: its small size makes it particularly susceptible to detection errors, occlusion, and blurring. This constraint fully justifies the use of specific approaches—such as adaptive tiling—to improve detection performance on small objects in future stages of the pipeline.

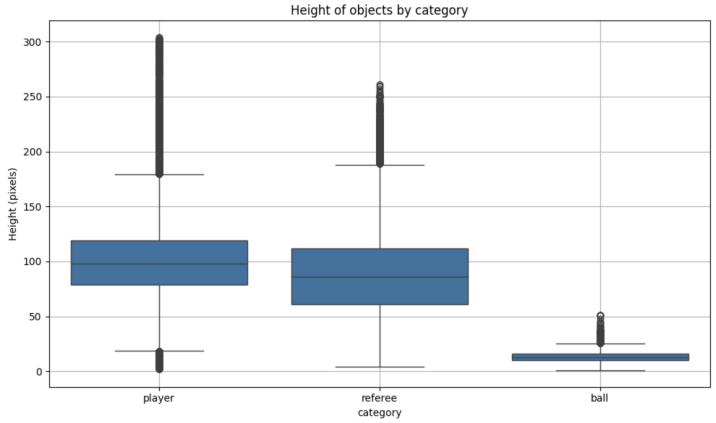


Figure 5: Distribution of bounding box heights by dynamic object category.

Visibility of jersey numbers per clip With a view to developing an OCR jersey recognition module, it is crucial to determine for each clip whether the players present are recognisable by their numbers. To do this, we measured, at the clip level, the percentage of players with at least one jersey number annotated in one of the 750 frames.

Figure 6 shows the distribution of this percentage across all 115 clips in the dataset. We can see that while the average is around 78%, a significant fraction of clips have more partial coverage, with less than 60% of players identified. Conversely, a majority of clips exceed 80%, with some reaching 95%.

This analysis reveals a crucial point for the development of an OCR module: the presence of a number in the `jersey` field does not guarantee that this number is visually readable on all frames where the player appears. Rather, it reflects the fact that the number was visible at least once during the clip, which can create a discrepancy between the annotation and the actual observable content.

This property has important implications for training: using these labels as ground truth at the frame level could introduce noise. It will therefore be necessary to adapt the granularity of the labels and the supervision strategies — for example, by aggregating several frames per player, or by filtering only clips deemed sufficiently informative.

3.4.2 StatsBomb Data

Before developing our stock valuation models, it is essential to thoroughly understand the structure, richness, and potential limitations of the dataset used. In this section, we present the characteristics of the StatsBomb dataset enriched with 360° data, as well as the initial exploratory analyses that will guide future methodological choices.

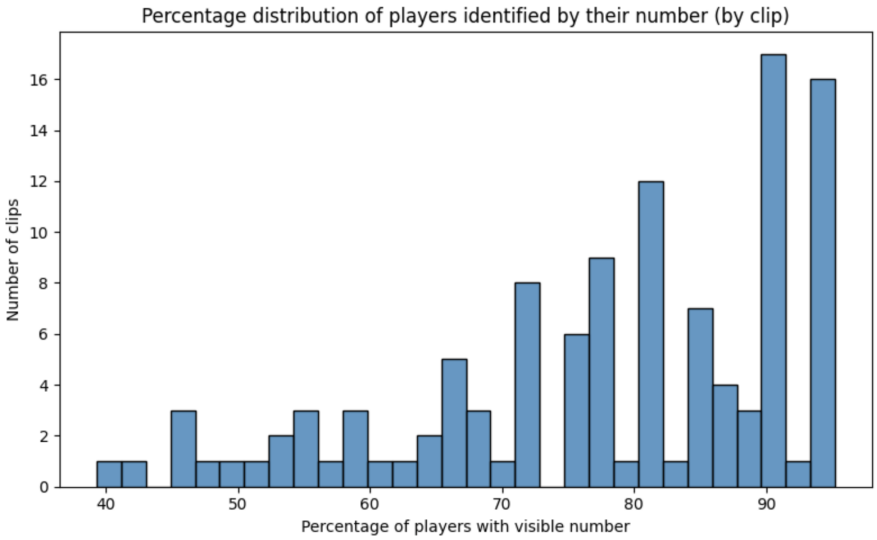


Figure 6: Distribution of the percentage of players identified by their jersey number, clip by clip.

Loading and structuring The dataset was imported and structured in `HDF5` format, with a set of actions encoded in SPADL format for each match. Each action contains, in particular: the start and end coordinates (x, y), the player ID, the type of action, the result, the part of the body used, and temporal meta-information.

All actions are enriched with player, team, and match information. The data was then merged with **StatsBomb 360** data, providing for certain actions the position of players present in the visible area at the moment of the action (via the `freeze_frame` and `visible_area` fields).

General summary of the dataset:

Characteristic	Value
Total number of shares	6,944,912
Number of matches	3,433
Number of players	8,925
Number of teams	310
Period covered	1958-06-24 to 2024-07-15
SPADL coordinates	$x \in [0, 105]$, $y \in [0, 68]$
Available columns	`game_id`, `team_id`, `player_id`, `start_x`, `start_y`, `type_name`, etc.

360° data coverage 360° data is only available for a limited subset of matches. Of the 6.9 million total actions, only **7.73%** have a `freeze_frame` or `visible_area` filled in. This coverage varies greatly depending on the competition.

Coverage rate by competition :

The coverage rates for the `freeze_frame` field vary greatly depending on the competition. Recent international competitions have excellent coverage, with, for example, **85.95%** for the UEFA Euro and **79.85%** for the UEFA Women's Euro. Others, such as the Women's World Cup (**48.09%**) or the FIFA World Cup (**39.17%**), offer partial but usable coverage. In contrast, most domestic leagues, such as the 1. Bundesliga (**10.07%**), have very limited coverage, and other competitions show a zero rate.

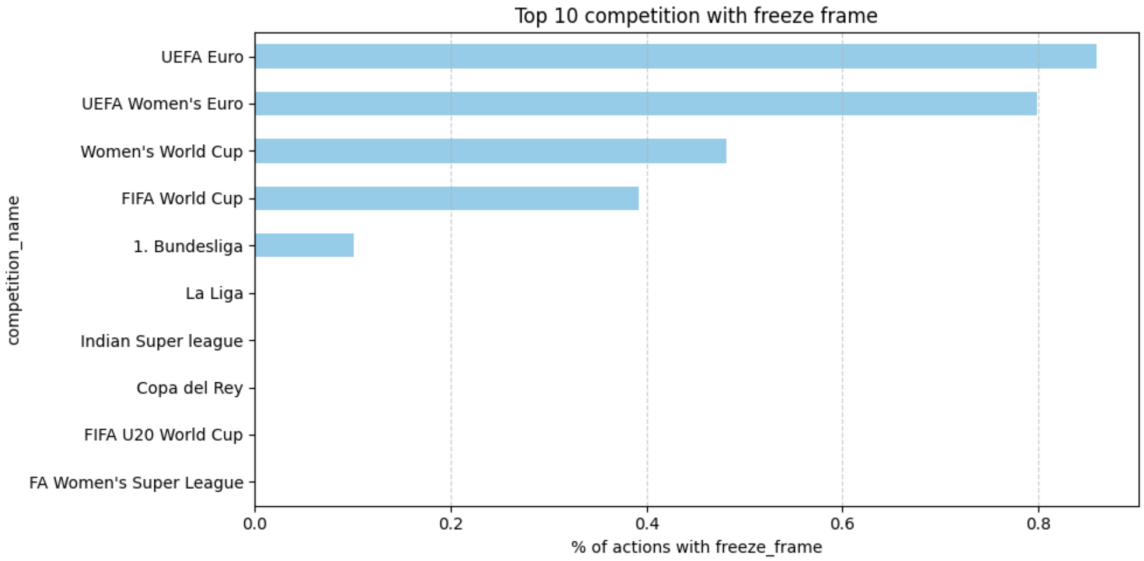


Figure 7: Distribution of freeze$_f rameratesbycompetition$

Selection of a 360° sub-dataset We selected matches with 360° coverage exceeding 80

Characteristic (sub-dataset)	**Value**
Number of actions retained	655,110
Number of matches	283
Number of goals detected	655
Number of unique players	2,253
Number of unique teams	103
Period covered	2021-06-11 to 2024-07-14
Average freeze$_f rame rate$	86.3%

Analysis of missing values Despite the good overall coverage rate in this subset, some actions still lack `freeze_frame`. Of the 655,110 actions, approximately **82,537** (12.6

We then checked whether these absences were isolated or located between two covered actions, which would allow for reasonable interpolation. Result: **35.87%** of the missing actions have a temporal context (previous *and* next actions covered).

This paves the way for interpolation or contextual imputation strategies if these frames prove necessary for further modelling.

Analysis of coverage by action type We analysed the distribution of the `freeze_frame` field according to action types. The graph below shows, for each type, the proportion of actions with a 360° annotation.

Some actions such as passes, shots, tackles or crosses have very high coverage (often >90

This disparity is important because we ultimately want to train a model on all actions. It is therefore essential to identify cases where player positioning is available or not. These results will guide the pre-processing strategy and the handling of missing values, particularly for modules using features derived from `freeze_frame` (e.g. defensive density, orientation, etc.).

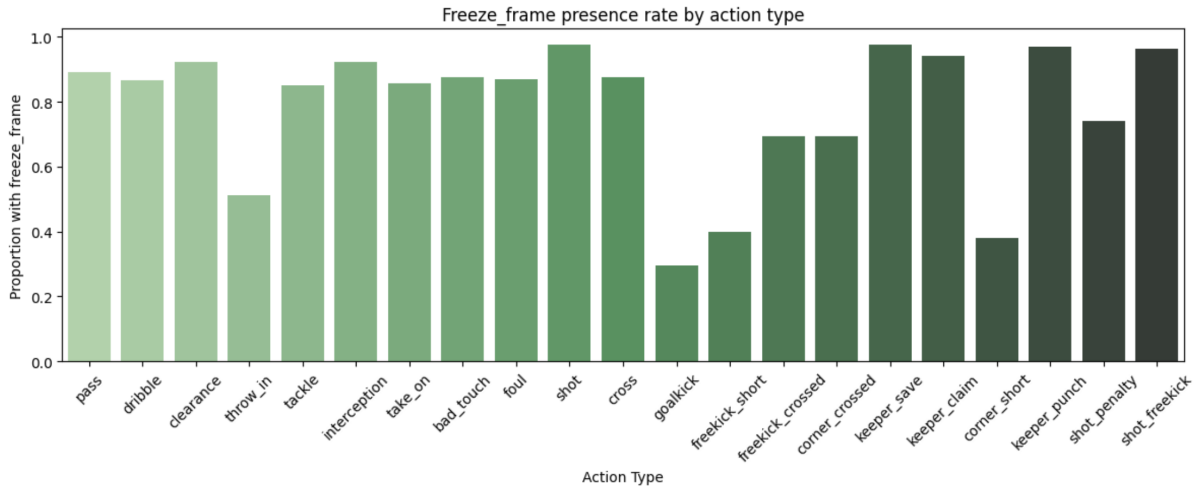


Figure 8: `freeze_frame` coverage rate by action type

Distribution of the visible area Each `freeze_frame` is associated with a visible area of the field, represented as a polygon via the `visible_area` field. To assess the overall quality of the observations, we calculated the area covered by these polygons (expressed in units of the field normalised to 105m × 68m).

As shown in Figure 9, illustrating the distribution of visible areas. There is a high concentration around values between 1000 and 3000, with an average of **2003** surface units. The maximum area observed is **6433.67**, which remains below the total area of the terrain ($105 \times 68 = 7140$). The minimum non-zero value is **130.30**, which corresponds to extremely limited visibility.

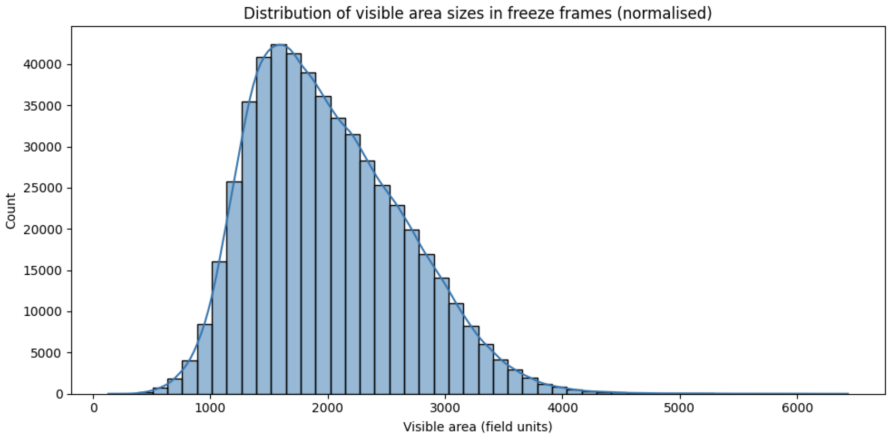


Figure 9: Distribution of the visible area ($\texttt{visible}_area)in freeze frames.$

This distribution allows us to anticipate the quality of the information available per action: actions associated with small visible areas are likely to provide incomplete context (particularly regarding teammates or opponents who are off-screen). It therefore offers an initial indirect measure of the contextual reliability of freeze frames, which we will then supplement with more qualitative analyses.

To complement this quantitative analysis, we now propose a more qualitative approach based on visual inspection of several freeze frames. The aim is to illustrate the disparities in contextual coverage by identifying different typical cases: an ideal situation, an average situation and a

problematic situation. These cases will serve as a reference to justify the need to introduce reliability assessment mechanisms in the modelling stages.

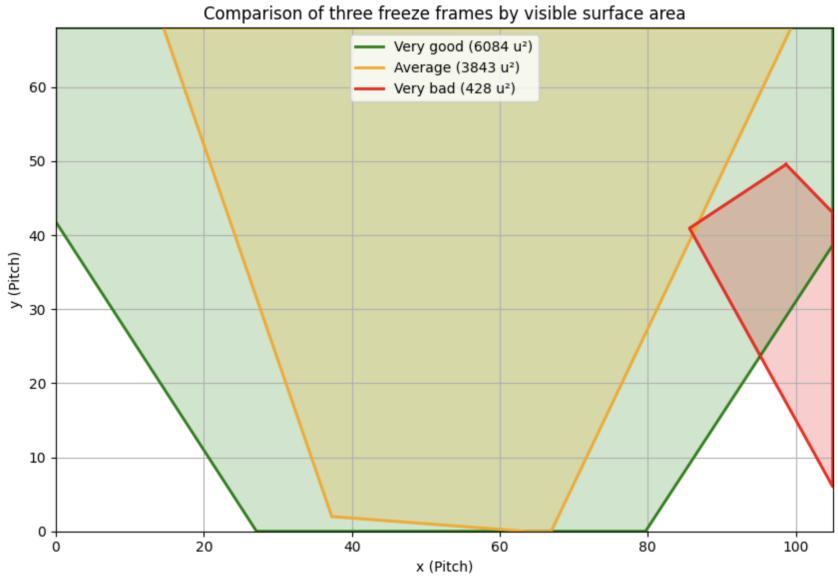


Figure 10: Comparison of three freeze frames according to their visible area.

This visualisation highlights the diversity of situations: some cases offer an almost complete view of half the pitch, while others only capture a small group of players in a corner of the image, often during set pieces (throw-ins, corners, etc.).

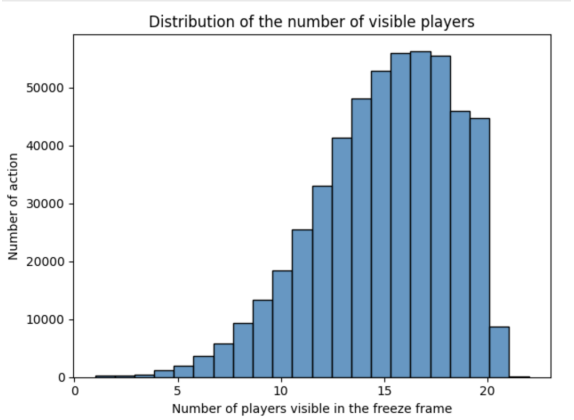
This observation highlights the natural and non-idealised nature of freeze frames: coverage varies depending on the context of the game, the camera or the position of the action, reflecting conditions similar to those encountered in raw videos.

However, by observing the orientation of the visibility polygons, it can be noted that some captures appear to come from cameras placed on the opposite side of the pitch, suggesting multi-camera acquisition. This is not a problem in itself, but it illustrates the diversity of shooting configurations in the dataset and the need to take this variability into account to ensure the robustness of the model from different points of view.

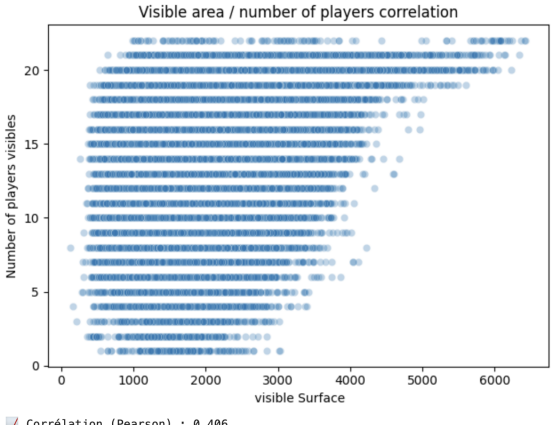
In order to refine our understanding of the context captured by freeze frames, we analysed the number of players visible in each situation. As shown in Figure 11, the majority of frames include between 13 and 17 players, with an average of 15.2 out of 22. This partial coverage is consistent with the intrinsic limitations of the broadcast camera, which generally cannot capture the entire field.

An additional zoom on the correlation between visible area and number of players present (Pearson correlation: $r = 0.406$) confirms an expected trend: the larger the visible area, the greater the probability of seeing a high number of players. This link, although moderate, justifies the interest in combining these two indicators to estimate the contextual richness of a situation.

Finally, it should be noted that the ball carrier is identified in approximately 86.4% of cases, an encouraging rate for future supervised tasks based on the location of the key player. On average, 7.6 opponents and 6.5 teammates (excluding the ball carrier) are visible per action.



(a) Distribution of the number of visible players



(b) Correlation between surface area and number of players

Figure 11: Analysis of the number of players visible per freeze frame.

Contribution of the StatsBomb dataset to spotting calibration Although StatsBomb data is not used directly to train or evaluate our spotting model — which is based on a separate corpus of annotated videos — it provides a structured framework for exploring certain temporal regularities in the game. These regularities guide the definition of relevant temporal constraints to be integrated into our post-processing logic (merging, filtering, consolidation).

Two visualisations from this dataset illustrate interesting temporal patterns:

- Some pairs of consecutive actions appear at the same second (`time_diff = 0`), suggesting that they are perceived as an instantaneous sequence in the course of the game. This is particularly the case for combinations such as `take_on` → `dribble` or `interception` → `pass`. This justifies exploring automatic grouping of events that are close in time during post-processing of predictions.

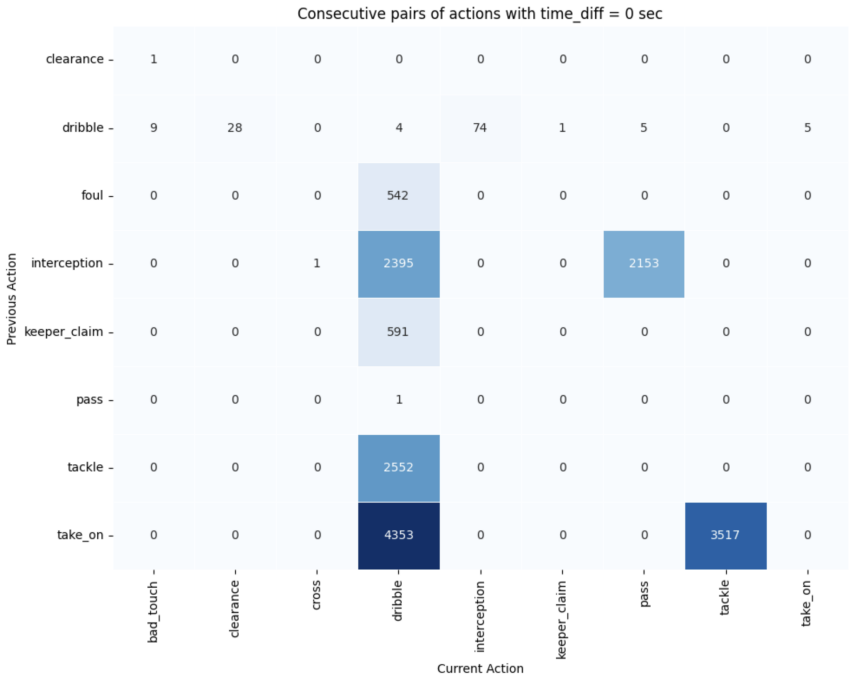


Figure 12: Pairs of consecutive actions occurring in the same second (`time_diff = 0`).

- Other transitions, such as those involving stoppages (penalties, free kicks, kick-offs), have much longer delays — sometimes exceeding 30 or 40 seconds. These observations will help set appropriate time thresholds when designing the spotting system, distinguishing between natural transitions (rapid sequences) and clear breaks in rhythm.

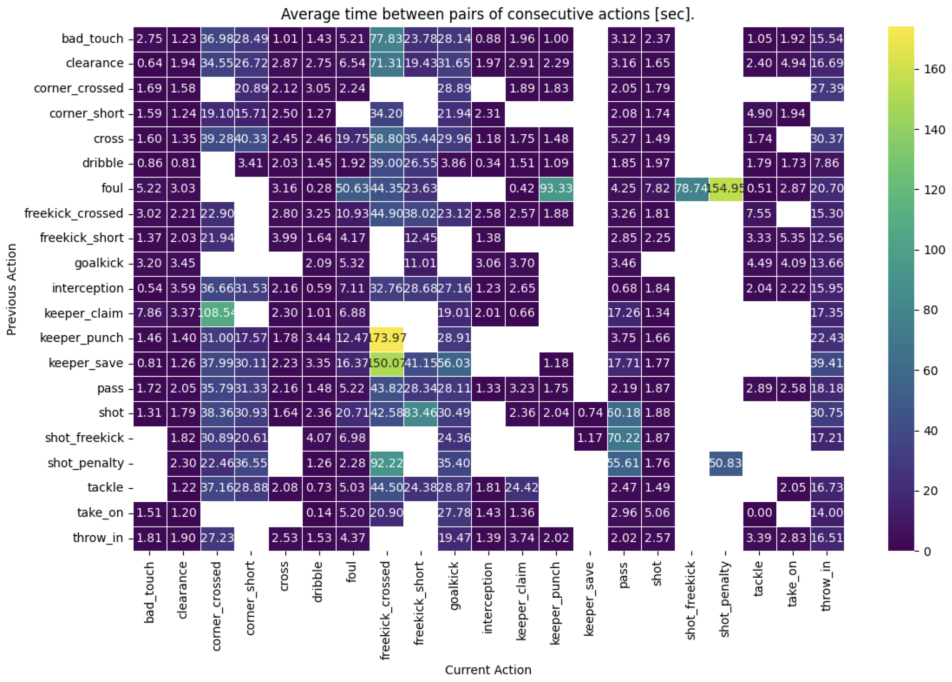


Figure 13: Average time (in seconds) between consecutive pairs of actions.

These analyses do not prejudge the exact dynamics of the video corpus, but they provide useful benchmarks for calibrating post-processing. They help to anchor our future hyperparameter choices in a measured football reality.

3.5 Evaluation Metrics

3.5.1 Introduction

In this section, we will present the various metrics that will be used in this report. By doing so, we will avoid redefining a metric each time it is used, and it will be considered known to the reader. Since the entire pipeline will perform several different tasks, each metric will be associated with a specific task. Common Metrics Let's start by presenting the most common metrics for machine learning evaluation. Precision Precision measures the proportion of positive predictions that are correct. From a more intuitive point of view, it gives us an indication of the model's propensity to make false detections (over-detection). It is defined by:

$$\text{Precision} = \frac{TP}{TP + FP}$$

where TP is the number of True Positives and FP is the number of False Positives.

Recall Recall (or sensitivity) measures the model's ability to detect all relevant objects. Intuitively, it tells us whether the model is missing important detections (under-detection). It is defined by:

$$\text{Recall} = \frac{TP}{TP + FN}$$

where FN represents False Negatives.

F1-Score The F1-score is the harmonic mean between precision and recall. It is useful for summarising the performance of a model in a single value when both aspects are important:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Brier Score The Brier Score measures the mean squared deviation between the probability predicted by the model and the actual class. Unlike accuracy or the F1 score, it takes into account the **quality of the probabilities** produced by the model, not just the classes. It is defined as:

$$\text{Brier Score} = \frac{1}{N} \sum_{i=1}^{N} (p_i - y_i)^2$$

where p_i is the predicted probability for observation i, and y_i is the actual class (0 or 1). A lower Brier Score indicates better performance.

ROC AUC The AUC (Area Under the Curve) corresponds to the area under the ROC curve, which plots the true positive rate (TPR) against the false positive rate (FPR) for different decision thresholds. This is a robust metric, especially in cases of **class imbalance**, and reflects the

model's ability to discriminate between classes. An AUC score of 1 represents a perfect classifier, while a score of 0.5 corresponds to a random classifier.

$$\text{TPR} = \frac{TP}{TP + FN}, \quad \text{FPR} = \frac{FP}{FP + TN}$$

AUC is particularly used in this project to evaluate the quality of VAEP-type probabilistic models.

3.5.2 Intersection over Union (IoU)

IoU (Intersection over Union) is a measure of spatial similarity between two bounding boxes: one predicted by the model and one annotated as ground truth. It is defined as the ratio between the area of the intersection and the area of the union of the two boxes:

$$IoU = \frac{A_{\text{intersection}}}{A_{\text{union}}}$$

This measure allows us to judge whether a prediction is correct. In most benchmarks (PASCAL VOC, COCO), a prediction is considered a *True Positive* if the IoU with a real box exceeds a threshold (e.g., 0.5) [11]. The IoU is therefore the basis for calculating precision, recall, and mAP in object detection tasks.

3.5.3 mean Average Precision (mAP)

mAP is one of the most widely used metrics for evaluating the quality of an object detection model [11]. It is the average of the average precisions (AP) calculated for each class. Each AP is obtained from the precision-recall curve, which is constructed by comparing the model's predictions with the ground truth, according to a given IoU threshold (usually 0.5, or a set of thresholds as in COCO [22]).The mAP is particularly useful because it provides an overview of the model's overall performance: it takes into account both the ability to avoid false positives (via precision) and to detect all relevant objects (via recall).**mAP formula:**AD

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \quad \text{where}_i = \int_0^1 p_i(r)\, dr$$

- N: number of classes
- AP_i: average precision for class i
- $p_i(r)$: accuracy function according to recall for class i

In this work, mAP will be used to evaluate the quality of player and ball detection, particularly in the bounding box detection module 3.8.3.

3.5.4 GS-HOTA

GS-HOTA (Generalised and Scalable Higher Order Tracking Accuracy) is a metric developed specifically to evaluate the task of Game State Reconstruction (GSR), i.e. the ability to correctly

locate, track and identify all players visible on a sports field from a video.

Unlike the standard HOTA metric, which relies on IoU similarity between bounding boxes, GS-HOTA relies on a new similarity score defined directly on the projected coordinates of the players and incorporating associated attributes (role, team, jersey number).

The similarity score between a prediction P and a ground truth G is defined as:

$$\text{Sim}_{\text{GS-HOTA}}(P, G) = \text{LocSim}(P, G) \times \text{IdSim}(P, G)$$

where:

$$\text{LocSim}(P, G) = \exp\left(\frac{\ln(0.05) \cdot \|P - G\|_2^2}{\tau^2}\right) \quad \text{and} \quad \text{IdSim}(P, G) = \begin{cases} 1 & \text{if all attributes are correct} \\ 0 & \text{otherwise} \end{cases}$$

The term $\text{LocSim}(P, G)$ measures the spatial location similarity between the prediction and the ground truth based on the Euclidean distance on the ground. This distance is transformed using a truncated Gaussian distribution with a tolerance of $\tau = 5$ metres, so that $\text{LocSim} \in [0, 1]$.

The term $\text{IdSim}(P, G)$ imposes a strict constraint on identification: all attributes must be correct (team, role, number) for a match to be validated. If a player is not visible, the GSR model must predict a neutral identity (e.g. role="player", team="null", jersey="null").

Once the associations between detections and ground truths have been made, the classic components of HOTA — DetA (detection accuracy) and AssA (association accuracy) — are then calculated and combined to give the final GS-HOTA score.

This metric therefore reflects not only the quality of localisation, but also that of multi-attribute identification, making it a comprehensive and rigorous metric for evaluating game state reconstruction from a single-camera video stream.

3.6 Interpretation and Model Explainability

3.6.1 Model interpretation and explainability

In addition to traditional quantitative evaluation using metrics such as AUC or Brier Score, it is essential to understand why a model makes certain decisions. This explainability step allows us to interpret the underlying mechanisms and better understand the impact of different variables on the predictions produced. To do this, we use the SHAP (SHapley Additive exPlanations) approach [23], a method now considered a benchmark for interpretability because it is based on rigorous principles from cooperative game theory.

Limitations of traditional approaches Traditional methods for variable importance in tree models — such as `feature_importances_` — have several well-identified limitations:

- They favour continuous or high-cardinality variables,
- They only provide *global* importance, without the possibility of local analysis.
- They do not capture interaction effects between variables.

- They can be inconsistent (a variable may be considered less important even if its actual influence increases).

General principle of SHAP The SHAP approach offers a mathematically sound alternative. Starting from a model f and an input $x \in \mathbb{R}^d$, SHAP decomposes the prediction $f(x)$ as a sum of the individual contributions of each variable:

$$f(x) = \phi_0 + \sum_{i=1}^{d} \phi_i \tag{1}$$

where ϕ_0 is the average prediction of the model (i.e., base value) and ϕ_i is the marginal contribution of variable i for this specific input.

The Shapley value ϕ_i is expressed as:

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \left[f_{S \cup \{i\}}(x) - f_S(x) \right] \tag{2}$$

where S is a subset of the variables (excluding i) and $f_S(x)$ is the prediction when only the variables in S are known. Although this exact formula is intractable for large dimensions, efficient implementations exist for decision trees via the TreeSHAP algorithm.

Overall importance of variables An initial use of SHAP values is to aggregate the mean absolute value of each variable across the entire dataset. This provides a reliable and unbiased *overall* importance. Unlike traditional methods, SHAP allows you to differentiate between variables that increase the prediction and those that decrease it.

Local explanation of specific actions The main advantage of SHAP lies in its ability to explain *each prediction* individually. It then becomes possible to analyse why a given prediction was produced by visualising the variables that influenced it and in what direction (positively or negatively).

This granularity is particularly valuable in an applied context, where we seek to understand the local behaviour of the model, validate its reasoning, or detect inconsistencies. As such, SHAP plays a central role in the validation, interpretation and practical use of tree-based models.

Positioning in this thesis SHAP analysis will be used systematically for all models based on decision trees (in particular Random Forests or XGBoost), in order to ensure fine-grained interpretability of both global and local predictions. This approach is fully integrated into a process of transparency and auditability of the models developed throughout the thesis.

3.7 Ball Action Spotting

The accurate detection of actions involving the ball, such as passes, shots or clearances, is an essential component of automatic football match analysis. This task, known as Precise Event Spotting (PES), aims to locate events with high informational value at a fine temporal resolution, often at the frame level. PES thus differs from more global approaches, such as activity

classification on a clip, where only the presence or absence of an action matters. Here, it is the precise moment of the action, in a sequence that is sometimes brief and ambiguous, that constitutes the signal sought.

3.7.1 Motivations and difficulties specific to spotting

Detecting ball actions presents a set of unique challenges:

- **Short duration of events**: actions such as a pass or a shot rarely extend beyond a few frames. It is therefore crucial to capture the weak signals that indicate their occurrence.
- **High similarity between contexts**: a calm phase preceding a pass may visually resemble a delay phase. It is therefore necessary to identify signals of intent, which are often subtle.
- **Temporal noise in the data**: even in manually annotated data sets, the precise moment of an action can be subject to interpretation. Annotations are therefore not entirely reliable. These difficulties justify the use of architectures capable of capturing long dynamics while maintaining high temporal resolution.

3.7.2 The T-DEED approach: philosophy and general structure

The T-DEED (*Temporal-Discriminability Enhancer Encoder-Decoder*)[39] model was designed to maximise temporal discriminability while respecting two fundamental constraints:

- maintain temporal resolution between input and output,
- allow the model to reason about long contexts to capture action start/end times.

To achieve this, the architecture is based on a skip-connection encoder-decoder structure, inspired by U-Nets[29] but adapted to the temporal domain. The idea is to progressively encode a sequence of T frames into representations with multiple granularities, then reconstruct them at constant resolution by injecting the associated local information at each decoding.

3.7.3 Encoder: Scalable Granularity Perception (SGP)

The encoder consists of a sequence of SGP blocks, each operating on the input temporal sequence. Each block extracts an extended context using dilated pooling, while maintaining a reasonable computational cost. This operation allows the model to consider large time windows without loss of resolution. A temporal convolution is then applied to detect local variations in the signal. This transformation acts as a filter capturing the short patterns typical of action transitions. Local attention is then calculated in a time band centred on each frame. This allows the contributions of neighbouring frames to be weighted according to their relevance for action detection. Finally, the result is merged with the initial input via an adaptive mechanism, which allows the model to dynamically combine local and contextual information.

3.7.4 Decoder: multi-level reconstruction by SGP-Mixer

The decoder reconstructs the output at constant resolution by integrating the information encoded at different levels. Each level applies upsampling followed by concatenation with the corresponding encoder representations. The reconstruction blocks use a structure inspired by MLP-Mixers and Perceiver IO. Two successive transformations are applied: one operates on the temporal dimension to model local dependencies, the other on the channel dimension to combine information of different types. This double processing allows each frame to decide which layers are most useful to it.

3.7.5 Temporal discriminability principle

An important innovation of T-DEED is the introduction of entropy regularisation, which is added to the main loss of the Binary Cross Entropy type:

$$\mathcal{L}_{\text{spot}} = -\sum_{t=1}^{T}\sum_{c=1}^{C}\left[y_{t,c}\log\hat{y}_{t,c} + (1-y_{t,c})\log(1-\hat{y}_{t,c})\right] \tag{3}$$

$$\mathcal{L}_{\text{ent}} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{c=1}^{C}\hat{y}_{t,c}\log\hat{y}_{t,c} \tag{4}$$

This regularisation pushes the model to make more decisive choices over time, favouring activations localised on a small number of frames.

3.7.6 Model assumptions and differences with our use

The T-DEED model was trained on complete matches filmed with multiple cameras, with temporal continuity and a rich spatio-temporal context.In our case, the data present several differences:

- The videos are isolated 30-second clips,
- A single dynamic camera is used, with rapid movements and cuts,
- Actions may appear at the beginning or end of a clip, without complete context.

These differences can affect the reliability of predictions, particularly for actions that are highly context-dependent.

3.7.7 Data preparation for inference

The model expects input clips of fixed length $T = 100$ frames. To process a longer video, we apply a sliding window with overlap (typically 75%). Each frame therefore appears in several temporal contexts, which reinforces the robustness of the predictions.

3.7.8 Merging predictions into a global timeline

The outputs of the clips are aggregated by averaging the activations of each frame across all clips containing it. This redundancy smooths out variations and reinforces recurring activations, while reducing isolated false positives. Post-processing: Soft Non-Maximum Suppression (SNMS) Soft-NMS filtering is applied to extract discrete events from continuous activations. For each class, local maxima are identified, then neighbouring activations are gradually attenuated according to a Gaussian function centred on these maxima. This strategy allows for flexible selection without abruptly suppressing ambiguous activations.

3.7.9 Final encoding of detections

Each valid detection is encoded according to the standard SoccerNet format, with the following fields: action name, time position (frame), confidence score, half, and stylised timestamp.

3.7.10 Critical discussion and limitations

Despite its qualities, T-DEED has certain limitations:

- Context sensitivity: some actions are better recognised in a dynamic context than in a static context,
- Misalignment with our target distribution: the pre-trained model is not perfectly suited to our videos,
- Blurred action boundaries: some actions do not have a clear start, which makes their spotting uncertain.

Possible improvements include finer calibration of thresholds, adjustment of time filters, or partial fine-tuning of our data if the volume allows it.concludes the description of the ball action detection module, which is an essential building block of our pipeline. It allows us to obtain a semantic representation of the game dynamics from simple raw videos, which is essential for subsequent analysis modules.

3.8 Tracking Players

3.8.1 Introduction

As discussed in the Previous Work section 2.3, tracking algorithms in the context of sports videos are still in their infancy in computer vision, and their maturity can be greatly improved. Several algorithms exist, each with their own strengths and weaknesses, and all offering different trade-offs between accuracy and speed.

The overall tracking task is actually a sequence of sub-problems: identity tracking between frames, multiple object detection (MOC), player recognition, re-identification, jersey number reading, etc. Given the complexity of the whole, it is not reasonable to aim for real-time

performance, even with the best Nvidia GPUs. The tool is therefore intended for post-analysis purposes.

In my case, I work locally on my Mac's GPUs and use 25 fps videos, which makes real time even more of a pipe dream. Of course, certain isolated components such as YOLO inference can run in real time on high-performance machines (in fact, YOLOv11 offers performance that allows real time even on a Mac GPU). But the combination of all these steps makes this goal unattainable within the scope of this project.

Therefore, choices have been made in order to obtain a tracking module that can be used for inference at a reasonable cost. Unsurprisingly, the starting point for this work will be the baseline of the Soccernet Gamestate Reconstruction challenge [19, 26, 32]. This baseline offers an ideal pipeline as a starting point and has been designed to allow each of its different modules to be modified fairly easily, while also allowing previously inferred module states to be loaded in order to save considerable time when focusing on a single module.

Although very practical, this baseline can, in principle, be improved. This is the purpose of the following lines, where we will begin with an individual evaluation of each module on the test set in order to establish the strengths and weaknesses of the pipeline and try to identify areas for improvement.

3.8.2 Baseline Architecture Overview

As we can see in Figure 14, the pipeline consists of several interconnected modules.

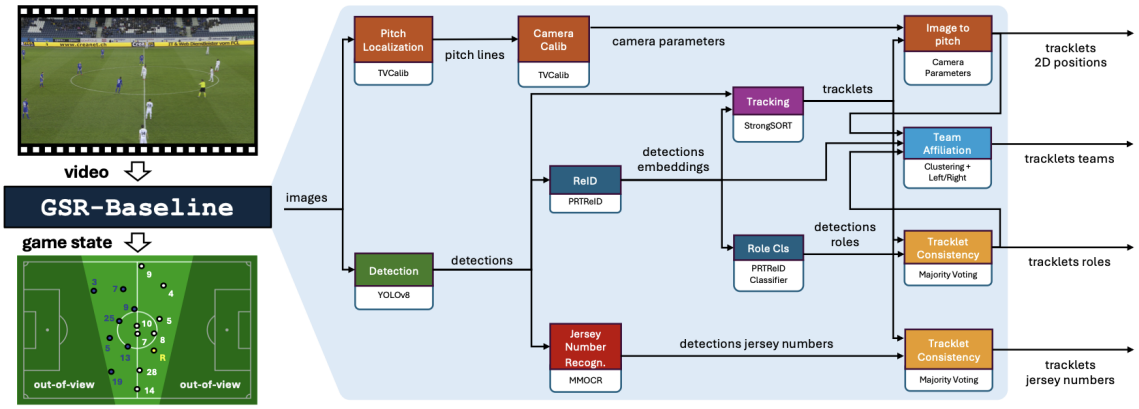


Figure 14: Pipeline Architecture from Soccernet GSR Paper [32]

It seems obvious that in order to gain a good understanding of the strengths and weaknesses of the model, each module should be evaluated in the order in which the input enters the pipeline. We therefore logically start by analysing the bounding box detection 3.8.3 and calibration modules, as they constitute the input to the pipeline. Once these two modules have been thoroughly analysed, we will evaluate re-identification, jersey number recognition and, finally, team evaluation.

3.8.3 BBox Detection

BBox detection in Multi-Object Recognition consists of returning the dimensions of a box that frames the object(s) to be recognised on a given frame. The baseline already includes YoloV8 as a model, more precisely the YoloV8x6 model, the heaviest model in terms of parameters and therefore inference time of the V8 from Ultralytics [37]. However, Ultralytics has just released a new model, V11, and claims to guarantee at least equivalent performance at much lower computational costs 15.

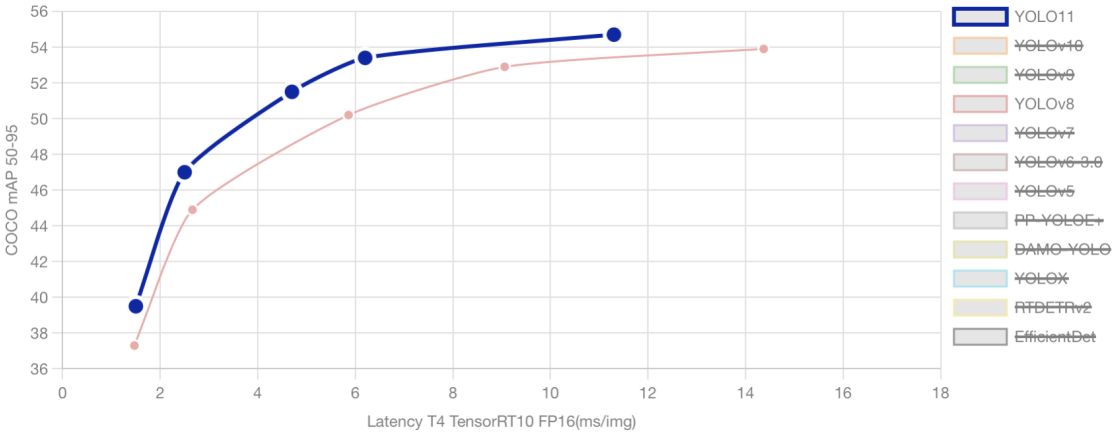


Figure 15: Performance benchmark of different YOLOv8 vs YOLOv11 models [38].

Given the performance results, it seems sensible to benchmark Yolov8x6, YOLOv11l and YOLOv11x directly. The three models will be evaluated on several metrics and will also include ball detection. The baseline does not currently provide for ball management, but as this is an essential part of scoring an action, we need to integrate it into the process. Even before any results are available, our initial intuition is that Yolo, in its classic pre-trained versions on COCO, will have great difficulty detecting the ball, which is very small (12pxX12px) and moves at relatively high speeds during shots and passes, and can therefore often appear blurred in the frames.

In order to ensure an unbiased estimator of model performance, each final evaluation will be performed on the Soccernet test set.

Model	$F1-Score_{Player}$	$Prec_{Player}$	$Recall_{Player}$	$AP@0.5_{Player}$	$AP@0.5_{Ball}$	**FPS**
Yolov8X6	0.943	0.936	0.950	0.887	0	10.55
Yolov11l	0.937	0.962	0.912	0.894	0	43.77
Yolov11x	0.933	0.966	0.902	0.901	0	32.29

Table 1: Performance of YOLO models on player and ball detection (test set)

The $mAP@0.5$, which corresponds to the average AP across all classes, is deliberately not shown here. Since the AP@0.5e for the *ball* class is zero for all models, due to the total absence of correct detection, including this value in an overall average would have masked the true performance on player detection.

Three main conclusions can be drawn directly from the results: 1) the models are very similar in terms of detection accuracy; 2) as expected, they are all unable to detect the ball with the version pre-trained on COCO; 3) the differences in performance in terms of time are quite significant.As a result, we will reject YOLO v8, which unfortunately has a precision/performance trade-off well below the other two. Based on this single trade-off, it would be clear to reject YOLOv11x as well; however, although slower than YOLOv11l, it remains perfectly acceptable in terms of time. Given the complexity of detecting small objects in an image, it seems premature to exclude it at this stage. We will therefore continue with the two v11 models, trying and evaluating several strategies to enable them to detect the ball in our frames.

In light of the previous results (see Table 1), it is clear that none of the models pre-trained on COCO are capable of correctly detecting the ball, with a AP@0.5 strictly zero on all tested models. This finding is not surprising: the ball in a match sequence is an extremely small object (often around 12×12 pixels) and moves at relatively high speeds, especially during passes or shots, which often makes it blurry or distorted in images. At the same time, the COCO dataset, on which the pre-trained versions of the YOLO models are based, contains no contextual equivalents, either in terms of size or in terms of visual environment (pitch, players, audience, etc.). These two factors combined are enough to explain the total failure of generic models on this class, which is nevertheless essential to any action recognition system. It is therefore necessary to implement a dedicated fine-tuning strategy, with a dataset specifically built around ball detection.

Given the small size of the ball and its random dispersion in the image, learning from complete images quickly proved ineffective: the proportion of pixels that are actually useful for learning is tiny, and the signal quickly becomes drowned out by the visual noise of the scene. To train a model specialised in ball detection, it was therefore necessary to build a dataset tailored to this very specific task. The approach adopted consists of dynamically extracting, for each image containing a small ball, tiles with a resolution of 384×384 centred on the relevant regions. Tiling is performed with an overlap of 128 pixels so as not to miss balls positioned at the edge of the area. To guarantee the quality of the labels and avoid noisy learning, only objects visible at more than 70% of their actual surface area are retained. Finally, to avoid any bias, the dataset is balanced to 80% of tiles containing a ball and 20% containing no ball, the latter being chosen randomly from among the images containing a ball annotation. The final format complies with the structure expected by YOLO, with an explicit separation of images and labels by split (`train/val/test`). This tiling strategy is supported by previous research, such as the study by Unel et al. (2019), which demonstrates that tiling significantly improves the detection of small objects by reducing scene complexity and increasing the relative resolution of objects of interest [27]. A typical example of this strategy is illustrated in Figure 16, where we see that a ball that is difficult to spot in the full image becomes perfectly visible once isolated in a centred tile.

Even before starting any training, a manual analysis of the dataset revealed several problematic cases in the ball annotations. The ball is often annotated when it is completely

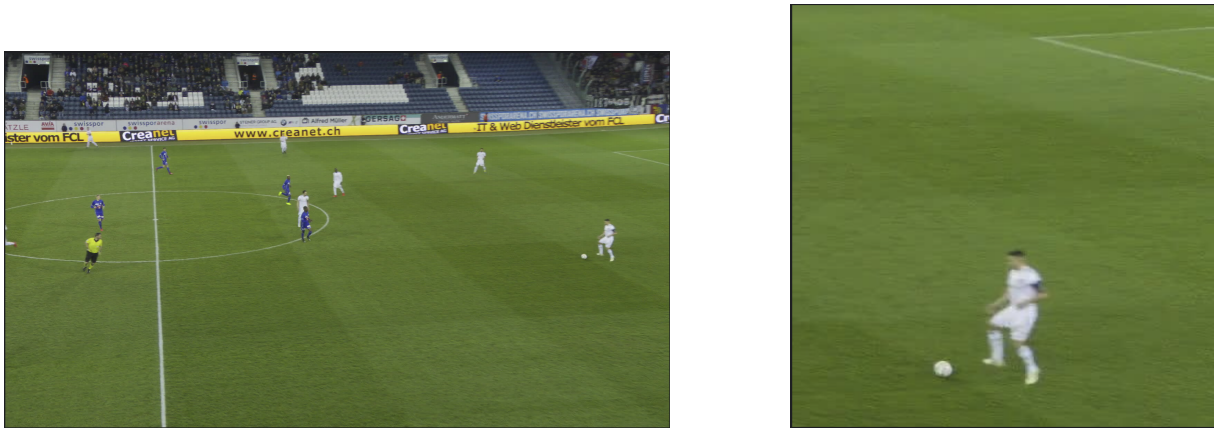


Figure 16: Left: complete image from the SoccerNetGS dataset. Right: centred tile of 384×384 pixels containing the ball.

or partially obscured by a player, making it almost impossible to detect, even for a human observer. This means that the results obtained by the models will likely be underestimated by traditional metrics such as mAP, since a number of false negatives are not actually attributable to the model.

Figure 17: Typical example of ball annotation that is difficult to use. The ball is present in the annotations, but completely hidden by a player.

Several strategies were then considered to train a high-performance model. A first attempt consisted of using a dual-head YOLO model capable of detecting both players and the ball in a single pass. However, this approach was quickly abandoned: the model was trained on highly zoomed tiles, which severely disrupted the learning of the *Person* class by altering the natural distribution of complete images. To avoid this confusion, the dual-head approach was discarded in favour of two specialised models, each trained on data tailored to its respective task.

The model chosen for the ball is YOLOv11l, fine-tuned exclusively on the generated tiles. Training was conducted over 50 epochs, with a batch size of 8, a learning rate of 0.001, and an image resolution set to 384×384. The model dedicated to players, on the other hand, remains trained on the original complete images, without tiling.

We achieve a mean Average Precision (mAP) of 0.67 for ball detection. While this performance is not outstanding, we anticipate that the most critical cases will still be handled adequately and will not significantly hinder our downstream analysis. A more in-depth discussion of these limitations and their potential impact will be provided in the Results section.

Post-processing analysis of the predictions showed that the vast majority of errors in the ball model occur in two recurring cases: either when the ball is in the air — and therefore located in front of a visually complex background such as the stands or the audience — or when it is moving very quickly, appearing blurred or even distorted. In both situations, the combination of movement and loss of contrast with the ground makes detection very difficult. Despite these limitations, the performance achieved remains more than adequate for our pipeline, which aims to locate the ball mainly at the beginning and end of actions, when it is usually on the ground, therefore well contrasted and not moving much.

Even with a model fine-tuned on centred tiles, detecting the ball in a complete image remains a complex task. In each frame, it may appear tiny, blurred, or positioned in a very busy visual environment. To address this constraint, an adaptive inference strategy has been implemented via a dedicated class, `BallInferenceEngine`, designed to maximise the chances of finding the ball while limiting the computational cost. This class orchestrates the joint inference of two specialised models: a first YOLOv11l model dedicated to player detection (*class Person*), applied systematically to the entire image, and a second model, also based on YOLOv11l, finely trained on centred tiles to detect only the ball.

The logic is based on a simple temporal heuristic. When a ball prediction is available at the previous frame, the engine attempts to exploit its spatial continuity by extracting a tile centred on the last known position. This tile is then passed to the ball model, enabling fast and targeted detection. This strategy is particularly effective in possession or slow transition phases, where the ball remains close to the ground and moves gradually.

If no detection is made, or if no previous prediction is available (e.g., at the beginning of a sequence or after a tracking loss), the engine automatically switches to *fallback* mode. The image is then divided into a set of 384×384 tiles generated by sliding scanning, with 25 % overlap. Each tile is analysed individually by the ball model, and valid predictions are projected into the global coordinate system of the image. This fallback mode guarantees complete coverage of the visual space, at the cost of slightly higher inference time.

This adaptive inference therefore reconciles accuracy and performance: the player model ensures reliable detection of human positions at all times, while ball detection is dynamically adjusted according to the context. By exploiting temporal continuity whenever possible, we reduce the number of tiles to be processed, while maintaining satisfactory robustness in complex phases. All of this logic is now natively integrated into our pipeline, without the addition of external modules.

3.8.4 Calibration

Context and objectives

Context and objectives Camera calibration is an essential step in our pipeline. It allows us to link the coordinates of an object detected in an image to its actual position in the field, particularly in the ground plane ($z = 0$). This is because image coordinates alone cannot be used directly for spatio-temporal or tactical analysis. The objective is therefore to estimate, for each frame, the extrinsic parameters of the camera (position and orientation), so that any point detected in the image can be reprojected to a physical position on the ground. In the rest of this section, we detail the method used to estimate these parameters and analyse the errors.

paragraphMethod used: TVCalib The calibration process used in this work is illustrated in Figure 18. It is based on the TVCalib method [35], applied in a default configuration without fine-tuning.

The pipeline consists of two main steps:

- **Pitch Line Recognition**: a segmentation model analyses each frame to detect the lines on the pitch (touchlines, centre line, penalty box, etc.). The result is stored in a Python dictionary in memory, associating each `image_id` with the types of lines detected and their key points in image coordinates.
- **Camera Calibration**: the detected lines are then compared to the geometric model of a standard field. TVCalib optimises the extrinsic parameters of the camera (rotation matrix and translation vector) for each image, so as to align the detected lines with the projected lines of the model.

The module output is a dictionary containing the estimated camera parameters for each `image_id`. These parameters are then used in the following steps to re-project the detected objects onto the terrain plane.

Initial assumptions and evaluation protocol As mentioned above, all analyses and comparisons carried out in this section are based exclusively on the *validation set*. The *test set* is strictly reserved for the final evaluation of the pipeline, in order to avoid any bias and ensure a representative evaluation.

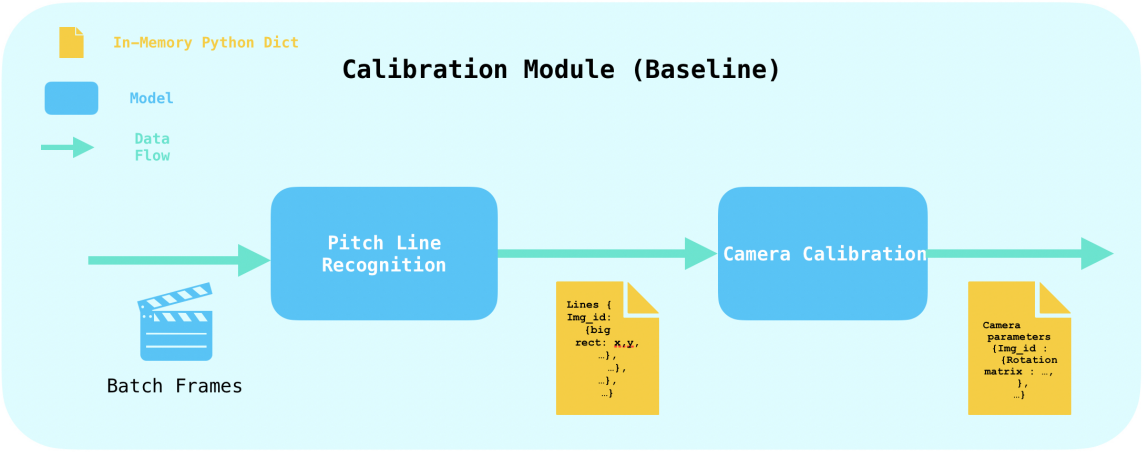


Figure 18: Representation of Calibration in Baseline

Before presenting the experimental results, it is relevant to formulate an initial intuition about the potential limitations of the system used. TVCalib was designed and trained to calibrate videos from multi-camera configurations. This framework implies that the model processes each image independently, without exploiting the natural temporal consistency present in a sequence from a single camera.

This lack of exploitation of temporal continuity is a structural limitation of TVCalib in our use case and suggests a relevant area for improvement: the introduction of a temporal stabilisation mechanism.

In order to objectively evaluate the raw performance of TVCalib, independently of the other modules in the pipeline, we have set up a specific evaluation protocol. For each frame, the estimated camera parameters are extracted. Then, from the available annotations, we retrieve the pairs of correspondences between *image* positions and *pitch* positions of annotated objects. By projecting the image coordinates onto the ground plane using the estimated parameters, we calculate the reprojection error as the average distance between the projected positions and the actual annotated positions on the ground.

Analysis of raw calibration First, as mentioned in the EDA section, the annotation file is incomplete; we only have pitch and image annotations for exactly 42,675 frames, whereas 58 clips of 30 seconds at 25 FPS should give us 43,500. Therefore, only the model's performance for frames with available annotations will be evaluated.

Several observations are apparent from the initial analysis. First, there is a very high variance; the significant difference between the mean and median, as well as the maximum error observed, clearly indicate that the model can completely misinterpret certain frames. This highlights the presence of genuine outliers, often linked to poor line detection.

Second, we note that when lines are detected correctly, performance remains satisfactory overall, as evidenced by the relatively low median error.

Statistics	**Value**
Total number of frames (GT ∩ Preds)	42,675
Number of valid frames (measured error)	41,496
Number of frames with invalid calibration (NaN)	1,179
Statistics on valid errors	
Average reprojection error	10.62
Median reprojection error	4.80
Minimum error	0.54
Maximum error	17,866.74
Standard deviation	111.54
1st quartile (Q1)	3.82
3rd quartile (Q3)	6.22
Percentile 5 (P5)	2.79
Percentile 95 (P95)	24.96

Table 2: Initial calibration statistics (TVCalib) on the validation set

Finally, it is essential to include NaN values in the evaluation of this baseline. Cases where TVCalib fails to estimate the camera parameters should be considered as severe failures. To give them a realistic weight in the overall statistics, we decide to assign these frames a fixed error equal to the 95th percentile (P_{95}) of valid errors. This strategy allows us to value these calibration failures while ensuring a consistent evaluation of the overall model performance.

This gives us the following values:

Error statistics	**Value**
Average reprojection error	11.02
Median reprojection error	4.87
Minimum error	0.54
Maximum error	17.866.74
Standard deviation	110.01
1st quartile (Q1)	3.85
3rd quartile (Q3)	6.48
5th percentile (P5)	2.81
95th percentile (P95)	24.96

Table 3: Initial calibration statistics (TVCalib) on the validation set with NaN penalty

We observe that after integrating uncalibrated frames with a P_{95} penalty, the average error increases slightly, from 10.62 to 11.02. This increase is logical and expected, since these frames represent model failures and must be treated as severe cases.

The median error remains virtually unchanged (from 4.80 to 4.87), confirming that correct predictions remain in the majority. Similarly, the standard deviation varies very little (111.54 to 110.01), showing that the addition of penalties did not cause excessive imbalance in the distribution.

These results confirm the relevance of choosing P_{95} as the penalty value: it reflects the impact of failed calibrations while maintaining an accurate representation of the model's

overall performance.

We can now implement different improvement strategies and objectively compare their performance with that of the baseline. But before doing so, it is important to understand the critical situations for the model, whether they are outliers or frames with NaN parameters, in order to design appropriate improvements.

We will therefore explore the error distribution in more detail, with the main objective of answering the following questions: 1) Are these critical situations distributed evenly across the different clips? 2) When they appear in a given clip, are they scattered randomly or grouped together in time? 3) Between these problematic cases, do we observe frames where the error becomes low again, or are entire sequences degraded?

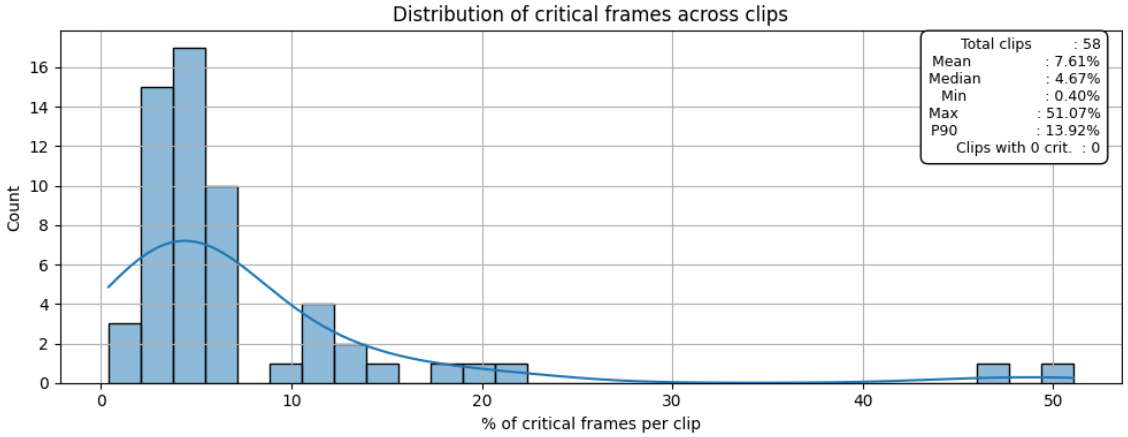


Figure 19: Distribution of critical frames per clip. Most clips have a low proportion of critical frames (error $>$ 25 or failed calibration)

By setting the critical error detection threshold to 25, we observe that the distribution of the percentage of critical frames per clip (Figure 19) remains broadly concentrated around the median, with an average of 7.61% and a 90^{e} percentile of only 13.92%. This indicates that, apart from two extreme cases, most clips have a relatively moderate critical error rate.

This observation suggests that the line detection method used upstream remains generally reliable and consistent across clips. No structural trends emerge, which means that the performance of the calibration module does not appear to depend on the specific context of a clip. The problematic cases identified therefore appear to be local exceptions, rather than symptoms of a widespread flaw in the method.

To illustrate the diversity of behavior observed between clips, we have selected four representative cases, each corresponding to a different level of calibration quality.

- **Critical clip** (clip 55): has a very high rate of critical errors, greater than 50
- **Unstable clip** (clip 85): contains many critical sequences, but interspersed with phases where the error drops sharply. This is a typical case where temporal correction could be effective.

– **Median clip** (clip 96): is around the median in terms of percentage of critical frames. It has a generally correct calibration with some localized fluctuations.

– **Stable clip** (clip 23): very low critical error rate, representing a well-calibrated case.

Figure 20 shows the evolution of the reprojection error for each of these clips, frame by frame, on a logarithmic scale. The threshold set at 25 is represented by the red dotted line. This visualisation provides a better understanding of the temporal dynamics of the errors and allows the feasibility of a correction strategy using interpolation to be assessed.

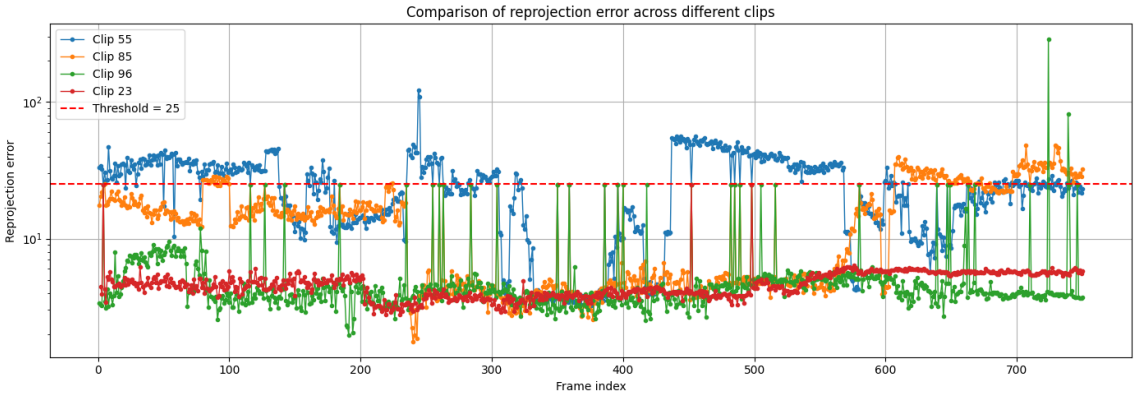


Figure 20: Comparison of reprojection error over time for four clips (log scale). While clip 55 consistently exceeds the threshold, clips like 85 show intermittent spikes, leaving room for correction. Clips 96 and 23 remain well-calibrated.

The extreme case (clip 55) shows a consistently high error, revealing a structural failure — likely due to an insufficient number of detected lines. However, these cases remain marginal and only affect two clips in the entire validation set.

The so-called 'bad' clips (such as clip 85) have well-identified critical sequences, but also natural recalibration phases where the error drops significantly. These observations confirm that, apart from extreme cases, calibration errors are often localised and temporary.

This validates the hypothesis that a strategy of improvement by temporal interpolation (or posterior correction) based on outlier detection would be both realistic and effective.

To be sure, let us consider the overall distribution of critical frame frequencies 4

Statistic	**Value**
Total number of critical sequences	1,642
Average sequence length	1.97 frames
Median sequence length	1.00 frames
Maximum sequence length	256 frames
Sequences longer than 10 frames	32
Sequences longer than 25 frames	15

Table 4: Statistics on the length of critical error sequences across all clips (error $>$ 25 or failed calibration).

This table further reinforces the hypothesis of temporal interpolation to correct outliers

and NaN. Indeed, the median shows us relatively low critical frame sequences (1). However, we know in advance that this strategy will not significantly improve our two critical clips, which are characterised by much too long critical frame sequences. Handling outliers and NaN As discussed above, our entire strategy will consist of managing outliers and NaN. Since for each clip, we can assume temporal consistency between frames $t-1$, t and $t+1$, it seems logical to be able to replace the value of the problematic frame t by interpolating its closest reliable neighbours. This brings us to our first problem: while it is easy to detect a NaN frame since it is indicated by the calibration module, detecting outliers requires a little more ingenuity. Initially, we explored a simple method for detecting outliers by analysing the evolution of camera parameters between two successive frames. The idea was that in a single-camera context, parameters such as position or angles (pan, tilt, roll) should evolve smoothly over time without the breaks that can be caused by a camera change. We therefore measured the frame-to-frame variations of these parameters and considered frames with abnormally large variations as outliers.

However, this approach has proven to be unreliable. On the one hand, some significant variations may be entirely legitimate, particularly in the event of camera movement or sudden changes in perspective. On the other hand, this method tends to miss cases where the camera 'hallucinates' in a stable manner, producing significant errors but without visible variations from one time step to the next. In summary, an approach based solely on temporal derivatives is too short-sighted and does not capture the complexity of calibration errors well.

It was therefore essential to find a much more robust method for outlier detection that fully considered the available information. Outlier detection in time series is a well-documented topic in the literature. In our case, it is more specifically the detection of local anomalies. Among the most effective methods are complex architectures such as Anomaly Transformer [41], TadGAN [13] and USAD [1]. These methods are capable of detecting subtle or structural anomalies, even in noisy or non-stationary contexts.

However, these methods are relatively heavy and seem, at first glance, to be relatively overkill (disproportionate) for our problem. In our case study, the calibration errors to be detected often manifest themselves as clear breaks or obvious aberrant behaviour in the camera parameters. We have therefore chosen to favour a simpler and more interpretable approach in the first instance.

We thus trained an XGBoost classifier [5]. However, a small feature engineering step was necessary to make our inputs more expressive so that the model could effectively detect outliers. For each frame t, we extract the predicted camera parameters:

$$\{p_t, \theta_t, r_t, x_t, y_t, z_t\}$$

corresponding respectively to the pan, tilt, roll and 3D position of the camera.

In order to capture the temporal dynamics of these parameters, we calculate:

– **The first temporal derivatives (denoted d_*)** :

$$d_{p_t} = p_t - p_{t-1}, \quad d_{\theta_t} = \theta_t - \theta_{t-1}, \quad \text{etc.}$$

These derivatives allow us to detect sudden changes in the camera's movement.

– **The second derivatives (noted dd_*)** :

$$dd_{p_t} = d_{p_t} - d_{p_{t-1}} = (p_t - p_{t-1}) - (p_{t-1} - p_{t-2})$$

They reflect sudden accelerations or breaks in dynamics, which are characteristic of outliers.

This type of modelling is relevant in our context, as we can assume that a physical camera tends to move smoothly. Sudden or erratic variations between two consecutive moments are therefore good indicators of incorrect or unstable calibration. Once this feature engineering step has been completed, we then have a feature vector for each frame that reflects not only the static state of the camera, but also its local dynamics. We also took advantage of the information provided by the line recognition module by giving the model different features such as the type of lines detected, the number of points detected per line, and the number of lines detected per frame. This allows the model to focus on outliers related to structural problems, leaving aside those inherent to hallucinations. We can therefore formulate outlier detection as a supervised binary classification problem: the goal is to predict, based on these features, whether a frame has a calibration error high enough to be considered abnormal.

To do this, we trained an XGBoost model [5], automatically labelling frames as outliers if their reprojection error exceeded a certain threshold. Several threshold values were tested (10, 15, 20, 25 m), as well as different class weights to compensate for imbalance. Finally, we explored various decision thresholds to select suspicious frames to be discarded or corrected by interpolation using a classic GridCVSearch algorithm.

In our case, the central objective of the outlier detection model is to **not miss any critical cases**. It is therefore essential to **minimise false negatives** (FN), i.e. defective frames that the model would fail to detect.

We therefore seek to maximise the *recall* on so-called 'bad' frames, even if this means tolerating a few additional false positives. This compromise is perfectly acceptable in our pipeline, since suspicious frames can then be interpolated or filtered.

Table 5 shows the best configurations tested for the XGBoost model, varying the reprojection threshold, class weighting, and decision threshold. All configurations achieve a recall greater than 99.4

These results confirm that the model can effectively detect abnormal frames while maintaining excellent accuracy on reliable frames. The configuration chosen for the rest of the study is the one with a threshold of 15 and a class weighting of 10, which maximises recall while keeping a moderate false positive rate.

Threshold	Pos. Weight	Recall (outliers)	Precision (outliers)	% of frames flagged
15	10	99.63%	74.39%	19.03%
15	5	99.49%	75.18%	18.80%
10	5	99.42%	86.78%	21.26%

Table 5: Best configurations of the outlier classifier (XGBoost), balancing detection recall and selectivity.

Although less sophisticated than the SOTA solutions mentioned above, this method already effectively identifies the most significant outliers, while ensuring fast training and low integration complexity in our pipeline.

Fix Invalid frame Now that we have developed a reliable way to detect outliers, we need to implement a strategy to estimate the camera parameters on these problematic frames.

We naturally start with the simplest and most intuitive method: 1D linear interpolation, performed independently on each camera parameter, using the closest neighbouring frames considered reliable. This approach already provides a correct estimate of the parameters in most cases.

To go further, we also explored the use of a *Kalman Filter* [20], a classic tool in state estimation, particularly suited to noisy or partially missing data. The filter models the evolution of camera parameters as a dynamic process, taking into account both the observed measurement and the prediction from the previous state.

The Kalman Filter is based on two main steps:

$$\begin{aligned} \text{Prediction:} \quad & \hat{x}_t^- = A \cdot \hat{x}_{t-1} \\ \text{Update:} \quad & \hat{x}_t = \hat{x}_t^- + K_t \cdot (z_t - H \cdot \hat{x}_t^-) \end{aligned}$$

where $\hat{x}_t$ is the estimated state (e.g., camera parameters), A is the transition matrix, z_t is the observed measurement (when available), K_t is the Kalman gain, and H is the observation matrix.

This method provides a smoothed and consistent version of the camera parameters over time, even in the presence of isolated outliers or NaN values.

We therefore recalculated the reprojection errors on all frames in the validation set, after post-processing by interpolation alone, then by interpolation combined with Kalman filtering. The overall results are presented in Table 6.

Method	**Mean**	**Median**	**Q1**	**Q3**	**Std. Dev.**
Interpolation only	6.35	4.68	3.75	5.89	13.47
Interpolation + Kalman	6.28	4.68	3.74	5.89	11.11

Table 6: Comparison of reprojection error statistics after interpolation, with and without Kalman filtering.

If we compare these results to the initial calibration of TVCalib (Table 3), several observations can be made.

Firstly, the average error drops from 11.02 to 6.28 after correction, a significant reduction of more than 40

Secondly, the standard deviation drops from 110.01 to 11.11, showing a massive reduction in variance. Calibration therefore becomes much more consistent and predictable over time. The median remains relatively stable ($4.87 \rightarrow 4.68$), which is consistent: frames that were already well calibrated are not affected, while extreme cases are corrected or mitigated.

In short, the system does not simply mask errors: it truly homogenises the calibration quality across the entire validation set.

Revisiting the structural limitations Although the previous analysis suggests that most calibration failures are local and can be corrected through temporal smoothing, a more in-depth examination of the clips reveals a structural weakness in the calibration system under specific conditions.

More precisely, we observe that the failures become systematic when the camera is positioned near the center of the field, preventing the detection of both lateral sidelines or one of the two penalty boxes. In such situations, the visible lines offer insufficient constraints for the geometric solver, leading TVCalib to converge toward implausible solutions—even if the detected lines themselves are correct.

This behaviour reflects a structural limitation of the calibration model: its reliance on a minimal set of diverse and well-distributed line types. When this condition is not met, even a stable visual context cannot prevent a drift in the estimated parameters.

While our correction strategy remains effective in the majority of cases, it cannot recover from these degenerate configurations. A more robust solution would require either an explicit detection of these ambiguous views, or the integration of prior knowledge about plausible camera placements to regularize the estimation process.

In summary, although our interpolation and filtering pipeline significantly improves the calibration quality overall, a small subset of cases reveals deeper structural issues that are currently beyond the scope of post-hoc correction.

Conclusion The results presented in this section confirm that the strategy implemented for detecting and correcting critical frames (outliers or NaN) works robustly and consistently. Without modifying the core of TVCalib, we have significantly improved calibration stability by automatically and selectively correcting weak points locally.

The use of simple but effective methods such as linear interpolation or Kalman filtering allows us to achieve a much more consistent calibration, while maintaining low errors on initially well-calibrated frames. This approach thus provides a solid basis for future integration into our pipeline.

3.8.5 OCR Jersey Number Recognition

Initial attempts to integrate an OCR system based on the `SoccerNet` baseline proved unsuccessful. Two modules were proposed: one based on `MMOCR` and the other on `EasyOCR`. The first was quickly discarded due to major incompatibilities between the required versions of `PyTorch`, `mmcv`, and the GPU components on macOS. Despite numerous attempts (downgrading, reinstalling, manual builds), the environment remained unstable. The second, `EasyOCR`, only worked on CPUs, but the processing time per image exceeded 2 seconds, making any inference on sequences of several hundred images much too slow.

We therefore opted for a modular reimplementation inspired by the approach presented in the paper by Koshkina et al. [21]. Rather than replicating their entire pipeline, we extracted only the modules relevant to our use case: the readability classifier, the pose estimator, and the OCR module. The other components of their framework, notably tracklet management and re-identification, were deliberately omitted as they are already supported in our own game state reconstruction pipeline (see Section 3.8.2).

Recognition pipeline Our jersey number recognition module works in three steps, each encapsulated in a dedicated class:

- **Readability filtering:** for each frame, the player crops are extracted from the bounding boxes provided by our detection module. Each crop is then sent to a binary classifier trained to distinguish players for whom a number is legible. This model is based on `ResNet34` pre-trained on `ImageNet`, whose last layer has been replaced by a linear layer to produce a sigmoid output. The model weights are those provided by the authors for `SoccerNet`, and no fine-tuning was performed.
- **Pose estimation:** Crops deemed readable are then sent to a pose estimator. Unlike the authors' initial approach, we used a newer `YOLOv11-Pose` [18] model, which is more easily compatible with our execution environment (M1/M2 GPU via MPS). This model provides the coordinates of the 17 COCO keypoints, from which we extract the two shoulders and the two hips. When all four points are available with sufficient confidence, an accurate bounding box for the torso is generated. Otherwise, a fallback strategy is applied based on the remaining valid keypoints.
- **Text recognition (OCR):** Finally, the torso areas are sent to a `PARSeq` [3] model, a sequential transformer designed for natural image text recognition. We use the `parseq_epoch=24-step=2575` checkpoint, pre-trained by the authors. This model predicts a sequence of characters, and in our case, we extract only the digits that make up the jersey numbers (often 1 or 2 digits), accompanied by a confidence score.

Integration into our pipeline Our implementation has been designed to work autonomously and modularly, with batch processing of images. The module takes an RGB image as input, along with a YOLO annotation file containing the bounding boxes of the players. For each frame, it returns a list of detected numbers and their reliability scores.

This approach allows us to seamlessly integrate OCR into our complete pipeline, relying solely on the outputs of the detection module.

3.8.6 PRTReid

The `PRTReid` module has been kept as is from the baseline proposed by Koshkina et al. [21], without any modifications on our part. This choice is not due to a lack of interest in the module's logic, but rather a reasonable compromise in the context of a complex pipeline, where it was not feasible to completely redevelop or requalify everything. This is especially true given that the approach proposed in the paper is distinguished by its overall consistency and highly satisfactory empirical results.

The module is based on a *Detection-Level Module* architecture, which acts on the predictions of each frame independently, without explicit use of a temporal memory. It comprises three key sub-modules, which run in cascade to progressively enrich each player detection with the attributes necessary for the complete reconstruction of the game.

- **Re-identification**: based on the visual appearance of the crop (typically the torso), an encoder generates a fixed-dimension embedding that is used to stably identify a player across frames. This vector is then used for the matching phase.
- **Visibility estimation**: each detection receives a visibility score between 0 and 1, based on a binary classifier. This score is used to weight matching decisions in case of ambiguity or partial occlusion.
- **Classification role**: a supervised model assigns each detection a role from a set of predefined classes (goalkeeper, referee, outfield player, etc.). This information is useful for the structural consistency of the game and for downstream action evaluation modules.

This module is therefore responsible for the semantic enrichment of each detected player: it does not perform tracking per se, but provides all the elements necessary for a matching-based tracking system. In our pipeline, it fits naturally between detection and the game state reconstruction (GSR) phase, adding these intermediate attributes.

Given the technical complexity of this module and the consistency of its initial implementation, we decided not to touch it. The authors report in their article an overall *GS-HOTA* score of 78.2 on SoccerNet-v2, with complete reconstruction of player identities, roles and locations. This level of performance, combined with the clarity of the code, seemed sufficient to us to be used as is, without questioning it in the context of our experiment.

3.9 Valuing Action

3.9.1 Introduction and objectives of the VAEP model

In football performance analysis, traditional statistics such as the number of goals, assists or successful tackles are of little use in assessing a player's real impact on the course of a match. These indicators are often insufficient to quantify actions that, although neither spectacular nor directly decisive, nevertheless strongly influence the probability of scoring or conceding a goal. This is where the **VAEP** (*Valuing Actions by Estimating Probabilities*) model, proposed by Decroos et al., comes in. Its objective is to *objectively evaluate each action in a match.*

The VAEP principle is based on a simple but powerful idea: each action can be evaluated according to its impact on the future chances of scoring or conceding a goal. More specifically, the model predicts for each game state:

- the probability that the team in possession will score in the next k actions (P_{score}),
- the probability that the same team will concede a goal in the same time frame ($P_{concede}$).

The **VAEP value** of an action a_i is then the difference between these probabilities before and after the action:

$$V(a_i) = \underbrace{P_{score}(S_i) - P_{score}(S_{i-1})}_{\text{offensive value}} \quad - \quad \underbrace{P_{concede}(S_i) - P_{concede}(S_{i-1})}_{\text{defensive value}}.$$

Thus, an action is considered valuable if it increases the chances of scoring or decreases the chances of conceding. Conversely, a bad action (loss of possession, poor pass, foul, etc.) will receive a negative value because it compromises the team's objectives.

In this work, we have chosen to completely reconstruct the VAEP model in order to:

1. train our own predictive models P_{score} and $P_{concede}$ using enriched action data;
2. integrate new **tactical features** from contextual data (player position, pressure, numerical superiority, etc.) in order to better capture collective dynamics;
3. offer a detailed analysis of the contributions of each feature via calibration curves, importance analyses and concrete examples.

The objective is twofold: to have a robust tool to quantify the real impact of game actions, and to evaluate the relevance of certain tactical variables (please specify to help the reader) to refine performance analysis. This approach complements the computer vision models presented above by providing a purely symbolic, statistical and contextual reading of the game.

3.9.2 Data used and SPADL format

The data used in this section has already been explored in detail in the exploratory analysis phase (section 3.4.2). It comes from the provider StatsBomb, in `events` format, and covers several hundred professional football matches. To make this data usable in the VAEP model, it is transformed into the **SPADL** (Soccer Player Action Description Language) format, a standard proposed by Decroos et al. to describe each action in a unified way.

The SPADL format allows each action to be represented using a set of structured columns:

- the type of action (pass, shot, dribble, etc.);
- the result (successful, unsuccessful, blocked, etc.);
- the part of the body used;
- the start and end coordinates of the action, normalised in a game plane $[0, 105] \times [0, 68]$;
- the player and team involved;
- the time elapsed since the start of the match.

Each line of the SPADL table therefore corresponds to an elementary action. From these actions, we construct **game states** S_i describing the context before a given action a_i. In accordance with the original work by VAEP, each state S_i is defined from the last three actions of the team in possession before a_i (sliding window). This choice allows us to capture both the current action and the immediate dynamics of the game.

To train the probability models P_{score} and $P_{concede}$, each action is labelled according to what happens in the following $k = 10$ actions:

- $y_{score} = 1$ if the team in possession scores in these 10 actions;
- $y_{concede} = 1$ if this team concedes a goal in this time frame.

This labelling allows the task to be reformulated as a supervised binary classification problem for each model.

Finally, to enrich the model, certain **advanced contextual features** were calculated based on the positions of the players at the time of each action. This data comes from *freeze frames* provided by StatsBomb 360, which indicate the positions of the players visible at the moment of the action. Although this data is partial (some players may be out of frame), it allows us to calculate useful indicators such as defensive pressure, passing opportunities, or numerical advantages around the ball. These aspects will be detailed in the next section.

It is important to note that, for reasons of compatibility with the spotting model presented in section 3.7.10, our valuation model only takes into account a **restricted subset of action types**. More specifically, we will only predict VAEP values for the following classes: *Assist, Drive, Header, High Pass, Out, Cross, Throw In, Shot, Ball Player Block, Player Successful Tackle, Free Kick, Goal.* This choice is dictated by the classes actually recognised by the automatic event model.

As a result, a **remapping of action types** from SPADL was necessary to align the action categories in the dataset with those predicted by the spotting model. For example, several

types of passes present in SPADL (such as `Short Pass`, `Smart Pass`, etc.) can be grouped under the category *High Pass* or *Assist* depending on the context. This mapping is essential to ensure consistency between the valuation of actions (VAEP model) and their automatic detection (spotting model).

3.9.3 Model input features

To estimate the probabilities P_{score} and $P_{concede}$ using supervised classifiers, each action is represented by a vector of explanatory variables describing the context in which it occurs. We distinguish between two categories of features: those derived directly from the SPADL format, and those we have designed based on contextual data, in particular player positions.

Features derived from SPADL The features extracted from the SPADL format describe the action itself and its immediate spatio-temporal dynamics. They include, in particular:

- **Action type (one-hot)**: binary encoding of SPADL action types (pass, shot, tackle, clearance, etc.). As mentioned above, these types have been remapped to correspond to the classes supported by the spotting model.
- **Result of the action**: binary (successful/failed) or categorised (success, intercepted, blocked, etc.).
- **Part of the body used**: foot, head, other.
- **Start and end coordinates** of the action on the pitch.
- **Distance and angle to goal** from the starting point and end point of the action.
- **Distance travelled by the ball** between the start and end of the action.
- **Time elapsed since the previous action**, to capture the tempo of play.

These variables, standardised in classic VAEP implementations, provide a robust basis for capturing the direct effects of an isolated action (e.g. distance of a shot, closed angle, etc.).

Advanced contextual features To better represent the collective and spatial impact of an action, we have enriched the basic descriptors with a set of **complex contextual features**, derived from *freeze frames* provided by StatsBomb 360. These features aim to quantify fundamental concepts of the game such as defensive pressure, available passing options and positional imbalances, based solely on the relative positions of the players visible at the moment of the action.

- **Numerical Superiority**: measures the number of teammates and opponents located between the ball carrier and the opposing goal. An offensive numerical advantage in this area is often correlated with a dangerous opportunity. Two features are extracted: the number of *teammates ahead* and *opponents ahead*, defined as players whose abscissa is strictly greater than that of the ball carrier.

– **Pressure Ratio**: quantifies the variation in defensive pressure experienced by the ball carrier before and after the action. Pressure is defined here as the number of opponents located within a given radius around the ball carrier (typically 5 metres). We note this difference:

$$\Delta P_i = \text{count}_{\text{before}} - \text{count}_{\text{after}}$$

A positive value indicates a reduction in immediate pressure, often associated with a successful dribble or an effective shift.

– **Passing Opportunities**: measures the number of teammates who are reachable at the moment of the action. A pass is considered open if (1) the line between the ball carrier and the receiver is not blocked by an opponent (proximity to a segment), or (2) if the pass is long, the receiver is deemed sufficiently free (no opponents nearby). This feature therefore incorporates both the readability of short trajectories and the freedom of the receiver for long passes.

– **Visibility Confidence**: two confidence scores are calculated to assess the completeness of freeze frames: one based on the number of visible opponents (compared to an expected maximum of 10), the other on the total number of visible players other than the ball carrier (relative to 21). These scores allow contextual features to be weighted according to the reliability of the available spatial data.

The integration of these descriptors makes it possible to capture essential tactical dynamics that are often absent from traditional tabular representations. They thus complement the SPADL features by providing contextual insight into the structure of the game at the moment of each action.

4 Result

4.1 Introduction

In this section, we provide a more in-depth evaluation of the results obtained in the various tasks of the pipeline. It should be noted from the outset that this analysis cannot be solely quantitative or systematic. Indeed, several constraints limit the possibility of a rigorous empirical evaluation.

Firstly, the available annotations do not allow us to directly measure the performance of the spotting module, as there is no explicitly annotated dataset for this task. Second, with regard to the evaluation of actions, assessing the quality of the model based solely on the accuracy of the predicted labels would not be relevant, given the very nature of the task, which aims more to estimate the impact of an action than to classify it.

Given these limitations, we have opted for a qualitative evaluation of the results. This will focus on the concrete implications of the various design choices and modifications to the

pipeline. We will also analyse specific cases in order to better understand the relevance, but also the limitations, of the models used.

Finally, it should be noted that a first reading of the results already reveals certain limitations in the overall performance of the pipeline. The sequence of modules, each of which is imperfect, seems to lead to an accumulation of errors that makes the idea of a fully functional end-to-end system illusory at this stage. This observation needs to be qualified, but it prompts us to approach the analysis with caution and not to overinterpret the overall results before exploring the performance of each component individually.

4.2 Spotting

First and foremost, it is important to note that the spotting module cannot be evaluated using a dataset annotated for this task. In the absence of usable ground truth, it is therefore impossible to accurately measure traditional metrics such as mAP or recall.

However, this does not mean that the module is useless or that its behaviour is incomprehensible. In the absence of a strict quantitative evaluation, we propose a qualitative analysis of the outputs: nature of the events detected, temporal relevance of the predictions, and the most frequent types of errors.

Our goal is not to assert the superiority of a model, but rather to understand what it captures—and, above all, what it does not capture. We will therefore attempt to judge the overall consistency of the predictions with respect to the source videos, as well as their practical usefulness in the rest of the pipeline (calibration, valorisation, etc.).

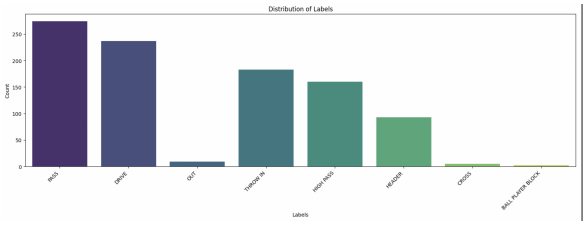
Although we do not have annotations that allow for a strict quantitative evaluation of our spotting module, it is possible to place our results in an indicative order of magnitude based on existing benchmarks. The T-DEED model, used here as a baseline without retraining, achieved a **Team-mAP@1 of 47.18** and a **Tight Average mAP of 57.53** according to the official results of the SoccerNet 2025 competition. These figures were measured on an annotated dataset containing numerous complete matches and sequences from television broadcasts with frequent camera changes.

In our case, the videos analysed are captured from a **single fixed camera**, without cuts or changes of shot. This removes a number of difficulties inherent in the task of spotting under typical broadcast conditions. In all likelihood, the model's performance in our context should therefore be at least comparable to that observed on the benchmark — or even slightly better, due to the absence of noise introduced by production effects.

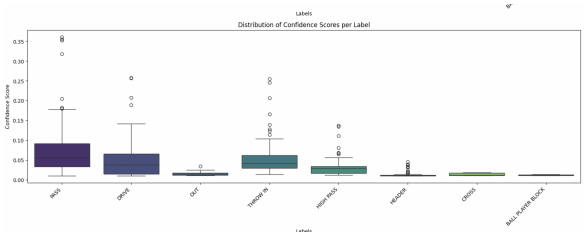
On average, the model detects **33.21 actions per second**, all confidence levels combined — a clearly unrealistic density for a football match. This shows a clear tendency towards overdetection, with significant noise in the raw predictions.

When a minimum confidence threshold is introduced to filter detections, a clear improvement in quality is immediately observed: the detected actions form more coherent sequences that more closely resemble readable phases of play. However, this selection by confidence

is not without side effects: it leads to a significant loss of **diversity** in the types of actions detected. As shown in the graphs below, only certain classes (notably *pass*, *drive*, or *throw-in*) are predominant at acceptable confidence levels, while others (such as *cross*, *ball-player block* or *header*) are almost completely absent.



(a) Distribution of predicted action labels.



(b) Distribution of confidence scores per action class.

Figure 21: Qualitative analysis of spotting predictions: label imbalance and low confidence on rare actions.

Furthermore, it is important to note that the model does not detect the start and end of an action in the traditional sense. Instead, it produces a sequence of labels associated with timestamps. Setting the confidence threshold too high therefore introduces a form of censorship that disrupts the game phase: some long actions are interpreted as a single event spread over several seconds, while others are simply ignored. Conversely, without a filter, the model tends to insert actions even in dead phases of the game (throw-ins, free kicks, waiting time, etc.), which leads to absurd detections.

This behaviour illustrates the limitations of unsupervised spotting in a single-model context: without time calibration or sequence logic, the model struggles to provide a fluid and usable reading of the match.

4.3 Tracking

Evaluation framework. For this task, we have an annotated test set, which enables rigorous quantitative evaluation using the GS-HOTA metric. In line with the baseline, we will directly compare the scores obtained before and after modifying our pipeline. The objective is not only to measure overall improvement, but above all to identify critical sequences: those where the score remains low or drops sharply following a modification.

In a second step, we will qualitatively analyse these problematic cases in order to understand the breaking points: which module is responsible for the degradation? Is it systematic or contextual? Conversely, we will also highlight certain sequences where the changes made have led to a clear improvement, in order to better understand the conditions under which our approach works best.

The challenge is therefore not only to do better, but to better understand why — and how — some reconstructed states are more reliable than others.

Comparison with the baseline. The GS-HOTA score obtained after modifications is **26.32**, compared to **22.26** for the initial baseline, representing a relative improvement of more than 18%. When compared to the results available on the official competition leaderboard[1], they remain well below the best submissions, which sometimes exceed 50 or even 60 GS-HOTA.

However, this significant gap is neither surprising nor alarming. It can be explained by several factors. On the one hand, the difference in resources is obvious: the majority of teams participating in the competition are made up of experienced researchers with significantly greater material resources, including large-scale GPU clusters. It would have been both unrealistic and presumptuous to aim for equivalent performance in an individual academic project.

On the other hand, the initial objective of this thesis was not to optimise the raw performance of the pipeline, but rather to exploit the baseline in inference mode in order to build a coherent processing chain. Only in a second phase were certain modifications explored—with a view to gradually appropriating the code and testing new modules, such as bullet detection (not taken into account in the GS-HOTA metric).

Finally, the modifications made relate exclusively to post-processing or heuristic adjustments. No structural overhaul of the model was undertaken. In this context, achieving a gain of more than 4 points on GS-HOTA remains an encouraging result. However, it would be unrealistic to expect to achieve the scores of the best submissions in the competition without deep training or a parallel pipeline.

Qualitative analysis. Qualitative analysis highlights a structural weakness in the pipeline, centred around the calibration module. This problem only became apparent to me at a late stage, when the reconstructions produced were being viewed in detail. Until then, the raw results seemed consistent, and there was no indication of any obvious error. It was only when observing certain specific sequences that the failure became apparent.

In relatively zoomed-in shots, where the touchlines or both rectangles of the pitch are not visible at the same time, the model loses its bearings. TVCalib, which is responsible for estimating the camera parameters from the visible lines, then has too few clues to orient itself correctly. This leads to complete hallucinations: players are projected outside the field, sometimes several metres from the pitch. This behaviour seems to be directly related to how the model works: the absence of a visible line is not used as information in itself, when it could instead strongly constrain the range of possible calibrations.

This fragility is obviously problematic in a multi-camera context, where each shot can vary greatly. In our case—with a single camera—it would have been possible to compensate for this lack of consistency by stabilising the calibration over several frames. Due to time constraints, this idea could not be tested or implemented, but it remains a promising avenue for future research.

[1] https://eval.ai/web/challenges/challenge-page/2251/leaderboard/5565

In comparison, errors related to pure detection (bounding boxes, number recognition, team affiliation) seem less frequent and more localised. They mainly occur in cases of heavy occlusion between players. Here again, introducing temporal continuity logic—for example, by requiring players to be present throughout all frames—would have limited certain abnormal breaks. Although simple to formulate, this type of temporal regularisation requires rigorous supervision to avoid smoothing out recurring errors, but could prove very beneficial in the medium term.

4.4 Valuing

4.4.1 Overall evaluation of the VAEP model

This initial analysis aims to evaluate the overall performance of the VAEP model on the entire dataset, without any prior filtering based on data quality or the addition of complex features. The objective here is not to directly predict the occurrence of a goal, but to estimate the influence that a given action could have on the evolution of the score. In this context, certain classic metrics such as the Brier score, Log Loss or AUC ROC can provide useful information on the calibration of the model, but they should not be considered an end in themselves.

Quantitative results. The model was trained separately to predict the probability that the ball carrier's team will score (*scores*) or concede (*concedes*) a goal following an action. The scores obtained on the validation set are as follows:

- **Scores (Y: scores)**: Brier score = 0.00951 (normalised: 0.84615), Log Loss = 0.04863 (normalised: 0.78196), ROC AUC = 0.81877.
- **Concedes (Y: concedes)**: Brier score = 0.00244 (normalised: 1.02832), Log Loss = 0.01863 (normalised: 1.11417), ROC AUC = 0.72998.

These results suggest that the model is relatively well calibrated, particularly for predicting scoring, which has better scores than predicting conceding. This imbalance is consistent with the more explicit nature of actions leading to a goal compared to those preceding a goal conceded.

Cumulative VAEP values per player. Table 7 shows the 10 players who accumulated the most VAEP value in total over the period observed. Unsurprisingly, several FC Barcelona players feature, including Lionel Messi, Xavi and Andrés Iniesta. This result can be explained in part by a representation bias, as the dataset contains a significant proportion of matches involving this team.

Table 7: Top 10 players by cumulative VAEP value (complete dataset)

Player	VAEP	Off. value	Def. value	No. of actions
Lionel Messi	482.47	508.13	-25.67	79,936
Xavi	94.69	96.80	-2.12	46,013
Andrés Iniesta	82.65	82.51	0.14	44,417
Sergio Busquets	82.49	82.68	-0.19	55,494
Gerard Piqué	68.00	63.96	4.04	43,025
Dani Alves	62.83	66.32	-3.49	37,222
Neymar	62.01	67.48	-5.47	20,554
Luis Suárez	55.14	57.68	-2.54	13,017
Javier Mascherano	47.29	45.23	2.06	24,452
Ivan Rakitić	39.86	40.48	-0.62	24,508

We can see that offensive profiles are particularly highlighted, but that the model also values certain more discreet defensive contributions, as evidenced by the values for Gerard Piqué and Javier Mascherano. This suggests that the model is capable of capturing different forms of impact depending on the roles.

VAEP per 90 minutes. To neutralise the effect of the volume of actions, we also calculated the VAEP value per 90 minutes played. This allows us to compare players with very different playing times, while identifying the most effective ones. The following table shows the best players according to this metric.

Table 8: Top 10 des joueurs selon la VAEP par 90 minutes

Joueur	VAEP/90	VAEP	Minutes jouées
Lionel Messi	0.8431	482.47	51 502
Gareth Bale	0.5462	19.89	3 278
Vivianne Miedema	0.5122	34.62	6 084
Riyad Mahrez	0.5046	19.00	3 389
Ronaldinho	0.5041	28.01	5 000
Zlatan Ibrahimović	0.5040	24.49	4 373
Francesca Kirby	0.4916	20.02	3 664
Dimitri Payet	0.4762	15.61	2 951
Ángel Di María	0.4711	27.03	5 164
Ji So-yun	0.4710	24.00	4 593

This metric highlights players who are more intermittent but highly effective in their interventions. It also includes several high-performing female players, demonstrating that the model remains consistent regardless of gender or playing time.

Discussion. Finally, it is important to note that some extreme values, although rare, are perfectly legitimate. A player such as Lionel Messi has VAEP scores that are significantly higher than all others, which could be perceived as a statistical anomaly. However, these values simply reflect the real and exceptional impact he has on his team's actions. This

observation serves as a reminder that in data science, outliers should not be systematically discarded: they may contain essential information or even be at the heart of the explanation sought.

4.4.2 Impact of freeze frames and complex features

After the overall evaluation, this section explores the impact of two factors on the performance of the VAEP model: (1) restricting the dataset to matches with at least 80 % valid *freeze frames*, and (2) adding so-called *complex* variables, which capture, in particular, the spatial structure around the ball carrier and the defensive configuration.

Quantitative performance. Table 9 presents the performance of the models on the restricted dataset, with and without the addition of complex features. We observe an overall improvement in scores, particularly with regard to predicting whether a goal will be conceded. These results suggest that the improved data quality and feature enrichment enable the model to better estimate the contextual impact of actions.

Table 9: VAEP model performance on the restricted dataset

Configuration	**Scores (Y: scores)**			**Concedes (Y: concedes)**		
	Brier	Log Loss	ROC AUC	Brier	Log Loss	ROC AUC
Without complex features	0.00864	0.04346	0.8566	0.00205	0.01205	0.8906
With complex features	0.00864	0.04342	0.8567	0.00208	0.01238	0.8882

SHAP analysis: before/after adding features. In order to better understand the model's decisions, a SHAP analysis was conducted on both targets. In the case of *concedes*, models without complex features rely mainly on the geometry of the action: distance and angle to the goal, final position on the field, or type of action. Figure 22 illustrates this hierarchy.

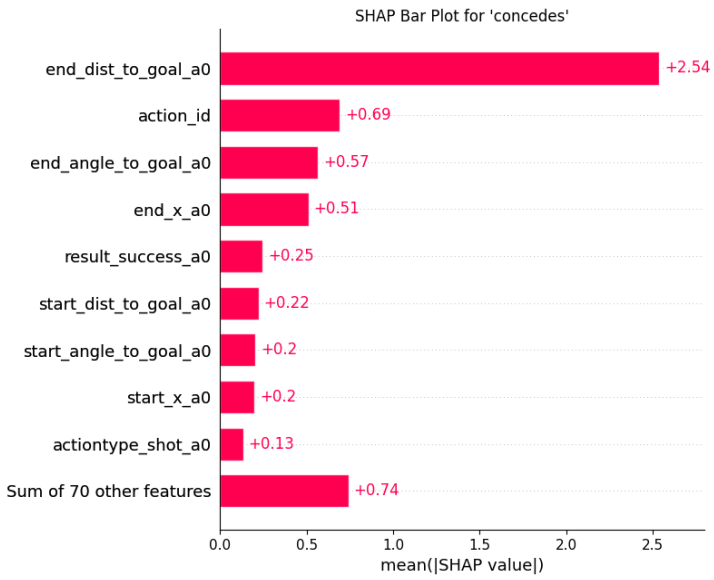


Figure 22: SHAP values (mean) – Concedes without complex features

After adding complex features, the situation changes significantly. Variables such as overall confidence in the context (`confidence_total`), the number of passing opportunities, and confidence in the opposing defensive structure now appear among the most influential, as shown in Figure 23.

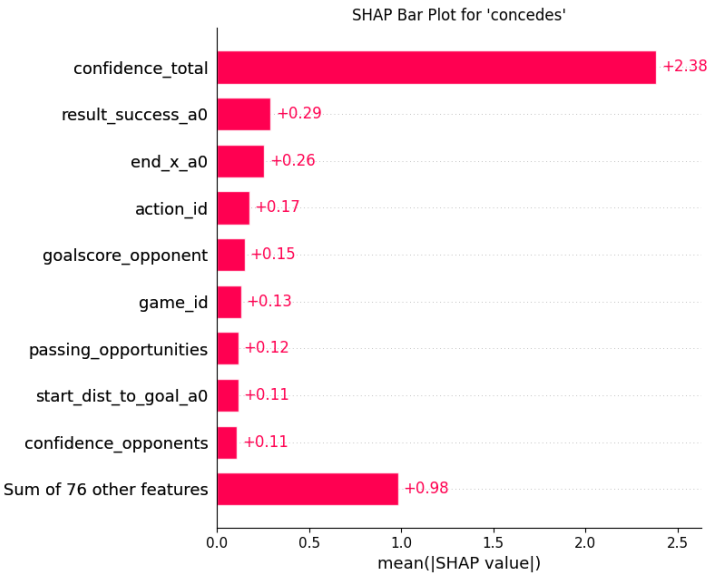


Figure 23: SHAP values (mean) – Concedes with complex features

This suggests that the model now has access to a more tactical representation of the situation and can better identify vulnerable defensive configurations or situations of imbalance.

Additional analysis of the target *scores*. While the overall structure of the SHAP values for the target *scores* remains dominated by geometric variables such as distance to goal, we also note the appearance of contextual variables such as the number of defenders in front or the opponent's density. Figure 24 illustrates this slight shift.

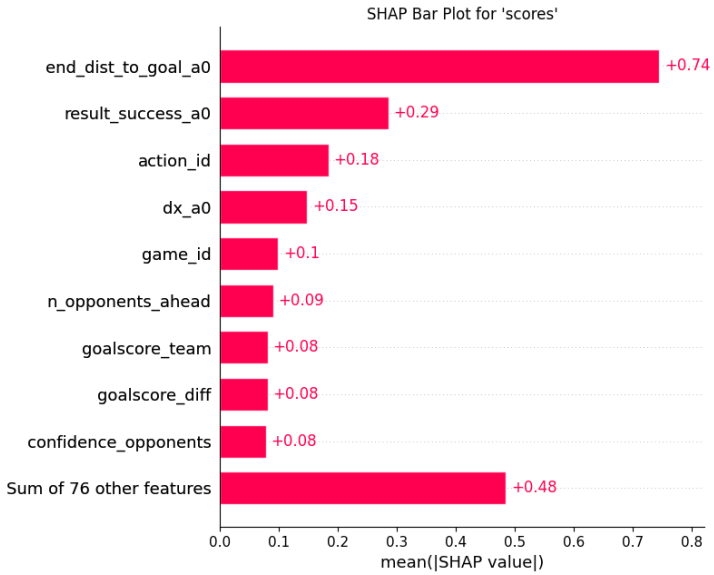


Figure 24: SHAP values (mean) – Scores with complex features

Discussion. This analysis shows that adding complex features does not fundamentally transform the dominant predictors, but adds an additional layer of contextual interpretability. Their contribution is most evident on the *concedes* target, which relies more on collective and tactical dynamics. Furthermore, the emergence of variables related to defensive density or passing opportunities shows that the model learns to recognise more structured patterns in the course of the game.

Finally, this explainability approach provides a better understanding of the model's limitations and strengths by revealing not only what it predicts, but also how it does so.

Qualitative analysis of valued players

Analysing the most highly valued players provides a better understanding of the qualitative impact of contextual feature enrichment. Two axes are explored: cumulative VAEP value, which reflects total contribution, and VAEP value per 90 minutes (with a threshold of 2,700 minutes), which identifies the most effective players with comparable playing time.

Cumulative VAEP. The top three in the ranking (Florian Wirtz, Álex Grimaldo and Granit Xhaka) are consistent in both versions of the model. All three play central roles at Bayer Leverkusen: Wirtz as a creative playmaker, Grimaldo as an ultra-offensive full-back, and Xhaka as a central playmaker heavily involved in building up play.

However, there are some notable changes after adding complex features. For example: - Odilon Kossounou, a central defender who is unremarkable in playmaking, drops out of the top 10; - Meanwhile, Josip Stanišić and Nathan Tella, full-backs or wingers with a high volume of forward runs, appear.

This suggests that the enriched model places more value on actions involving spatial awareness and strength comparisons, rather than isolated interventions without context.

Table 10: Top 10 cumulative VAEP – Without complex features

Player	VAEP	Offensive	Defensive
Florian Wirtz	15.20	15.48	-0.28
Granit Xhaka	12.40	11.85	0.55
Álex Grimaldo	11.83	12.83	-1.01
Jonathan Tah	8.81	8.58	0.23
Robert Andrich	8.18	7.74	0.44
Exequiel Palacios	6.73	6.89	-0.16
Edmond Tapsoba	6.40	6.09	0.31
Cody Gakpo	6.33	6.25	0.08
Odilon Kossounou	5.61	5.52	0.09
Jeremie Frimpong	5.57	6.00	-0.46

Table 11: Top 10 cumulative VAEP – With complex features

Player	VAEP	Offensive	Defensive
Florian Wirtz	15.33	15.73	-0.40
Álex Grimaldo	12.42	13.02	-0.60
Granit Xhaka	11.46	10.57	0.88
Robert Andrich	7.69	7.23	0.46
Jonathan Tah	7.57	6.91	0.67
Jeremie Frimpong	7.17	7.42	-0.25
Exequiel Palacios	6.15	6.66	-0.51
Cody Gakpo	6.00	6.02	-0.01
Josip Stanišić	5.55	5.43	0.13
Nathan Tella	5.55	5.50	0.05

VAEP per 90 minutes (2700 min). By filtering players who have played at least 2,700 minutes, the ranking highlights those with a high impact per action. The podium remains unchanged (Wirtz, Grimaldo, Xhaka), confirming their consistency.

It should be noted that: - Jordan Pickford, already present in the simple version, gains stability in the enhanced version: his profile as a goalkeeper focused on restarting play and long kicks seems to be better valued by the complex features. - Lucy Bronze, Alex Greenwood, etc., are no longer mentioned, in line with the exclusion of other corpora. - The most defensive players (Tah, Laporte) remain present, but are sometimes slightly downgraded in favour of players capable of contributing to creation or imbalance (Frimpong, Kimmich).

Table 12: Top 10 VAEP/90 minutes – With simple features (>2700 min)

Player	VAEP/90	VAEP	Minutes
Florian Wirtz	0.486491	15.20	2812
Álex Grimaldo	0.343974	11.83	3095
Granit Xhaka	0.156532	12.40	7128
Jonathan Tah	0.136191	8.81	5820
Joshua Kimmich	0.130067	4.42	3058
Xherdan Shaqiri	0.111834	4.02	3238
Ona Batlle	0.109663	3.42	2807
Aymeric Laporte	0.104646	5.13	4415
Alex Greenwood	0.090369	3.01	3000
Jordan Pickford	0.087408	2.82	2907

Table 13: Top 10 VAEP/90 minutes – With complex features (>2700 min)

Player	VAEP/90	VAEP	Minutes
Florian Wirtz	0.490779	15.33	2812
Álex Grimaldo	0.361209	12.42	3095
Granit Xhaka	0.144642	11.46	7128
Jonathan Tah	0.117119	7.57	5820
Joshua Kimmich	0.113958	3.87	3058
Jordan Pickford	0.107732	3.48	2907
Xherdan Shaqiri	0.102664	3.69	3238
Ona Batlle	0.100611	3.14	2807
Aymeric Laporte	0.095581	4.69	4415
Lucy Bronze	0.080475	3.27	3652

Conclusion. The addition of complex features does not disrupt the established leaders, but clearly changes the secondary valuations: the most linear or defensive profiles without active contribution lose ground to players who are more involved in the team's dynamics (restarts, projections, contextual pressure). The model thus becomes more sensitive to actual tactical interactions, rather than simply the geometric characteristics of actions.

4.5 Global

Overall assessment. When we take a step back to observe the results produced by the entire pipeline, one thing becomes clear: the individual performance of the modules does not guarantee the final quality of the reconstructions, far from it. Each brick seems to work relatively consistently when tested in isolation, but the sequence of these modules introduces an accumulation of errors — sometimes minor locally, but which ultimately compromise overall stability.

This gradual drift can be explained by several factors. First, there is no strong anchoring within the pipeline: each module works on predictions from the previous one, without any real feedback or correction mechanism. Second, the nature of the errors changes as

we progress: the first ones are often localised (incorrect bbox, misidentified player), while those observed at the output of the pipeline are structural (players off the field, inconsistent trajectories, loss of identity).

This accumulation makes it difficult to exploit the reconstructed sequences for tactical purposes. Certain phases of play are captured well — particularly in central areas with little occlusion — but it is still too risky to base a complete analysis on these reconstructions, as aberrant cases can occur suddenly.

That said, some partial results remain encouraging. This is particularly the case for the ball detection module, whose limitations were highlighted in section **??**: despite dedicated fine-tuning, the model is not always able to correctly identify the ball in all frames — especially when it is hidden, moving quickly, or confused with other white objects.

However, in the very specific context of the start of the action — a key moment for future exploitation — the constraints are significantly different. The ball is often on the ground, moving less quickly, and contrasts well with the background of the pitch, which maximises the chances of detection. Based on this intuition, we empirically evaluated the module's ability to provide a ball position in the immediate vicinity of the action start frame, as detected by the spotting module. Allowing a tolerance of ± 5 frames around this starting point, we observed that a ball position was available in more than 80% of cases.

It is obviously difficult to assess the exact accuracy of these positions (particularly due to possible calibration errors), but the mere fact that a plausible location is present at the right time is a positive sign. This tends to confirm that, despite its overall weaknesses, the module can provide useful information in contexts where it is most critical.

Ultimately, this thesis shows that a modular approach can be used to build a functional and intelligible pipeline, but that its reliability depends heavily on the interactions between modules. A logical next step would be to introduce global consistency mechanisms: plausibility checks, temporal regularisation, or even lightly annotated supervision at certain key points. This work opens up concrete prospects in this direction.

It is important to conclude this analysis with a clear warning: in its current state, the junction between the spotting and tracking modules does not allow for the reliable use of reconstructed sequences as the basis for a valuation model. The accumulated errors—whether absurd detections, temporal breaks, or inconsistent reconstructions—make any quantitative exploitation risky.

A valuation model relies on strong assumptions: the segmentation of actions must be clear, the sequences must be consistent and continuously readable, and the position of the ball and the players must be credible at all times. However, these conditions are too often violated here. The numerical values produced from such data then lose all meaning and risk introducing more noise than information.

In other words, until the spotting + tracking pipeline is consolidated—through correction mechanisms, overall consistency, or intermediate annotations—it is best not to attempt to

extract performance metrics through a valuation model. This observation, while somewhat frustrating, at least clarifies the priorities for future iterations.

5 Conclusion

The aim of this thesis was to build a modular pipeline capable of automatically analysing football matches from simple videos captured from a side view. Using existing building blocks (spotting, tracking, calibration, OCR, etc.), the objective was to test the feasibility of completely reconstructing the state of play, up to and including the evaluation of actions.

The work carried out shows that this approach is technically possible, but that its implementation raises a number of structural challenges. Each module, taken in isolation, can provide convincing results in simple cases. However, the sequence of these modules introduces an accumulation of errors that are difficult to control — calibration errors, noisy detections, identity losses — which compromise the reliability of the overall reconstruction. As it stands, it is not possible to use the outputs of this pipeline directly as input for a valuation model, as there are still numerous temporal and spatial inconsistencies.

However, these limitations should not obscure the relevance of the approach. The modular approach remains valid, if only for the clarity it brings to the analysis of weaknesses. It allows for testing localised adjustments, introducing regularisation mechanisms, and above all capitalising on rapid advances in research. Current SOTA models, although impressive, are not yet robust enough to be put directly into production on videos from non-professional equipment. However, their rate of improvement—as evidenced by successive increases in the GS-HOTA score on public leaderboards—suggests credible prospects in the short to medium term.

In this context, it remains perfectly legitimate to focus on gradual performance improvements, which could soon make it feasible to use similar pipelines for automatic action recognition, even from modest captures made on consumer equipment. This thesis does not claim to provide a complete solution, but rather to document the steps, obstacles and technical choices encountered in the construction of such a system. In short, it is a way of better understanding what already works — and what remains to be consolidated before considering a real-world application.

References

[1] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A Zuluaga. Usad: Unsupervised anomaly detection on multivariate time series. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3395–3404, 2020.

[2] Bavesh Balaji, Jerrin Bright, Harish Prakash, Yuhao Chen, David A Clausi, and John Zelek. Jersey number recognition using keyframe identification from low-resolution broadcast videos, 2023.

[3] Darwin Bautista and Rowel Atienza. Scene text recognition with permuted autoregressive sequence models. In *European Conference on Computer Vision*, pages 178–196, Cham, 10 2022. Springer Nature Switzerland.

[4] Mengqi Cao, Min Yang, Guozhen Zhang, Xiaotian Li, Yilu Wu, Gangshan Wu, and Limin Wang. Spotformer: A transformer-based framework for precise soccer action spotting. In *2022 IEEE 24th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6, 2022.

[5] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794. ACM, August 2016.

[6] Anthony Cioppa, Adrien Deliège, Silvio Giancola, Bernard Ghanem, Marc Van Droogenbroeck, Rikke Gade, and Thomas B. Moeslund. A context-aware loss function for action spotting in soccer videos, 2020.

[7] Anthony Cioppa, Silvio Giancola, Adrien Deliege, Le Kang, Xin Zhou, Zhiyu Cheng, Bernard Ghanem, and Marc Van Droogenbroeck. Soccernet-tracking: Multiple object tracking dataset and benchmark in soccer videos. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, page 3490–3501. IEEE, June 2022.

[8] Tom Decroos, Lotte Bransen, Jan Van Haaren, and Jesse Davis. Actions speak louder than goals: Valuing player actions in soccer. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1851–1861, 2019.

[9] Adrien Deliège, Anthony Cioppa, Silvio Giancola, Meisam J. Seikavandi, Jacob V. Dueholm, Kamal Nasrollahi, Bernard Ghanem, Thomas B. Moeslund, and Marc Van Droogenbroeck. Soccernet-v2: A dataset and benchmarks for holistic understanding of broadcast soccer videos, 2021.

[10] Julien Denize, Mykola Liashuha, Jaonary Rabarisoa, Astrid Orcesi, and Romain Hérault. Comedian: Self-supervised learning and knowledge distillation for action spotting using transformers, 2023.

[11] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–338, 2010.

[12] Yazan Abu Farha and Juergen Gall. Ms-tcn: Multi-stage temporal convolutional network for action segmentation, 2019.

[13] Alexander Geiger, Dongyu Liu, Sarah Alnegheimish, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Tadgan: Time series anomaly detection using generative adversarial networks, 2020.

[14] Silvio Giancola, Mohieddine Amine, Tarek Dghaily, and Bernard Ghanem. Soccernet: A scalable dataset for action spotting in soccer videos. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, page 1792–179210. IEEE, June 2018.

[15] Silvio Giancola, Anthony Cioppa, Bernard Ghanem, and Marc Van Droogenbroeck. Deep learning for action spotting in association football videos, 2024.

[16] Vladimir Golovkin, Nikolay Nemtsev, Vasyl Shandyba, Oleg Udin, Nikita Kasatkin, Pavel Kononov, Anton Afanasiev, Sergey Ulasen, and Andrei Boiarov. From broadcast to minimap: Achieving state-of-the-art soccernet game state reconstruction, 2025.

[17] James Hong, Haotian Zhang, Michaël Gharbi, Matthew Fisher, and Kayvon Fatahalian. Spotting temporally precise, fine-grained events in video, 2022.

[18] Glenn Jocher, Jing Qiu, and Ayush Chaurasia. Ultralytics YOLO, January 2023.

[19] Victor Joos, Vladimir Somers, and Baptiste Standaert. TrackLab. https://github.com/TrackingLaboratory/tracklab, 2024.

[20] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

[21] Maria Koshkina and James H. Elder. A general framework for jersey number recognition in sports video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 3235–3244, June 2024.

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13*, pages 740–755. Springer, 2014.

[23] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *CoRR*, abs/1705.07874, 2017.

[24] Adrien Maglo, Astrid Orcesi, Julien Denize, and Quoc Cuong Pham. Individual locating of soccer players from a single moving view. *Sensors*, 23(18):7938, 2023.

[25] Amir M. Mansourian, Vladimir Somers, Christophe De Vleeschouwer, and Shohreh Kasaei. Multi-task learning for joint re-identification, team affiliation, and role classification for sports visual tracking. In *Proceedings of the 6th International Workshop on Multimedia Content Analysis in Sports*, MM '23, page 103–112. ACM, October 2023.

[26] Amir M. Mansourian, Vladimir Somers, Christophe De Vleeschouwer, and Shohreh Kasaei. Multi-task learning for joint re-identification, team affiliation, and role classification for sports visual tracking. page 103–112. ACM, October 2023.

[27] F Ozge Unel, Burak O Ozkalayci, and Cevahir Cigla. The power of tiling for small object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 0–0, 2019.

[28] Harish Prakash, Jia Cheng Shang, Ken M. Nsiempba, Yuhao Chen, David A. Clausi, and John S. Zelek. Multi player tracking in ice hockey with homographic projections, 2024.

[29] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[30] Sanchayan Santra, Vishal Chudasama, Pankaj Wasnik, and Vineeth N. Balasubramanian. Precise event spotting in sports videos: Solving long-range dependency and class imbalance, 2025.

[31] João V. B. Soares and Avijit Shah. Action spotting using dense detection anchors revisited: Submission to the soccernet challenge 2022, 2022.

[32] Vladimir Somers, Victor Joos, Silvio Giancola, Anthony Cioppa, Seyed Abolfazl Ghasemzadeh, Floriane Magera, Baptiste Standaert, Amir Mohammad Mansourian, Xin Zhou, Shohreh Kasaei, Bernard Ghanem, Alexandre Alahi, Marc Van Droogenbroeck, and Christophe De Vleeschouwer. SoccerNet game state reconstruction: End-to-end athlete tracking and identification on a minimap. June 2024.

[33] StatsBomb. Introducing on-ball value (obv), 2021.

[34] StatsBomb. What is expected threat (xt)? possession value models explained, 2021.

[35] Jonas Theiner and Ralph Ewerth. Tvcalib: Camera calibration for sports field registration in soccer, 2022.

[36] Jonas Theiner, Wolfgang Gritz, Eric Müller-Budack, Robert Rein, Daniel Memmert, and Ralph Ewerth. Extraction of positional player data from broadcast soccer videos, 2021.

[37] Ultralytics. Ultralytics yolov8. https://yolov8.com, 2023. Accessed: 2025-03-24.

[38] Ultralytics. Modèles yolov11 - documentation ultralytics. https://docs.ultralytics.com/fr/models/yolo11/, 2024. Accessed: 2025-03-24.

[39] Artur Xarles, Sergio Escalera, Thomas B Moeslund, and Albert Clapés. T-deed: Temporal-discriminability enhancer encoder-decoder for precise event spotting in sports videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3410–3419, 2024.

[40] Artur Xarles, Sergio Escalera, Thomas B. Moeslund, and Albert Clapés. Astra: An action spotting transformer for soccer videos, 2024.

[41] Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Anomaly transformer: Time series anomaly detection with association discrepancy, 2022.