ETSY: A rule-based approach to Event and Tracking data SYnchronization

Maaike Van Roy, Lorenzo Cascioli and Jesse Davis

Department of Computer Science, KU Leuven & Leuven.AI, Leuven, Belgium
{firstname.lastname}@kuleuven.be

Abstract. Event data, which records high-level semantic events (e.g., passes), and tracking data, which records positional information for all players, are the two main types of advanced data streams used for analyses in soccer. While both streams when analyzed separately can yield relevant insights, combining them allows us to capture the entirety of the game. However, doing so is complicated by the fact that the two data streams are often not synchronized with each other. That is, the timestamp associated with an event in the event data does not correspond to the analogous frame in the tracking data. Thus, a key problem is to align these sources. However, few papers explicitly describe approaches for doing so. In this paper, we propose a rule-based approach `ETSY` for synchronizing event and tracking data, evaluate it, and compare experimentally and conceptually with the few state-of-the-art approaches available.

1 Introduction

Over the past years, there has been a dramatic increase in the amount of in-game data being collected about soccer matches. Until a few years ago, clubs that were using data mostly only had access to event data that describes all on-the-ball actions but does not include any information about what is happening off the ball. Hence, the configurations and movements of players, both during actions and in between, are missing. For example, it is impossible to distinguish between a pass from midfield with 5 defenders vs. 1 defender in front of the ball. Nonetheless, event data on its own can help address crucial tasks such as valuing on-the-ball actions [1,6,7,8,9,10,13] and assessing in-game decisions [4,5,11,12].

More recently, top-level clubs have installed in-stadium optical tracking systems that are able to record the locations of all players and the ball multiple times per second. Thus, tracking data provides the necessary context that is missing in event data. However, it is not straightforward to perform tactical and technical analyses solely based on tracking data as it does not contain information about events such as passes, carries, and shots which are crucial to make sense of a match since they contain semantic information.

Consequently, the richest analyses require integrating both data streams. Unfortunately, these streams are often not time synchronized. That is, the event at a specific timestamp does not correspond to the tracking frame with the same timestamp. Two main factors can create such misalignment. First, the clocks

used in event and tracking data collection might start at slightly different times, introducing a constant bias. Second, because event data are manually annotated, the timestamp of an event can occasionally be inaccurate due to human reaction time or mistakes. Unfortunately, there are few approaches described in the literature for synchronizing these data types and the relative merits of existing approaches are unclear. In this paper, we propose a rule-based approach for accomplishing this. Then, this paper attempts to answer the following questions:

Q1 What are the current state-of-the-art synchronization approaches?
Q2 Is a simple rule-based approach sufficient to synchronize event and tracking data, or is a more complex approach needed?
Q3 What are the advantages and disadvantages of each approach?

Additionally, we provide a publicly available implementation of `ETSY`.[1]

2 Problem Statement and Existing Approaches

Formally, the task of synchronizing event and tracking data from the same game is defined as follows:

Given:

1. Event data of a game, which contains for each on-the-ball action its location on the pitch (i.e., the x and y coordinate), the type (e.g., pass, shot, interception), the time of the action, the result, the body part used, the player that performed the action and the team he plays for.
2. Tracking data of the same game, which typically contains 10 or 25 frames per second (FPS), and includes in each frame all x,y-coordinates of all players and the x,y,z-coordinates of the ball at that moment in time.

Do: Assign each event a matching tracking frame such that it corresponds to the match situation at the moment of the event (i.e., the ball is at the same location in the event and tracking data, the player that performs the action in the event data is the same player that is in possession of the ball in the frame, and this player performs the same action as recorded in the event data).

There are few publicly described synchronization approaches. The approach by Anzer & Bauer [2,3] uses two steps to synchronize the data streams. First, it matches the kickoff event with its analogous tracking frame and computes the offset in time between them. It then uses this offset to shift all timestamps in the tracking data to remove the constant bias. To match the kickoff event with its analogous tracking frame, they use the movement of the ball to identify the kickoff frame within the first frames of the tracking data when the game has started. Second, for each event, it determines windows of possession where the player is within two meters of the ball and uses a weighted sum of features (e.g.,

[1] https://github.com/ML-KULeuven/ETSY

time difference between the event and the tracking frame, distance between ball and player, distance between ball coordinates in the event and tracking data) to determine the best frame. Grid search on a manually labeled test set is used to optimize the weights. Currently, their approach has only been evaluated on the most relevant actions, passes and shots, and yields satisfactory results for both.

Alternatively, `sync.soccer` is a bio-informatics-inspired approach by Clark & Kwiatkowski.[2] Their method is based upon the Needleman-Wunsch algorithm (similar to Dynamic Time Warping) that can synchronize any two sequences of data. In a nutshell, this approach compares every event with every tracking frame and uses a self-defined scoring function (i.e., a weighted combination of time difference between the event and the tracking frame, distance between ball and player, distance between ball coordinates in the event and tracking data, and whether the ball is in play) to measure the fit between them. Next, it constructs the synchronized sequence with the best overall fit. This approach is more general than that of Anzer & Bauer as it allows to synchronize any event type instead of only passes and shots, and without the need for manually labeled examples. However, the weights of the scoring function need to be tuned separately for each game, which is not straightforward and time intensive.

3 ETSY

We now outline our rule-based synchronization approach `ETSY`. To represent the event data, we use SPADL [6] which is a unified format to represent all on-the-ball actions in soccer matches. Therefore, any event stream that can be transformed to SPADL can be used. Next, we outline the necessary preprocessing steps to prepare the data and describe our algorithm.

3.1 Preprocessing

Before aligning the two data streams, four preprocessing steps are performed.

1. Event and tracking data often use different coordinate systems to represent the pitch, with event data often using the intervals $[0, 100] \times [0, 100]$ and tracking data the IFAB coordinate system (i.e., $[0, 105] \times [0, 68]$). Therefore, the event coordinates are transformed to match the coordinate system of the tracking data.[3]
2. The event data of each team is transformed to match the playing direction of the tracking data.
3. Only the tracking frames in which the game is not officially paused are kept. This removes e.g., frames before the start, VAR checks, and pauses due to injury. This ensures that events are matched with open-play frames only.
4. The velocity and acceleration of the ball are computed and added to each tracking frame.

[2] https://github.com/huffyhenry/sync.soccer

[3] The following package provides such a transformation: https://mplsoccer.readthedocs.io/en/latest/gallery/pitch_plots/plot_standardize.html.

3.2 Synchronization

The event and tracking data of each game are divided into the first and second period. The synchronization is run for each period separately. Our rule-based algorithm consists of the following two steps:

Step 1. Synchronize kickoff. Similar to Anzer & Bauer, we remove the constant bias between the timestamps in the event and tracking data by aligning the kickoffs. We determine which frame best matches the kickoff event by identifying, in the first five seconds of the game, the frame in which the ball is within two meters from the acting player and where the acceleration of the ball is the highest. The tracking data timestamps are then corrected for this offset.

Step 2. Synchronize remaining events. We synchronize all remaining SPADL events except "non_action", which is not an action, and "dribbles", which are imputed in the SPADL conversion but not present in the original data. Algorithm 1.1 summarizes the two steps needed: (1) identify the qualifying window of frames, and (2) score each frame in this window to find the best one.

```
For each action:
    window = get_window_of_frames_around(action, t_a)
    frame, score = get_matching_frame(action, window)
```

Algorithm 1.1. Core synchronization approach of `ETSY`.

`get_window_of_frames_around(action,` t_a`)` identifies a qualifying window of frames in which the matching frame is most likely to be found. It retrieves all tracking frames within a time window of $2 * t_a$ seconds around the tracking frame with the same (adjusted) timestamp as the event's timestamp.

`get_matching_frame(action, window)` assigns a score to each frame in the window based on how well it matches the action. First, it filters out all frames within the window that cannot be considered due to general and event-specific consistency rules. As a general rule, the timestamp of the frame should be later than the last matched frame. Other filters are action specific and we provide an overview in Section 3.3. Second, it scores each remaining frame using the (unweighted) sum of three linear functions with the same range of output values. These functions are (1) a function that maps the distance between the ball and the acting player in the tracking frame to a score $\in [0, 33.33]$, (2) a function that maps the distance between the acting player's location in the event data and the tracking data to a score $\in [0, 33.33]$, and (3) a function that maps the distance between the ball's location in the event data and the tracking data to a score $\in [0, 33.33]$. This yields a minimum score of 0 and a maximum score of 100. The frame with the highest total score is returned as the best matching frame.

3.3 Action-Specific Filters

Table 1 provides a summary of the employed action-specific filters. First, we group all SPADL actions into five categories, depending on their semantics:

– “Set-piece" denotes all possible actions performed during a set-piece.
– “Pass-like in open-play" denotes actions performed during open-play that move the ball away from the acting player.
– “Incoming" denotes actions where the acting player is receiving the ball.
– “Bad touch" simply denotes a bad touch.
– “Fault-like" denotes either a foul or tackle.

Next, we define for each category the time window parameter t_a. For all categories but set-pieces, we choose t_a equal to five seconds. Set-pieces typically require some set-up time and their annotated timestamps might be more off with respect to other actions. Hence, we extend t_a to 10 seconds to increase the chances that the matching frame is contained in the window.

Finally, we define the action-specific filters. We want the ball to be sufficiently close to the acting player. Thus, we enforce a maximum threshold on the distance between the two, allowing for some measurement error in the data. The distance for bad touches is set a bit larger as those typically already move the ball further away from the player. Similarly, fault-like actions do not need to be in close proximity to the ball. Filters on ball height aim at excluding frames where the ball height is not coherent with the body part used to perform the action. When the ball is up in the air, players typically do not use their feet to kick it; vice versa, a low ball is rarely hit with the head. Actions performed with hands (e.g., throw-ins, keeper saves) can happen at higher heights, as the players can jump and use their arms. In some cases, we add an extra limitation on whether the ball should be accelerating or decelerating. The ball should clearly be accelerating when a set-piece or pass-like action is performed with a player’s feet. Similarly, the ball should be decelerating when the acting player is receiving the ball.

Table 1. Action-specific filters and time window parameters.

Type	**Actions**	$\mathbf{t_a}$	**Distance**	**Height**	**Extra**
Set-piece	throw_in, freekick, corner, goalkick, penalty	10s	$\leq$ 2.5m	$\leq$ 1.5m (with foot), $\leq$ 3m (other)	accelerate (with foot)
Pass-like in open-play	pass, cross, shot, take_on, clearance, keeper_punch	5s	$\leq$ 2.5m	$\leq$ 1.5m (with foot), $\geq$ 1m (with head), $\leq$ 3m (other)	accelerate (with foot)
Incoming	interception, keeper_save, keeper_claim, keeper_pickup	5s	$\leq$ 2m	$\leq$ 3m	decelerate
Bad touch	bad_touch	5s	$\leq$ 3m	$\leq$ 1.5m (with foot), $\geq$ 1m (with head), $\leq$ 3m (other)	/
Fault-like	foul, tackle	5s	$\leq$ 3m	$\leq$ 4m	/

4 Evaluation

We evaluate our approach on one season of a European first-division league. After removing games where one data source had too many errors (e.g., ball location was not recorded), we have 313 games. We compare the proposed approach to the existing approaches, both experimentally and conceptually. The approach of Anzer & Bauer is left out of the experimental comparison, as we do not have access to matching video footage nor experts to label a part of our data set.

4.1 Experimental Setup

It must be noted that the original implementation of `sync.soccer` is written in Haskell and only accepts StatsPerform's F24 event feeds and ChyronHego's Tracab files. As these formats are different from the data we have available, we have created a Python implementation of the same algorithm that can use our data set and compare against our implementation.

The original `sync.soccer` implementation does not remove the constant bias between the event and tracking data timestamps. Therefore, this method relies entirely on the weights of the score function, which need to be tuned separately for each game. This is not straightforward and requires one to trade off the time difference between the events and tracking frames with the different distance metrics used. Including the kickoff synchronization step outlined in our approach to remove the constant time-bias mitigates this problem and improves the quality of the synchronization. In the comparison, we have included both `sync.soccer` approaches and used equal weights for all parts of the score function.

4.2 Experimental Comparison

It is not straightforward to measure the quality of the resulting synchronization as we do not have access to ground truth labels. Even when one has access to video footage of the game to validate with, it is unlikely that the video's timestamps will perfectly align with the extracted tracking data. Consequently, it is difficult to link the tracking frames to the snapshots in the video and determine the exact moment of an event. Thus, it is hard to report an accuracy-like metric. Therefore, we propose to look at two alternative metrics: the coverage (i.e., the percent of events for which a matching frame can be found), and the agreement with `ETSY` within s seconds (i.e., the percent of events for which the found frame is within a range of s seconds from the one identified by `ETSY`). Additionally, we compare the runtime of the different approaches.

Coverage and runtime Table 2 summarizes the coverage and runtime of `ETSY`, `sync.soccer` and a naive timestamp-based approach, both with and without the time-bias adjustment using the kickoff alignment step.

By construction, `sync.soccer` pairs every event with a tracking frame and hence achieves a coverage of 100%, regardless of how bad the resulting match is.

Table 2. Experimental comparison of `ETSY`, `sync.soccer`, and a timestamp-based approach on runtime and coverage. A $*$ indicates the time-bias adjusted version.

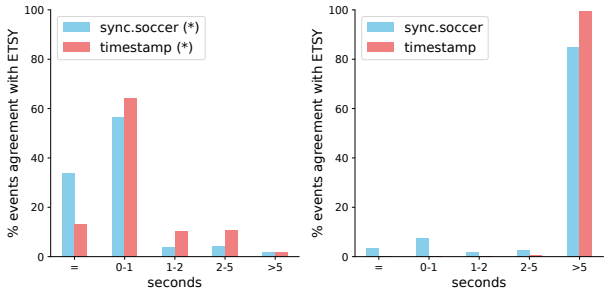
Approach	Average time / game	Coverage
`ETSY`	2.7min ($\pm$ 17s)	91.61 %
timestamp	21s ($\pm$ 2s)	95.90%
timestamp ($*$)	21s ($\pm$ 2s)	99.83%
`sync.soccer`	8.6min ($\pm$ 38s)	100%
`sync.soccer` ($*$)	8.6min ($\pm$ 38s)	100%

The timestamp-based approach finds a matching frame for an event as long as a corresponding timestamp exists among the tracking data. When misalignment due to clock bias is present, some events at the beginning or end of a period might not be covered because the tracking frames have either not started yet or are already finished. In contrast, when no suitable frame can be found according to the rule base, `ETSY` does not return a match. Over all considered events, `ETSY` yields a coverage of 91.61%. We perform an analysis on `ETSY`'s missed events in Section 4.3. Additionally, the score given to each match by our algorithm can be used as a threshold during a subsequent analysis process. For example, one could retrieve all passes with a matched frame and a score $> 95\%$ (i.e., the method is quite certain that this is the exact frame that matches the event) to ensure only perfectly synchronized data is used in the analysis.

Naturally, using a timestamp-based approach is the fastest as this involves only a retrieval of the frames with the matching timestamps. Compared to our implementation of `sync.soccer`, `ETSY` performs its synchronization roughly six minutes per game faster. The runtime of our `sync.soccer` implementation is consistent with that reported by Clark & Kwiatkowski. Note that including the time-bias adjustment step only incurs a negligible amount of extra runtime.

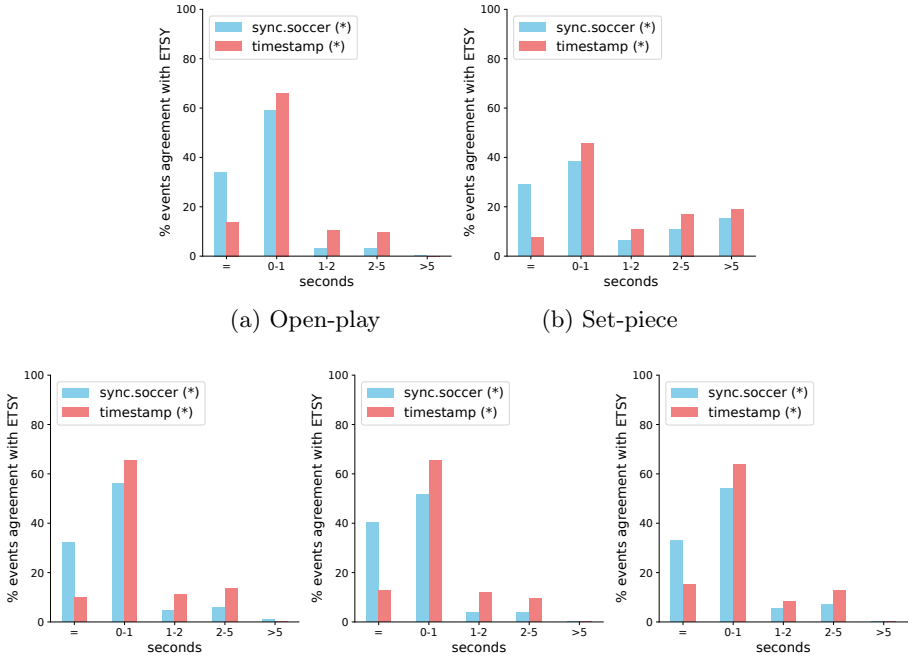
Agreement Figure 1 shows the agreement with `ETSY` for both `sync.soccer` and timestamp-based approaches. Without time-bias adjustment, most of the found frames lie more than five seconds apart and hence no longer correspond to the same match context. In contrast, the time-bias adjusted versions perform much better. Most of the frames lie within one second of the frame found by `ETSY`. This indicates the need for a time-bias adjustment in the existing approaches.

Additionally, we inspect the agreement for each of the defined action categories in Figure 2. This gives us an indication of which actions are easier to match than others. We only compare with the time-bias adjusted versions as the kickoff alignment step was shown to be necessary. For bad touches, open-play, incoming, and fault-like actions, the distributions are quite similar. The majority of the found frames are within one second from the frame identified by `ETSY`, indicating a rather strong agreement. However, for set-pieces, we see an increase in the number of events with a frame that is further off from the one identified by `ETSY`. This could possibly indicate that the fixed time window used for set-pieces is not ideal and might need adjusting in the future.



(a) Time-bias adjusted (b) Time-bias not adjusted

Fig. 1. Agreement with `ETSY` for (a) the time-bias adjusted and (b) the non-adjusted versions of `sync.soccer` and the timestamp-based approach. All action types are included and the agreement is computed over five disjunct windows.



(a) Open-play (b) Set-piece

(c) Fault-like (d) Bad touch (e) Incoming

Fig. 2. Agreement with `ETSY` for the time-bias adjusted versions of `sync.soccer` and the timestamp-based approach. The agreement is calculated for each action category.

4.3 Missed Events Analysis

Next, we look at what happens in those cases where `ETSY` does not find a suitable match. The method does not find a matching frame when all the frames in the selected window are filtered out by the rules presented in Sections 3.2 and 3.3. Thus, we take the events where no matching frame is found and in turn drop one of the filters to verify if without it a matching frame would have been found. This allows us to analyze whether some of the imposed rules are largely responsible for the misses. Table 3 shows the results for each action category.

Note that the row values do not sum to 100%. As long as there is at least one frame left after filtering, a matching frame is identified by the algorithm. However, different filters might exclude different frames. Thus, it is possible (and likely), that multiple filters are concurrently responsible for not returning a match, and removing each filter could "free" a different frame in the window.

Table 3. `ETSY` missed events analysis. The percentages indicate how many of the unmatched events would have a matching frame if each filter was dropped.

Action type	Total misses	Distance	Height	Time	Acceleration
Set-piece	982	100%	1%	10%	8%
Pass-like in open-play	24783	91%	17%	42%	25%
Incoming	4718	98%	3%	41%	22%
Bad touch	309	89%	19%	58%	/
Fault-like	883	98%	1%	69%	/

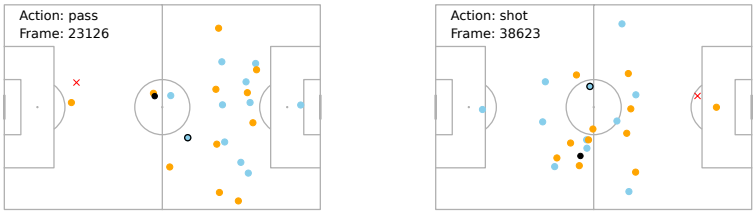
In most cases, removing the distance filter would yield a matching frame. However, the associated frame would probably be wrong, as the ball would be far away from the acting player. It is possible that these are cases where either a slight error exists in the tracked locations, thus stretching distances between player and ball, or in the annotated timestamps, meaning that the window of selected frames does not include the correct frame. While filters on ball height and acceleration show a minor influence, the filter on timestamps also has a consistent effect. Except for set-pieces, roughly half of the misses would be avoided if frames whose timestamp is earlier than the last matched frame would be retained. This happens when an earlier event is matched to a slightly incorrect frame that is further in the future. This error propagates and prevents synchronizing the next couple of actions. For example, this can occur when a number of actions happen very quickly after one another.

4.4 Example Synchronization

Next, we use an example to compare the different approaches. Figure 3 shows two random events that are synchronized according to all three approaches.

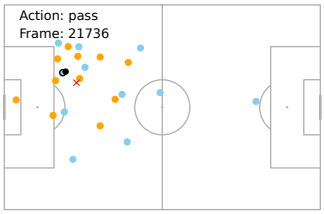
When using the timestamp-based approach, the identified frames do not match the situation described by the event data (i.e., the ball and/or acting player are not near the location indicated by the red cross). In contrast, both

ETSY and sync.soccer (time-bias adjusted) do find a correct matching frame, although not the exact same one. In both cases, ETSY identifies the frame previous to that of sync.soccer. Regardless of this slight difference, the match situation found, and thus the context added to the event, is still the same.

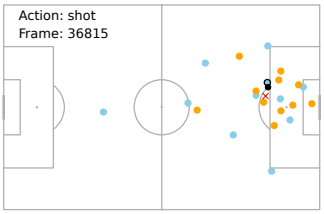


(a) Event 1 - timestamp

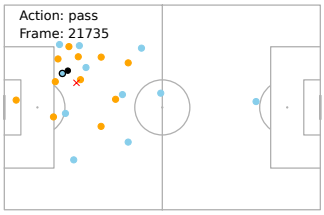
(b) Event 2 - timestamp



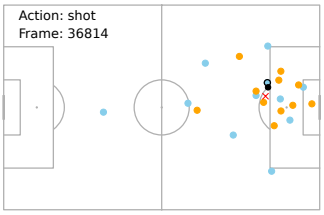
(c) Event 1 - sync.soccer



(d) Event 2 - sync.soccer



(e) Event 1 - ETSY



(f) Event 2 - ETSY

Fig. 3. Illustration of two random events synchronized by three approaches. The red cross indicates where the event takes place in the event data, the black encircled player is the acting player according to the event data. All player positions are shown according to the matched frame and the ball is shown in black.

4.5 Conceptual Comparison

Finally, we perform a conceptual comparison between all four approaches. A summary can be found in Table 4.

Table 4. Conceptual comparison of `ETSY`, `sync.soccer`, the timestamp-based approach, and Anzer & Bauer's approach. A – means a property is only partially present.

Approach	Automated	All actions	No extra data	Bias solved	Code
`ETSY`	X	X	X	X	X
timestamp	X	X	X	–	
`sync.soccer`	–	X	X	–	X
Anzer & Bauer	–			X	

Both `sync.soccer` and `ETSY` are a general open-source approach to synchronize *all actions* in event data with their matching tracking frame. In contrast, the approach by Anzer & Bauer has so far only been applied to and proven to work for passes and shots. Additionally, it requires an expert to manually label a set of training data and thus relies on more information than is available in the event and tracking data. As neither the video footage nor experts are always available, both `sync.soccer` (especially when augmented with a time-bias adjustment) and `ETSY` provide a more general approach to synchronize event and tracking data. The original `sync.soccer` approach is not entirely automated as it still requires one to fine-tune the weights of the scoring function for each game. This can be mitigated by adding a time-bias adjustment step, after which the approach produces an automated and satisfactory synchronization. Naturally, the timestamp-based approach is automated and can be applied to synchronize all actions. However, its performance is unacceptable.

5 Conclusion

This paper addresses the task of synchronizing soccer event data with tracking data of the same game, which is a problem that is not often explicitly mentioned in the literature. This paper provides an overview of the current state-of-the-art approaches, introduces a simple rule-based approach `ETSY`, and compares the approaches both experimentally and conceptually. In contrast to existing approaches, the simple rule-based approach performs a satisfactory synchronization for all action types while using less time and without manual intervention. We have publicly released `ETSY`'s implementation at https://github.com/ML-KULeuven/ETSY.

In the future, we aim to evaluate our method on other event and tracking data formats to assess the influence of data accuracy on our approach. The effect of different weights for the scoring function could be analyzed as well. Additionally, we would like to evaluate our method using video footage, and investigate improvements to e.g., mitigate the cascading misses problem.

Acknowledgements This work has received funding from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" program, the Research Foundation – Flanders under EOS No. 30992574, and the KU Leuven Research Fund (iBOF/21/075). We thank Jan Van Haaren for the useful feedback during the development of this work.

References

1. AmericanSoccerAnalysis: What are Goals Added. Available at https://www.americansocceranalysis.com/what-are-goals-added (2020)
2. Anzer, G., Bauer, P.: A Goal Scoring Probability Model for Shots Based on Synchronized Positional and Event Data in Football (Soccer). Frontiers in Sports and Active Living **3** (2021)
3. Anzer, G., Bauer, P.: Expected passes - Determining the difficulty of a pass in football (soccer) using spatio-temporal data. Data Mining and Knowledge Discovery **36**, 295–317 (2022)
4. Beal, R., Chalkiadakis, G., Norman, T.J., Ramchurn, S.D.: Optimising Game Tactics for Football. In: Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems. p. 141–149. AAMAS '20 (2020)
5. Beal, R., Changder, N., Norman, T., Ramchurn, S.: Learning the Value of Teamwork to Form Efficient Teams. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 7063–7070 (2020)
6. Decroos, T., Bransen, L., Van Haaren, J., Davis, J.: Actions Speak Louder than Goals: Valuing Player Actions in Soccer. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. p. 1851–1861. KDD '19 (2019)
7. Merhej, C., Beal, R.J., Matthews, T., Ramchurn, S.: What Happened Next? Using Deep Learning to Value Defensive Actions in Football Event-Data. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. p. 3394–3403. KDD '21 (2021)
8. Rudd, S.: A Framework for Tactical Analysis and Individual Offensive Production Assessment in Soccer Using Markov Chains. In: New England Symposium on Statistics in Sports (2011)
9. Singh, K.: Introducing Expected Threat. Available at https://karun.in/blog/expected-threat.html (2019)
10. StatsBomb: Introducing On-Ball Value (OBV). Available at https://statsbomb.com/articles/soccer/introducing-on-ball-value-obv/ (2021)
11. Van Roy, M., Robberechts, P., Yang, W.C., De Raedt, L., Davis, J.: Leaving Goals on the Pitch: Evaluating Decision Making in Soccer. In: Proceedings of the 15th MIT Sloan Sports Analytics Conference. pp. 1–25 (2021)
12. Van Roy, M., Robberechts, P., Yang, W.C., De Raedt, L., Davis, J.: A Markov Framework for Learning and Reasoning About Strategies in Professional Soccer. Journal Of Artificial Intelligence Research **77**, 517–562 (2023)
13. Yam, D.: Attacking Contributions: Markov Models for Football. Available at https://statsbomb.com/2019/02/attacking-contributions-markov-models-for-football/ (2019)