Event Pattern Matching in Football

Tariq El-Bahrawy
Carleton University
Ottawa, Canada
tariqelbahrawy@cmail.carleton.ca

Mohammad Bin Yousuf
Carleton University
Ottawa, Canada
mohammadyousuf@cmail.carleton.ca

Booshra Nazifa Mahmud
Carleton University
Ottawa, Canada
booshranazifamahmud@cmail.carleton.ca

ABSTRACT

The growing importance of data analysis in sports, particularly football, has provided opportunities to explore the complex dynamics of the game, offering insights into player behaviors, team strategies, and game patterns. However, the intricacies of football, characterized by the dynamic interaction of numerous events, present significant challenges for traditional pattern recognition techniques. In this paper, we introduce a novel approach to pattern matching in football event data, leveraging Dynamic Time Warping (DTW) as our trajectory similarity algorithm to capture the temporal and spatial aspects of football events. DTW, a well-established technique for time series analysis, enables the quantification of similarity between different sequences by allowing non-linear alignments. We hypothesize that DTW can effectively capture the spatiotemporal dynamics of event sequences, offering a robust measure of pattern similarity and facilitating a more nuanced understanding of game patterns. We detail our innovative approach, discuss the principles behind our application of DTW for pattern recognition in football, and present the results of our study, including visual examples of similar patterns. Furthermore, we discuss the potential impacts of our findings on game strategies and player performance evaluations. Our research aims to develop a powerful tool for identifying patterns in football, contributing to the advancement of sports analytics, and providing valuable insights for coaches, players, and analysts.

1 INTRODUCTION

The role of data analysis in sports, particularly in football, has grown tremendously in recent years. With detailed event data, we can explore the complex dynamics of the game, offering insights into player behaviors, team strategies, and game patterns. The core of this opportunity is the availability of detailed event data, which, when harnessed effectively, can unlock a myriad of game-changing insights. Such insights have the potential to revolutionize training methodologies, game strategies, and player performance evaluations. However, the complexity of football event data, especially spatiotemporal sequences of player actions and movements, poses significant challenges for traditional pattern recognition techniques.

The intricacies of football, marked by the dynamic interaction of numerous events, [15] present significant challenges for traditional pattern recognition techniques. Football, unlike many other sports, is free-flowing and involves complex, interrelated sequences of events happening in both space and time. The task of capturing these sequences and deriving meaningful patterns from them is non-trivial.

In this paper, we introduce a novel approach for pattern matching in football event data, leveraging Dynamic Time Warping (DTW) as our trajectory similarity algorithm to address this complexity calls for innovative methods that can capture the temporal and spatial aspects of football events. Originally developed for speech recognition, DTW has been applied across a variety of fields, including sports analytics, to compare and align sequences of varying lengths [16]. It is a well-established technique for time series analysis, and offers a unique way of quantifying similarity between different sequences by allowing non-linear alignments [19]. In the context of football, we hypothesize that DTW can effectively capture the spatiotemporal dynamics of event sequences, offering a robust measure of pattern similarity. This allows for a more nuanced understanding of game patterns, going beyond simple event counts or static spatial positions.

We detail our innovative approach to football event data analysis, discussing the principles behind our application of DTW for pattern recognition in football. Our approach begins by randomly selecting an input (reference) sequence of five event locations. The goal is to select an input sequence that starts from the goal keeper's box and ends at the opponents box and five events were enough for that sequence, we can use more or less events for the input sequence depending on the situation we want to analyze. Subsequently, we initiate an interrogation of our extensive event data repository to identify occurrences within a spherical region defined by a 5-yard radial distance from our initial input point. The outcome of this process results in the generation of a multiplicity of sequences, each encompassing five locational points. Then we perform DTW to discern the most similar sequences to our input sequence. This process is guided by the principle that the degree of similarity between sequences is inversely proportional to the Euclidean distance computed by the DTW algorithm between the input sequence and the sequences under consideration [13].

We also show the results of our study, including visual examples of similar patterns, and discuss how these could impact game plans and how we measure player performance. We aim to create a strong tool for finding patterns in football, helping to improve the field of sports analytics, and offering new insights for coaches, players, and experts. In this paper, Section 3 delves into the relevant literature, while Section 4 describes the dataset used. The methodologies are explored in Section 5, followed by results and discussion in Section 6. Finally, Section 7 rounds off with the conclusion and outlines potential avenues for future research.

2 OBJECTIVES

The primary goal of the project is to find the most similar sequences of event locations to an input sequence of event locations. The following objectives will help us achieve our goal:

- Creating a scalable database using MongoDB to store events from matches and their relevant information provided by StatsBomb.

- Writing queries and indexes that will help us find sequences of events efficiently.
- Using a trajectory similarity algorithm (DTW) to find the top-k similar sequences to the input sequence.
- Evaluating our results by visualizing the sequences found with the input sequence.

3 RELATED WORK

The increasing availability of event data in sports has spurred a surge of interest in advanced data analysis techniques. Over the past decade, the application of data analysis and machine learning techniques in sports has seen rapid growth, spurred by the availability of rich datasets [2]. Some researchers specifically work on building a collection of detailed soccer match data including performance analysis and identifying success factors [11]. Football, being a highly dynamic and complex sport, has attracted considerable interest from researchers aiming to extract meaningful insights from the data. The literature on football analytics spans a wide range of topics, encompassing various aspects of pattern recognition, player performance evaluation, and team strategy analysis.

An essential aspect of football analytics is the extraction and analysis of spatiotemporal patterns from event data. In the context of football, the concept of event data refers to detailed information about individual actions by players during a match, such as passes, shots, and tackles. However, understanding the patterns in these event sequences is a challenging task due to the spatiotemporal characteristics of football. Researchers have employed various techniques to capture these patterns, ranging from simple descriptive statistics to more complex machine-learning algorithms [14].

Among these approaches, clustering techniques have been commonly used to identify common patterns in player and team movements [6]. Recent advancements in sports analytics have led to the development of more sophisticated models for predicting game outcomes, such as the study by Lucey et al. (2015) [7], which employed strategic features from spatiotemporal data to improve shot prediction in soccer. Similarly, Bialkowski et al., (2014) [3] applied spatiotemporal pattern recognition to automatically detect game tactics from player trajectories. The research conducted by Grund (2012) in 'Tactical pattern recognition in soccer games using special self-organizing maps,' [5] used pattern recognition methods to identify strategic patterns in football, laying the groundwork for future research in the field. Despite these advancements, traditional pattern recognition techniques often struggle with the complexity of football event data, as they are not designed to handle the dynamic, continuous, and free-flowing nature of the game [4]. Recognizing this limitation, recent research has shifted towards more flexible and robust techniques.

Dynamic Time Warping (DTW), an algorithm originally developed for speech recognition, has emerged as a promising tool for this purpose. DTW allows for non-linear alignment of sequences, making it uniquely suited to compare sequences of varying lengths [10]. Recent studies have demonstrated the utility of DTW in sports analytics, particularly in comparing and aligning sequences of player actions and movements [22]. The application of DTW in football analytics is still a nascent field, with much potential for exploration. By offering a robust measure of pattern similarity, DTW provides a pathway to a more nuanced understanding of game patterns, beyond simple event counts or static spatial positions. The study outlined in this paper builds upon this existing body of research, proposing a novel approach to pattern matching in football event data using DTW.

4 DATASET

The dataset we are using for this project is sourced from StatsBomb [18]. The dataset consists of five main components including events, lineups, matches, three-sixty, and competitions. With a collection ranging from 2003 onwards, StatsBomb's open data consists of tracking data of most major league events collected and annotated. For our particular use case, we will not be using the three-sixty dataset. The competition data includes all relevant information on the various league competitions recorded and has expanded to include 90 global leagues by 2022. The matches include in-depth information on the setup of matches surprisingly even including the referee's country. The event data is the most relevant in our use case and will be heavily focused on throughout our project. Events data includes thorough play-by-play information on the movement of the ball throughout the match. It contains general attributes such as play_pattern, possessions, locations, and others for all matches. Additionally, it includes event-specific attributes that expand on each play and its relevant information. We have used around 4.7 million events from 1321 matches from different competitions including the English Premier League and the Spanish La Liga.

The events data is stored in JSON format, like all the other files. However, the events data includes nested data in some of the attributes. The data includes information on the time stamps of each event, player information involved in the event, location of the player and event, player events, and event play pattern. For each event, a Type attribute includes all the information that expands on the outcome/decisions of the play.

5 METHODOLOGY

This section discusses the methodology that was implemented to bring the vision to life. This section covers the libraries used, data import and storage methods, preprocessing steps, distance calculation, sequence sorting, and visualization.

5.1 Libraries

The following subsection describes the libraries used in this project. These libraries include Pandas, NumPy, Matplotlib, mplsoccer, pymongo, FastDTW, and Scipy. These libraries are essential for data manipulation, visualization, and analysis of football event data. Each library is briefly explained below.

5.1.1 Basic Libraries. Libraries such as Pandas [8], Numpy [20], and Matplotlib are crucial libraries for any Python project. In this project, we rely on Pandas and Numpy for the data structures and functions working with tabular data and multi-dimensional arrays and matrices. By using these libraries we are able to effectively manipulate the results of the queries to the MongoDB databases. Matplotlib, a visualization library, was used to create plots to visualize the sequences.

5.1.2 OS Libraries. The OS and subprocess libraries are used to interact with the operating system in Python and is used to start and stop the MongoDB databases. The BSON.SON is a module that is used to encode and decode BSON (Binary JSON) objects in Python. It is used to manipulate the results of the queries to MongoDB.

5.1.3 FastDTW. The fastdtw library is a Python implementation of the FastDTW algorithm, which is a variant of the Dynamic Time Warping (DTW) algorithm. The DTW algorithm is commonly used for sequence alignment, and it measures the similarity between two time series that may have different lengths and speed. The FastDTW algorithm is faster and requires less memory than the standard DTW algorithm, while still maintaining high accuracy [17]. It is used in this code to calculate distances between sequences of events.

5.1.4 SciPy. This project uses a sublibrary of SciPy [21] called the spatial distance. It provides functions for calculating distances between two points in n-dimensional space. The euclidean function from this library is used in conjunction with the fastdtw library to calculate distances between sequences of events.

5.1.5 PyMongo. A library that provides a Python interface for working with MongoDB, a popular NoSQL database [9]. It is used in this code to query a MongoDB database for sequences of events.

5.1.6 mplsoccer. A library that provides tools for creating soccer visualizations in Matplotlib [12]. It is used in this code to create soccer pitch visualizations in the plots of sequences of events.

5.2 Data Storage

As mentioned, the project works with over 4.7 million events from over 1300 matches. The number of events are ever increasing and as such we need to cater to expanding our database. In this project, we utilize MongoDB, a cross-platform document-oriented database program, that is classified as a NoSQL database program. Instead of using tables and rows as in traditional relational databases, MongoDB is comprised of collections and documents. Documents comprise sets of key-value pairs and are very fundamental in MongoDB. The collections contain sets of documents and is similar to tables in a relational database. MongoDB uses a document data model, which allows it to be more flexible and scalable compared to traditional relational databases [1].

MongoDB is a crucial component of this project as it serves as the database where all the event data is stored. With MongoDB, we can efficiently store and retrieve event data based on different criteria, such as location, match ID, and event type. This allows us to easily retrieve specific sequences of events that occur in a given location on the field. MongoDB's ability to handle unstructured data and its flexible document model make it an ideal choice for storing event data in our project.

5.3 Data Import

The fundamental dataset underpinning our research was sourced from the renowned StatsBomb open data repository [18]. This extensive and detailed repository serves as a prominent hub for football event data, encompassing matches from a diverse array of leagues and competitions across the globe. The data stored in this repository extends beyond mere scores and player statistics, delving into more granular event data that captures the intricate dynamics of football matches. These events include but are not limited to, player movements, passes, shots, and tackles, all of which provide rich and detailed insight into the unfolding game patterns and strategies.

To facilitate our efficient and targeted access to this comprehensive dataset, we utilized the PyMongo library [9]. PyMongo, a Python driver for MongoDB, offers an interface for interacting with MongoDB databases. By providing a comprehensive set of tools for interacting with MongoDB, PyMongo allowed us to execute complex queries on the dataset, sift through the vast amount of data, and extract the pertinent event sequences necessary for our analysis.

The MongoDB database, in which the StatsBomb event data is stored, provides a flexible and scalable data model that is particularly suited for handling complex and hierarchical data structures inherent in football event data. The structure of the MongoDB database allowed us to efficiently query the database and retrieve the relevant event data needed for our DTW pattern recognition analysis.

In essence, the combined use of the StatsBomb open data repository, the PyMongo library, and the MongoDB database provided us with a robust and flexible data infrastructure to support our innovative approach to football event data analysis.

5.4 Data Preprocessing

In the preprocessing stage, we developed a custom function that extracts sequences of events from the data. The function queries the MongoDB database for events that happened within a specified radius of a particular location on the field. The resulting sequences include the location of the event and the locations of the next four events that happened in the same match.These sequences of events are then used to calculate distances between different sequences.

5.5 Distance Calculation

In the process of measuring the distances between the sequences, our methodology involved the use of the fastdtw library, a Python package that effectively implements the dynamic time-warping (DTW) algorithm. This algorithm, originating from the field of speech recognition, has been recognized for its proficiency in aligning sequences, regardless of their respective lengths or variability. The underlying principle of DTW, which allows for non-linear alignments of sequences, has been instrumental in allowing us to effectively compare and measure the similarity between distinct sequences of events within our football data.

To compute the measure of similarity (or dissimilarity) between the sequences, we opted for the Euclidean distance as our chosen parameter for the FastDTW function. The Euclidean distance, often referred to as the "ordinary" straight-line distance between two points in a Euclidean space, provides a straightforward yet powerful means to quantify the degree of dissimilarity between two data sequences. At this stage of our analysis, we designated the first sequence in our sorted list as the 'reference sequence'. This sequence, having the smallest DTW distance from the input sequence,

Input: Input x-coordinate, Input y-coordinate, Radius in yards
Output: Sorted list of sequences
Connect to MongoDB server;
Select database "statsbomb";
Select collection "eventswithid";
Query documents within a radius of (input_x, input_y);
for *each document in query result* **do**
  if *current document is the first one* **then**
    Store the first sequence as the reference sequence;
    if *current document has "location" field* **then**
      Append the document's "location" to the reference sequence;
      Query the next four documents in a match and add their locations to the reference sequence;
      Append reference sequence to sequences list;
    end
  end
  else
    Calculate distance to reference sequence using FastDTW algorithm;
    if *current document has "location" field* **then**
      Append document's "location" to current sequence;
      Query the next four documents in the match and add their locations to the current sequence;
      Append current sequence to sequences list;
    end
  end
end
Calculate distances between the reference sequence and all other sequences;
Combine distances with sequences using zip;
Sort by distance and add reference sequence to the front of the list;
Plot the sorted sequences using matplotlib and mplsoccer;

Algorithm 1: Algorithm for finding and sorting sequences of events in a football field

served as our primary point of comparison for subsequent distance computations.

We then proceeded to calculate the Euclidean distances between this reference sequence and all other sequences in our list. This involved a pairwise comparison between the reference sequence and each subsequent sequence, where the DTW algorithm was applied to align the sequences and the Euclidean distance was computed to quantify the degree of similarity. This methodical process of distance computation and sequence alignment served as a critical component of our pattern recognition approach, enabling us to effectively rank the sequences based on their level of similarity to the input sequence. The results generated from this stage of analysis provided valuable insights into the patterns and dynamics underlying the football event data.

5.6 Sequence Sorting

The sequences were sorted based on their distances from the reference sequence. The reference sequence was placed at the beginning of the list, followed by the sequences with the lowest distances to the reference.

Sequence	Location	Distance
1 vs 2	(6.1, 26.1)	25.78
1 vs 3	(9.3, 31.7)	78.69
1 vs 4	(7.9, 31.0)	222.99
1 vs 5	(9.7, 31.9)	125.95
1 vs 6	(9.7, 31.9)	174.25
1 vs 7	(9.1, 31.0)	224.98
1 vs 8	(5.6, 26.0)	177.63
1 vs 9	(8.2, 28.0)	115.73
1 vs 10	(8.2, 28.0)	69.63
1 vs 11	(8.2, 27.3)	124.62
1 vs 12	(9.0, 28.6)	289.16
1 vs 13	(10.3, 29.9)	227.00
1 vs 14	(8.2, 31.6)	85.10
1 vs 15	(8.2, 31.6)	117.43
1 vs 16	(7.5, 32.0)	149.55
1 vs 17	(4.1, 32.9)	243.13
1 vs 18	(4.7, 31.6)	135.04
1 vs 19	(4.7, 31.6)	194.96
1 vs 20	(3.5, 32.3)	243.61
1 vs ..	(..., ...)	.....

Table 1: Distances between Sequence 1 and other sequences

5.7 Visualization

The sequences are then visualized on a football pitch using the mplsoccer library. This library offers functions for drawing pitches and visualizing football data. Each sequence is plotted as a set of points on the pitch, with arrows connecting each point to the next point in the sequence. This visualization allows us to compare and analyze different sequences of events and gain insights into the patterns and strategies used by different teams during matches.

5.8 System Overview

Below, we summarize the system overview as described in Algorithm 1.

First, the user decides on the radius of the search, and a query is created using the '$geoWithin' operator, which selects documents whose location field falls within a specified range area. The user also keys in the input sequence, which is taken in as the 'reference_sequence'. It is stored in the 'reference_sequence' list.

Next, the 'collection.find()' method is called to execute the query and retrieve the matching documents. A loop is then used to iterate over each document in the result set, with a counter variable 'count' used to keep track of the index of each sequence.

The program then iterates through all other documents in the result set, a sequence is created by appending the location of the current document to the 'sequence' list. The next four documents

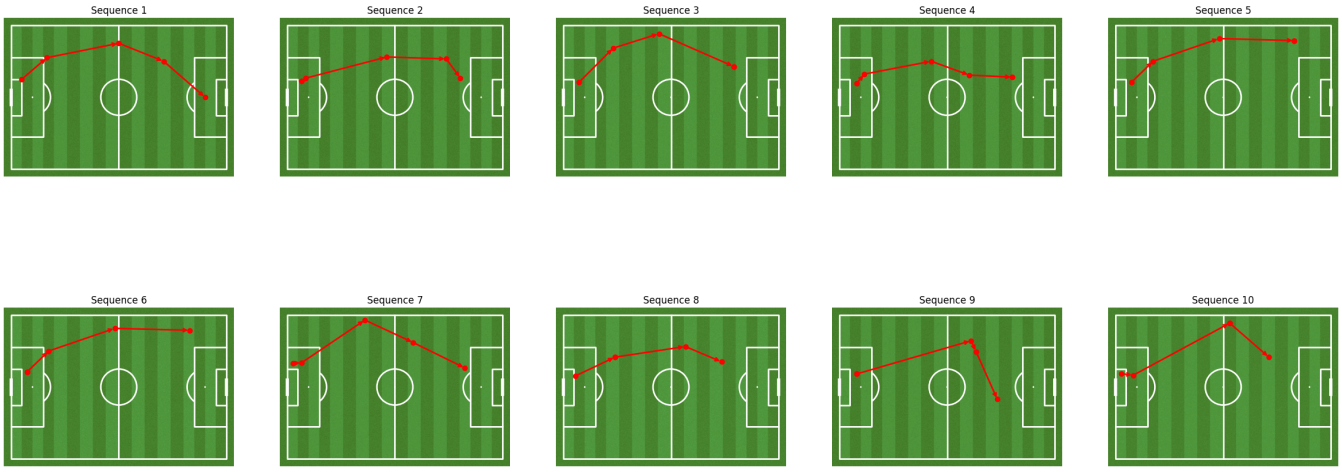

Figure 1: Top 10 Hits for Input Sequence A (Reference Sequence)

in the same match with a higher index value are then found, and their locations are appended to the 'sequence' list as well.

Finally, the 'fastdtw' algorithm is used to calculate the distance between the reference sequence and the current sequence, using the Euclidean distance metric. The distance and the sequence index are then printed to the console, and the current sequence is appended to the 'sequences' list.

5.9 Experiment Setup

The experiment was conducted on two machines to ensure that the functionalities were usable cross-platform. The systems are summarized below:

System 1 is a macOS-based system. The system operates on macOS Ventura version 13.1. The processor is an Apple Silicon M1 chip. The memory is 8.0 GB. The system uses Python 3.9.16, PyMongo 4.3.3, MongoDB 6.0.5, and Mongosh 1.8.0.

System 2 is a Windows-based system. The system operates on Windows 11. The processor is AMD Ryzen 7 6800H with Radeon Graphics (3.20 GHz) with a 64-bit operating system and x64-based processor. The system has 16.0 GB memory. The system uses Python 3.9, PyMongo 4.3.3, and MongoDB 1.36.2.

6 RESULTS

In this section, we present the results generated by the system after testing user input sequences of events.

Input Sequence A `[[[6.0, 30.0], [20.0, 18.0], [60.0, 10.0], [85.0, 20.0], [108.0, 40.0]]]`

We used Input Sequence A for our test run, which was chosen by the team. The system was designed with the intention to traverse through a massive repository of 4.7 million events, with the objective of identifying sequences that display a close resemblance to our input sequence. To ensure a level of precision in our pattern matching, we set the radius to 5 during the testing phase. Given that the StatsBomb data is measured in yards, the $geoWithin function was programmed to pinpoint coordinates within approximately 5 yards of each respective point in Input Sequence A.

The results generated by our algorithm were overwhelming. Our algorithm generated a total of 24,646 hits with varying degrees of similarity. Table 1 displays the top 20 results. This remarkable figure not only showcases the algorithm's ability to mine through an extensive dataset but also it's precision in identifying patterns that closely align with the input sequence.

After calculating the distances, we reshuffled the sequences into ascending order starting with the sequence with the smallest overall distance from Input Sequence A. Then, we visualized the top 10 hits to Input Sequence A in Figure 1, which we refer to as the Reference Sequence in our code.

To facilitate a visual interpretation of our findings, we generated diagrams to represent the top 10 hits against Input Sequence A. As depicted in Figure 1, the sequences appear to be quite similar within the specified range of the radius. This visual representation effectively demonstrates the accuracy of our pattern recognition approach, further validating our use of DTW in analyzing football event data.

We tested the system on two different setups, and in both instances, the runtime was quite similar.

The Search Query that interacts with MongoDB takes 0.0s. It seems to be instantaneous regardless of the number of results returned. When querying for the next N events, the program takes approximately 50s when querying about 28,000 sequences. When it queries 48,000 sequences, it takes approximately 1m 10s. We expect that the runtime increases with the number of sequences. The Distance Calculation takes approximately 5-10s when working with 40,000 sequences. The Sorting Method takes approximately 0.1s. Displaying the Matches takes a couple of seconds depending on the system being used.

In sum, the results of our study offer compelling evidence of the power and potential of our innovative approach. The system's performance, as demonstrated in the testing phase, underscores its capability to handle large datasets and its precision in identifying and ranking sequences based on their similarity to a given input sequence. These results provide a solid foundation for further exploration and application of DTW in the field of sports analytics.

7 CONCLUSION

In conclusion, our research presents a novel application of Dynamic Time Warping (DTW) for pattern recognition in football event data. By grappling with the inherent complexities of spatiotemporal sequences in football, we have demonstrated how our approach can unlock rich, nuanced insights that go beyond traditional analysis methods. The methodology we have developed not only provides a robust measure of pattern similarity, but also opens up new avenues for understanding game dynamics, player performance, and team strategies.

Testing with the user Input Sequence A illustrates the power of our approach. Through querying a vast database of 4.7 million events, our system successfully identified sequences closely resembling our input, underlining its potential to process complex football event data. The system's ability to generate a substantial number of hits (24,646 in total) and then rank them based on the similarity to the input sequence proves its efficiency and precision. The visualizations of the top 10 hits further confirmed the system's capability to identify similar sequences within a specified radius. The success of this approach has wide-reaching implications for sports analytics, particularly in developing game strategies and analyzing player performance. It not only improves the field of sports analytics by providing a powerful tool for pattern recognition in football but also offers valuable insights for coaches, players, and analysts.

The exploration we have undertaken is, however, only the beginning. While our study has shown promising results, the full potential of our methodology can only be realized through further research and application. Looking ahead, we see a wealth of opportunities for refinement and expansion. Future work could involve fine-tuning the parameters of our DTW application, exploring other similarity measures, or applying our methodology to more extensive datasets and other sports.

REFERENCES

[1] A Anon. 2023. What is mongodb? https://www.mongodb.com/what-is-mongodb

[2] Zhongbo Bai and Xiaomei Bai. 2021. Sports big data: management, analysis, applications, and challenges. *Complexity* 2021 (2021), 1–11.

[3] Alina Bialkowski, Patrick Lucey, Peter Carr, Yisong Yue, Sridha Sridharan, and Iain Matthews. 2014. Large-scale analysis of soccer matches using spatiotemporal tracking data. In *2014 IEEE international conference on data mining*. IEEE, 725–730.

[4] Javier Fernandez and Luke Bornn. 2018. Wide Open Spaces: A statistical technique for measuring space creation in professional soccer. In *Sloan sports analytics conference*, Vol. 2018.

[5] Andreas Grunz, Daniel Memmert, and Jürgen Perl. 2012. Tactical pattern recognition in soccer games by means of special self-organizing maps. *Human movement science* 31, 2 (2012), 334–343.

[6] Joachim Gudmundsson and Michael Horton. 2016. Spatio-temporal analysis of team sports–a survey. *arXiv preprint arXiv:1602.06994* (2016).

[7] Patrick Lucey, Alina Bialkowski, Mathew Monfort, Peter Carr, and Iain Matthews. 2015. quality vs quantity: Improved shot prediction in soccer using strategic features from spatiotemporal data. (2015).

[8] Wes McKinney. 2020. Data Structures for Statistical Computing in Python. https://pandas.pydata.org/.

[9] Inc. MongoDB. 2021. PyMongo 3.11.4 Documentation. https://pymongo.readthedocs.io/.

[10] Meinard Müller. 2007. Dynamic time warping. *Information retrieval for music and motion* (2007), 69–84.

[11] Luca Pappalardo, Paolo Cintia, Alessio Rossi, Emanuele Massucco, Paolo Ferragina, Dino Pedreschi, and Fosca Giannotti. 2019. A public data set of spatio-temporal match events in soccer competitions. *Scientific data* 6, 1 (2019), 236.

[12] Andrew Petersen. 2021. mplsoccer: A Matplotlib Extension for Visualizing Soccer / Football Stats, Tracab and Event Data. https://mplsoccer.readthedocs.io/.

[13] Chotirat Ann Ratanamahatana and Eamonn Keogh. 2005. Three myths about dynamic time warping data mining. In *Proceedings of the 2005 SIAM international conference on data mining*. SIAM, 506–510.

[14] Robert Rein and Daniel Memmert. 2016. Big data and tactical analysis in elite soccer: future challenges and opportunities for sports science. *SpringerPlus* 5, 1 (2016), 1–13.

[15] Robert Rein, Dominik Raabe, and Daniel Memmert. 2017. "Which pass is better?" Novel approaches to assess passing effectiveness in elite soccer. *Human movement science* 55 (2017), 172–181.

[16] Hiroaki Sakoe and Seibi Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing* 26, 1 (1978), 43–49.

[17] Stan Salvador and Philip Chan. 2004. FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space. http://www.cs.unm.edu/~mueen/FastDTW.html.

[18] OpenData Statsbomb. 2023. Statsbomb/open-data: Free football data from StatsBomb. https://github.com/statsbomb/open-data

[19] Paolo Tormene, Toni Giorgino, Silvana Quaglini, and Mario Stefanelli. 2009. Matching incomplete time series with dynamic time warping: an algorithm and an application to post-stroke rehabilitation. *Artificial intelligence in medicine* 45, 1 (2009), 11–34.

[20] Stefan Van Der Walt, S. Chris Colbert, and Gaël Varoquaux. 2011. The NumPy Array: A Structure for Efficient Numerical Computation. https://numpy.org/.

[21] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, and İlhan Polat. 2020. SciPy: Open source scientific tools for Python. https://www.scipy.org/.

[22] Yu Yi Yu, Paul Pao-Yen Wu, Kerrie Mengersen, and Wade Hobbs. 2022. Classifying ball trajectories in invasion sports using dynamic time warping: A basketball case study. *Plos one* 17, 10 (2022), e0272848.