Utrecht University

Artificial Intelligence

Master Thesis

Towards automatically classifying football formations for video analysis

Author
T. Boomstra
5721369

Supervisors
Dr. M. van Ommen UU
Prof. dr. R. Veltkamp UU
P. Morel AFC Ajax

May 16, 2022

Acknowledgements

This research would not have been possible with the help of various persons. First, I would like to thank Dr. Thijs van Ommen assistant professor at Utrecht University for his sharp view and quick adaption to all the algorithm options I discussed with him. This came forward in the consistent meetings we had, which kept me on track during this proces. Furthermore, I would like to thank Pieter Morel head of video analysis at AFC Ajax for having the confidence to support this research in an environment of video analysts, where AI is still a very foreign and new field. Lastly, I want to dedicate this research to my father who passed away in 2015 and helped me fall in love with the beautiful game of football.

"Je gaat het pas zien als je het doorhebt."

Johan Cruijff

Abstract

Formations form the tactical basis of football analytics. Formations influence how aggressive a team plays, how they create chances and how they defend. Because of the increased availability of positional data, there has been a rise in the research towards automatically recognizing football formations. However, most of the previous research has not been clear on the effectiveness of their algorithm and has not shown how these algorithms can be used by domain experts. This study has three aims. First, it reports on the accuracy of three previously proposed algorithms (Müller-Budack et al., 2019; Narizuka & Yamazaki, 2019; Shaw & Glickman, 2019) to show their effectiveness. Secondly, it proposes several improvements for these algorithms. Lastly, it shows how these algorithms can be incorporated into the workflow of domain experts. In this research, the existing algorithms perform between 13% and 28% accurate (Müller-Budack et al., 2019; Shaw & Glickman, 2019) respectively on a difficult dataset based on Wyscout reports. I propose several improvements which increase the accuracy of the best performing algorithm to 45% and which showed a promising 78% in a case study. Finally, I examined how these algorithms could be used by domain experts. First, a method is proposed to convert the formation analysis into an XML, which can be imported into the video tools used by the domain experts. Secondly, I show how these algorithms can be used for data analysis.

Contents

1 Introduction

In 1950 there was uproar in England. Aston Villa star Trevor Ford had just been sold for 30,000 pounds. When asked about this price tag, he said: *"These fees are crazy. I think I shall settle down and not let this £30,000 tag worry me, but at the moment I hope someone else soon takes over as the costliest player in the game."* (Graham, 2014). Fast-forward to 2017 and Neymar moved from Barcelona to PSG for a record-breaking fee of over 220 million euros (Larkin & Reeves, 2018). According to global media company Forbes, the most important clubs in the world are now worth well over 4 billion dollars now and have yearly revenues that top 700 million (Ozanian, 2021). The matches for promotion to the highest league in England (Premier League) guarantee the winners an amount of 100+million pounds annually (League, 2017). All this goes to show that football has turned from sports into business. For each game, the stakes have become important. Therefore, football clubs are increasingly looking for ways to find something that might give them the edge over other clubs. One of the most fundamental things that might give teams an advantage over other teams is thorough team analysis.

Traditionally, football match analysis has been through training and coaching. Trainers developed exercises that would correspond to real-world football situations. This remains the bedrock of football today. However, new techniques have started to penetrate football. Since the early 2000s, video analysis has become a popular way of match analysis (Barris & Button, 2008). Video analysis is an excellent tool since it provides the coaches with the ability to watch what happened afterwards, without having to rely solely on memory or notes. This greatly reduces bias and increases the reliability of analysis. Another aspect that has become popular in football is data analysis. In the past, the data that was analyzed was mainly limited to frequency distributions of certain game events (amount of shots, passes etc.). This data was shallow, and thus video analysis was seen as more insightful than data.

Over the last few years, there have been some revolutionary steps in the combination of match analysis and data. The World Cup Final in 2006 kick-started this: it was the first match in which positional data was available (Memmert & Rein, 2018). In the next decade, positional data has become more widely available to a variety of clubs and institutions. This has allowed football to move away from analysis that is often arbitrary and ambiguous. Positional data allows algorithms to manipulate the data. In turn, this permits for analysis on a large scale, rather than relying on human intelligence to watch many games. With positional data being available, the door to science was opened. It has lead to an explosion of research, with countless papers being published each year. Over the last four years the amount of papers published relating football and data analysis has nearly doubled (Google trends, 2021). This has the potential to revolutionize football. These approaches, however, have often failed to address the implementation part. To show this I will first describe several

trends and then I will describe some recurring limitations in these papers.

1.1 Trends

The first considerable trend is the automatic recognition of football events. Football clubs often have several scouts or video-analysts who annotate matches. For example, when a corner happens, they press a certain key to annotate this event. With the rise of accessible positional data, research has gone into automatically detecting these events. This would give domain experts to then focus on other parts. Because events are classified in the thousands to millions at clubs and statistics providers such as Opta, it might seem easy to train a classic learning algorithm to automatically annotate these events. Many papers have been published in this regard, using the latest ML techniques such as LSTMs, CNNs and GANs (Barra et al., 2021; Sorano et al., 2020; Theagarajan & Bhanu, 2020). The main problem is that each team annotates events differently. For example, Ajax has a different playing style and philosophy than Feyenoord and thus the scouts are interested in different events and in fact might interpret similar events differently. Therefore, commercial partners such as Opta have yet to create an algorithm for automatic recognition of events. Also, some playing style related annotations are very hard to recognize automatically. Correctly classifying ball recovery was only 62% accurate in research conducted by Barra et al. (2021). The passing recognition algorithm developed by Sorano et al. (2020) reached an accuracy of 70%. Yet these events are not even the most difficult to recognize, since some playing principles consist of several passes and tactical choices. So, the scientific community is far away from solving this case. This goes back to the challenges mentioned in the introduction. The scientific community is developing methods that can right now hardly be used by domain experts.

The second development is the enormous interest in goal prediction (Inan, 2020; Pettersson & Nyquist, 2017; Wagenaar et al., 2017). These papers only include a minor fraction of the total research done in goal prediction. Newer, larger architectures are being proposed every year to predict goals more accurately. However, what is the actual use-case of such research? At first, it may seem interesting: after all, goals are what determines the outcome of a football match. Though it may seem a paradox, goals contain little information. If an algorithm predicts two goals beforehand, what should a football staff do with that information? It gives no insight how trainers should change things and also why they should change things. It is as if you are looking at a black box that predicts the future, but gives no way to change the future. Perhaps if goal prediction is combined with other research it could become interesting. For example, what events lower the chance of a goal? Goals in itself do not seem to offer much information. Nonetheless, the research might be applicable and interesting for betters and betting companies (Stübinger et al., 2019).

A third development is more insightful for the analysis of football. Positional data has allowed a move away from shallow metrics such as possession, shots and corners that just like goals give little information about the actual game. These new metrics are called Key Performance Indicators (KPI). They model particular aspects of modern football and provide a basis for scientific, data-driven analyses (Memmert & Rein, 2018). One example is the pressing index: the average speed of all players with respect to the ball is calculated to measure the pressing behavior of the defensive team during the transition phase after losing ball possession. Research found that losing teams significantly press more when a game is lost by a difference of 2 or more goals ($W = 271, p < 0.05$) (Memmert & Rein, 2018). These metrics are a step in the right direction. A positive example has been the good reception of the Expected Goal metric, which is now used by a variety of clubs. It measures the quality of a chance by calculating the likelihood that it will be scored. This score is based on several things, such as the angle, distance to the goal, etc. The value of an Expected Goal, also known as xG, lies between one and zero: zero represents a chance that cannot be scored and one represents a chance that a player would be expected to score every single time (Rathke, 2017).

A fourth field is pattern recognition. It can be closely connected to KPI, in which KPI can be used to identify playing styles. Fernandez et al. (2016) did some extensive research on this topic. They used 19 performance measures (14 attacking, 5 defending) to identify playing styles. Playing styles give some insight into how opponents are playing, but still there seems no follow-up plan how to actually use this information. For example, in the same article, Fernandez-Navarro et al. (2016) give a full table of how certain KPIs influence playing style, but they offer no suggestions how coaches should use this data. Another example is the research by Wang et al. (2015), where they use Latent Dirichlet Allocation to find certain passing patterns in Barcelona football data. Unfortunately they only offer some heatmaps as examples of what a coach might get to see from this.

All in all, I have described different trends in football. Modern techniques in machine learning are used by data analysts, but the researchers do not seem to discuss how football clubs should go about using them. Considering that decision-making at clubs is done by trainers and scouts, who have no statistical background, this is problematic, since valuable research is not being used.

Since this research is commissioned by both Ajax and Utrecht University, a unique opportunity presents itself. On the one hand, it allows me to use these algorithms, but on the other hand these algorithms can be tested for their usefulness at an actual professional football club.

2 Related work on formations

This research will focus on pattern recognition in football. As explained in the previous section, pattern recognition has in potential the most explanatory value for match analysis. Generally, pattern recognition in football relies on positional data captured by tracking devices in stadiums, often conducted by STATS or Chreonhego.

The specific pattern that will be researched is a formation. Formations form the basis of football tactics. Tactics can be defined by how a team manages space and individual actions during the game. A formation describes how the players in a team generally position themselves on the pitch (Fradua et al., 2013). Ayanegui-Santiago (2009) defines it as "A specific structure defining the distribution of players based on their positions within the field of play." Müller-Budack et al. (2019) put it differently: a formation describes the spatial arrangement of players by dividing them in tactical groups (defenders, midfielders, attackers). This explains why formations are often presented as e.g., 4-3-3 (4 defenders, 3 midfielders, 3 attackers). This study considers a formation to be defined as

Definition 1. *A set of ten player roles in which a role is defined by the relative spatial distribution over a given time, and in which the set is characterized by how similar it is to a limited amount of predefined formations.* [1].

This definition requires some extra explanation. The ten player roles are defined by their spatial distribution, since a role demands a player to be in certain positions. For example, a left-winger will generally be on the left side of the pitch and high up the pitch. The spatial distributions are relative, since sometimes teams play on a formation on their own half and sometimes on the half of the opponent or in between. These formations can still be the same, which is why absolute positions do not convey much information. Lastly, although there are infinite sets of relative player positions possible, there are only so many formations possible. Usually, domain experts have set up ground rules for how a formation looks. Generally there are around twelve to twenty formations characterized by domain experts (Müller-Budack et al., 2019; Shaw & Glickman, 2019), so a set will be one of those formations and thus different sets of position distributions can belong to the same formation. Though formations might be caught in a definition, catching them in algorithms has been proven difficult. This has to do with the nature of the game. Football is a free flowing game and the roles from definition 1 are loosely defined. In some 4-3-3s the backs will play like midfielders, but in other tactics they play as defenders. However both are seen as 4-3-3.

[1] The goalkeeper is omitted as he/she has no role in defining the formation. There are different keeper roles, e.g. Neuer of Bayern Munich plays in a sweeping style, but this does not define the formation.

However, being able to recognize these formations can have sizeable consequences. Different formations have different strengths, and thus playing different formations has different results. Therefore, it is one of the first things that opposition analysts aim to identify in their assessment of a team (Low et al., 2021). For example, Aquino et al. (2017) found that players have more high intensity sprints in certain formations. Memmert and Rein (2018) noted that certain formations receive fewer goals than others. Memmert et al. (2019) found that teams using the 3-5-2 formation outplayed more defenders when passing under pressure, compared to a 4-2-3-1 formation. Newer research established that playing width was larger when attacking against a 4-4-2 defense, compared to a 5-3-2 defense (Low et al., 2021).

Bialkowski et al. (2014) published a widely influential paper that was a first proper attempt to automatically detect formations from tracking data[2]. The authors made important assumptions about formations that are used throughout all the new publications regarding football formations. They argued that formations reflect the relative spatial arrangement of the players and not the absolute positions. However, they assumed that the dominant formation is stable within a match half and this is not sufficient to describe the complex and varying tactical formations in modern football, as they often change within a half. In the years after, Bialkowski et al. (2016) improved their previous method in which they use heatmaps to discover player roles, but they still assumed stable formations within a half. Their improved algorithm reached an accuracy of 75%. Machado et al. (2017) made a football analysis tool that uses k-means clustering to assign each player into one of three tactical groups (defender, midfielder, attacker). Their method however is limited in the sense that it only used the x coordinates. Furthermore, in contrast to their methodology, in some formations there are more than three tactical groups. For example, a 4-2-3-1 splits the midfielders into two groups. Wu et al. (2018) proposed a visual analytics system, that could distinguish between attacking and defensive formations. This paper, however, focused mostly on the translation from video to positional data and the visual aspect of such the analysis tool.

In 2019, three new papers were published that allowed for automatically recognizing formations on short sequences. Shaw and Glickman (2019) wrote an article on dynamically recognizing football formations. Their algorithm clusters football formations of 2 minutes and they proposed a Bayesian classification algorithm, to classify new observations. A limitation of this research is that the authors fail to present the actual results of their research in terms of key metrics such as accuracy and recall. It is thus unknown how well this method performs. In the same year Narizuka and Yamazaki (2019) proposed an algorithm based on Delaunay networks that could cluster formations on a variable time period. The most substantial limitation is that their algorithm cannot be used

[2]Before the 2010s tracking data was not widely available if at all, thus the research before this period had little effect and mainly focused on translating video to data.

to automatically cluster multiple games. Lastly, Müller-Budack et al. (2019) proposed a classification method for formations by using Ground-Truth annotations. Because they do not rely on clusters, this makes their method ad-hoc and quicker. Their method, however, only showed 20% accuracy.

3 Research questions

In this research, I will focus on the current state of the research regarding automatically classifying formations. Furthermore, I will investigate whether it is possible to improve the current state-of-the-art techniques. Therefore, the following research questions are researched:

- RQ1: Is it possible to accurately cluster and classify football formations using existing geometric algorithms?
- RQ2: How do these algorithms compare against each other?
- RQ3: How can elements from these algorithms be combined into a better, new algorithm?

The existing geometric algorithms as referred to in RQ1-RQ3 were the three papers that were published in 2019 (Müller-Budack et al., 2019; Narizuka & Yamazaki, 2019; Shaw & Glickman, 2019) In my introduction, I mentioned that a lot of research with regard to football and data can often not be used by actual domain experts. Therefore, my fourth research question sees to adress implementation.

- RQ4: How can this research be implemented in the workflow of domain experts?

4 Three methods for clustering and classifying formations explained with their limitations and how to overcome these

In the next section three popular methods for clustering or classifying football formations will be explained in detail. I will also explain the strengths and weaknesses of each method.

4.1 Method proposed by Shaw and Glickman (2019)

In this section the paper by Shaw and Glickman (2019) is explained in detail. The authors propose a hierarchical clustering method for clustering football formations and a Bayesian component to classify new formations. The authors used 100 games for their clusters. Apart from explaining their paper, I have also added several limitations and problems of this paper as well as possible solutions to these problems and limitations.

4.1.1 Relative positions

The first aspect of the research of Shaw and Glickman (2019) is that the authors make use of relative positions. In contrast to Bialkowski et al. (2014) and Narizuka and Yamazaki (2019) the relative positions are measured towards a different player instead of to the center of mass (calculated by averaging all the positions and further explained in equation 13). The authors give the following reasoning: outfield players in a team will tend to cover only a small fraction of the pitch. Also, players move coherently as a group to maintain their spatial configuration. Formations are therefore defined by the relative positions of the players. Summarized, Shaw and Glickman (2019) argue that a player's position is influenced by a few players and not the whole team. Therefore, they refrain from taking the distance from the center of mass of the formation. Instead, they calculate the distance to the nearest neighbor. To calculate these relative positions, they have come up with an iterative algorithm. Assume that there is a set of 10 players vectors $P(t) = \{\vec{p_1}(t), ..., \vec{p_{10}}(t)\}$ in which $\vec{p_i}(t)$ refers to the absolute position of player i at time t: $\vec{p_i}(t) = [x_i(t), y_i(t)]$. First, they calculate an average distance vector between each player. For the first step of the algorithm, the positions over time t need to be averaged per player.

$$\vec{a_i} = \frac{1}{T}\sum_{j=1}^{T} \vec{p_i}(t_j) \quad \forall \vec{p_i}(t) \in P \tag{1}$$

in which a_i refers to the average positions of player during time t. Then they calculate the distance between each player pair

$$D_{i,j} = F(\vec{a_i}, \vec{a_j}) \quad \forall \vec{a_i} \in P; \forall \vec{a_j} \in P \quad where\ i \neq j \tag{2}$$

in which F is the Euclidean distance

$$F(\vec{a_i}, \vec{a_j}) = \sqrt{\sum_z (a_{iz} - a_{iz})^2} \tag{3}$$

between two players. Then, the authors set the centroid c as the densest player which is determined by the average distance to the third-nearest neighbor. Once they have set this player, they check its closest neighbor. The relative position of this neighbor is the relative position to the previous player (ignoring any player already considered in the process). They continue this process until no player is left. Thus, the relative positions $\vec{r_i}$ of then 10 players at time t are obtained. An important note is that these observations are made of the relative positions over a certain length. The authors choose to aggregate consecutive possessions into two-minute, non-contiguous time periods. They exclude possessions that last for less than five seconds since it is assumed that they are too short for either team to establish an offensive or defensive stance.

4.1.2 Wasserstein Distance

Based on these relative positions, per player, a bi-variate distribution is made. This bi-variate distribution consists of the mean $\mathbf{m}$ in x and y and the co-

variance matrix $\mathbf{\Sigma}$: the deviation of a player's position in that observation. These bi-variate distributions can be used to quantify similarity between formations. To quantify the distance between two formations, the Wasserstein distance is used. This is a distance function defined between probability distributions. The Wasserstein distance between two multivariate distributions $\mu_1 = \mathcal{N}(\mathbf{m_1}, \mathbf{\Sigma_1})$ and $\mu_2 = \mathcal{N}(\mathbf{m_2}, \mathbf{\Sigma_2})$ is calculated in the following manner: (Olkin & Pukelsheim, 1982)

$$W(\mu_1, \mu_2)^2 = \|\mathbf{m}_1 - \mathbf{m}_2\|^2 + \text{trace}\left(\mathbf{\Sigma}_1 + \mathbf{\Sigma}_2 - 2\left(\mathbf{\Sigma}_2^{1/2}\mathbf{\Sigma}_1\mathbf{\Sigma}_2^{1/2}\right)^{1/2}\right) \quad (4)$$

This Wasserstein distance is calculated between each player of two observations. This results in a 10x10 matrix of distributions distances between players called D_{ij}. However, the distance cannot simply be summed. In order to get a final distance measure, there needs to be a role-assignment. Therefore, the authors propose to use the Hungarian algorithm which returns the pairs of players that minimize the cost. The Hungarian algorithm can find the optimal paring of the players from two observations. The Hungarian algorithm runs in $O(n^3)$ (Jonker & Volgenant, 1986) and is thus quite computationally expensive, but still much cheaper than trying all the 10^{10} combinations for the smallest distance. So each player is compared to a similar player in the two different observation:

$$W_{total}^2 = \min \sum_i \sum_j D_{ij} X_{ij} \quad (5)$$

in which D_{ij} is the Distance matrix. Player i of observation 1 is matched with player j of observation 2. $X_{i,j}$ is the allocation matrix returned by the Hungarian algorithm. As shown in equation 5, there is a summation over these player pairs distances. This summation is used as the final distance between two formations. The distance is thus a measurement of how close two formations are to each other. To deal with wide and narrow formations, a scaling factor k is introduced. Two formations may be identical in terms of their shape (e.g. a traditional 4-4-2), but one may be a more compact or expanded variant of the other. However, the algorithm needs to cluster these as the same. Therefore, center of mass is scaled with a factor k (scaling the co-variances accordingly) that minimizes the distance between two observations.

4.1.3 Agglomarative hierarchical clustering

To cluster the formation observations into distinct formations, agglomarative hierarchical clustering is used. The authors choose to use 100 matches as a training set, and choose 20 as the number of clusters. This unsupervised learning technique is quite simple. It works as shown in algorithm 1.

Algorithm 1 Agglomorative clustering.

1: Turn each formation observation into a singleton cluster
2: For each pair of clusters C_i, C_j calculate the distance.
3: Merge the pair with the smallest distance into one cluster.
4: Continue step 2 & 3 until a stopping criterion is met.

Step 3-4 require some extra information as there are several options to go about these steps. First, there are several ways to merge clusters. In the following section, I will explain how these methods work, and what their advantages and disadvantages are. Also, a new method suitable for this specific problem will be proposed.

The min method, also known as single-linkage, can be denoted as min distance (P_i, P_j) where $P_i \in C1$ and $P_j \in C2$. In practice, this means selecting two points from different clusters where the distance is minimized. It is visualized in figure 1. This method is cheap to compute and works for non-elliptic shapes. However, it does not work well for clusters with outliers, since it merges on single observations (Müllner, 2011).

Figure 1: Distance based on minimum.

Another way is the max distance, also known as the complete-linkage. It is similar to the min method, though the outcomes can be quite different. It is denoted as max Distance (P_i, P_j) where $P_i \in C1$ and $P_j \in C2$. This is visually represented in figure 2.

Figure 2: Distance based on maximum.

The most important advantage compared to the min method is that the max method can deal with outliers. However, it tends to break large clusters. A third way is average clustering, in which the distance between each observation from two clusters is calculated. Over all these distances, the average is calculated. This can be denoted as

$$\overline{D} = \frac{1}{|C_1| \cdot |C_2|} \sum_{P_i \in C_1} \sum_{P_j \in C_2} d(a, b) \tag{6}$$

and is shown in Figure 3.

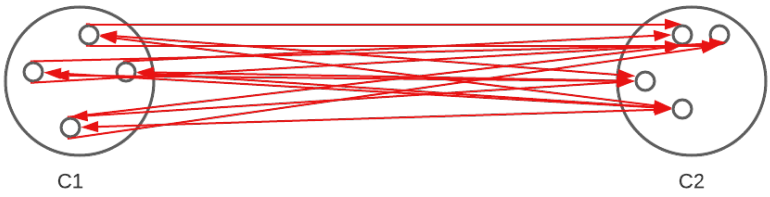


Figure 3: Distance based on average.

This method does well if data has noise, since the average is based on all the points rather than single points as with the min and max distance. However, as the average of this cluster will generally be close to new elements, this method is biased towards large clusters. The last conventional method I will take into account is Ward's method. It is also the one proposed by Shaw and Glickman (2019). Instead of measuring the distance directly, it analyzes the variance of clusters (Yuxuan Hu & Meng, 2018). Ward's method says that the distance between two clusters, C_1 and C_2, is how much the sum of squares will increase

when we merge the two clusters together. We can write this as

$$\Delta(C_1, C_2) = \sum_{i \in C_1 \cup C_2} ||\overrightarrow{P_i} - \overrightarrow{M}_{C_1 \cup C_2}||^2 - \sum_{i \in C_1} ||\overrightarrow{P_i} - \overrightarrow{M}_{C_1}||^2 - \sum_{i \in C_2} ||\overrightarrow{P_i} - \overrightarrow{M}_{C_2}||^2 \tag{7}$$

$$=$$

$$\frac{N_{C_1} N_{C_2}}{N_{C_1} + N_{C_2}} ||\overrightarrow{M}_{C_1} - \overrightarrow{M}_{C_2}||^2 \tag{8}$$

where $\overrightarrow{M_i}$ is the center of cluster i and N_i is the number of points in it. Δ is known as the cost of merging two clusters. In the beginning of agglomarative clustering, each cluster is a singleton consisting of one observation. Therefore, the sum of squares begins at zero. As clustering starts, Ward's linkage will keep the sum of squares as low as possible. Under the assumptions of Ward's linkage, the Euclidean distances between the center of the cluster ($\overrightarrow{M_i}$) and the different points in the clusters P_i where $i \in C$ and C are all the clusters are needed. Because the objective function is based on the distances between the centroids of the clusters, it is necessary to use the squared Euclidean distance as the metric to calculate distances between objects.

The last method is specific for this problem. It is explained in Algorithm 2.

Algorithm 2 Wasserstein clustering.

1: DistanceMatrix = $DistanceMatrixBetweenEachObservation(AllObservations)$
2: ClusterList = $InitializeCluster(AllObservations)$
3: cluster1, cluster2 = $SmallestDistance(DistanceMatrix)$
4: MergedCluster = $MergeClusters(cluster1, cluster2)$
5: ClusterList = $UpdateClusterList(ClusterList, MergedCluster)$
6: DistanceMatrix = $UpdateDistanceMatrix(MergedCluster, DistanceMatrix)$
7: Continue step 3-6 until a stopping criterion is met.
8: return ClusterList

First a distance matrix between each observation is calculated. After this, the two observations with the minimal distance are merged based on the Hungarian algorithm. Recall that from the Hungarian algorithm we know which player matches with which player, so the columns can be merged based on what players are similar. The Wasserstein distance between the merged cluster and the other clusters can be calculated. This is repeated by again selecting the lowest distance between two observations and keep doing this until a stopping criterion is met. We keep stock of the current clusters in a list of clusters. A stop criterion could be either a cluster amount or a distance cap (the distance is too high for a merge to make sense). Arguably this algorithm is not much different from algorithm 1, but there are two important differences. First, if

Ward's linkage is used, there will be some sort of approximation of the Wasserstein distances, but not the actual Wasserstein distances between two clusters. For that, the Wasserstein distance between two observations would need to be recalculated. Which is where the second difference comes in. The Hungarian algorithm is used as intermediate step to recalculate the distances.

4.1.4 Bayesian Classification

The previous part has discussed clustering formations, but Shaw and Glickman (2019) wrote an extension to classify new observations. The equation classifies new observations according to

$$\mathbb{P}(\mathrm{o} \mid \mathrm{C}) \approx \underset{k}{\operatorname{argmax}} \prod_{i=1}^{10} \int \mathrm{p}\left(y \mid k\mu_{\mathrm{i,C}}, k^2\Sigma_{\mathrm{i,C}}\right) \mathrm{p}\left(y \mid \mu_{\mathrm{i,o}}, \Sigma_{\mathrm{i,o}}\right) \mathrm{d}y \tag{9}$$

in which $\mathbb{P}$ is the probability of observation o belonging to cluster C. This extension is powerful as it allows testing the accuracy of this algorithm and it makes the algorithm useful in real world use-cases: new observations can be classified. The way this algorithm works is by assuming that similar formations consist of similar distributions. If the distribution of player i from observation o is similar to the distribution of player i of cluster C, then they will both have high probability on certain areas on the field. And thus the multiplication of both will be larger than 0. But if the distributions do not match, the C probability for y might be high, but the o probability y will be low, and thus the product is small. Thus, the more similar the two distributions, the higher the product. Note that y implicitly refers to position x, y, since it would make little sense to integrate over just the y positions. To assign each player in an observation to a specific role in a cluster, the Hungarian algorithm is used, as described in the previous section. An *argmax* over the scaling property k is used to deal with narrow and wide formations.

The authors say little about how this integration can be achieved, so I will provide a more detailed explanation down below. Assume that for each player p in both C and o we have that $\mathcal{N}_y(\mathbf{m}, \mathbf{\Sigma})$. Note that the probability density function of a single multivariate-gaussian distributions is given by

$$\phi(\mathcal{N}) = \left(\frac{1}{2\pi}\right)^{p/2} |\Sigma|^{-1/2} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mathbf{m})'\Sigma^{-1}(\mathbf{x}-\mathbf{m})\right\} \tag{10}$$

in which $|\Sigma|$ denotes the determinant of the co-variance and Σ^{-1} the inverse. Now the probability distribution of two multivariate Gaussian distributions needs to be found, since there is an observation probability density as well as a cluster probability density (both per player). Suppose there are two probability distributions $\mathcal{N}_y(\mathbf{m}_1, \mathbf{\Sigma}_1)$ and $\mathcal{N}_y(\mathbf{m}_2, \mathbf{\Sigma}_2)$ that both follow equation 10, then to calculate the final probability density, the product of the Gaussian densities needs to be found. Thus we get

$$\mathcal{N}_y(\mathbf{m}_1, \mathbf{\Sigma}_1) \cdot \mathcal{N}_y(\mathbf{m}_2, \mathbf{\Sigma}_2) = c_3 \mathcal{N}_y(\mathbf{m}_3, \mathbf{\Sigma}_3) \tag{11}$$

in which

$$
\begin{aligned}
c_3 &= \mathcal{N}_1\left(\mathbf{m}_2, \left(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2\right)\right) \\
&= \frac{1}{\sqrt{\det\left(2\pi\left(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2\right)\right)}} \exp\left[-\frac{1}{2}\left(\mathbf{m}_1 - \mathbf{m}_2\right)^T \left(\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2\right)^{-1}\left(\mathbf{m}_1 - \mathbf{m}_2\right)\right] \\
\mathbf{m}_3 &= \left(\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1}\right)^{-1}\left(\boldsymbol{\Sigma}_1^{-1}\mathbf{m}_1 + \boldsymbol{\Sigma}_2^{-1}\mathbf{m}_2\right) \\
\boldsymbol{\Sigma}_3 &= \left(\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1}\right)^{-1}
\end{aligned}
\tag{12}
$$

c_3, $\mathbf{m}_3$ and $\boldsymbol{\Sigma}_3$ refer to the new combined Gaussian distribution (Petersen, Pedersen, et al., 2008), the proof can be found in Bromiley (2003). Since it is known that the integral of $\mathbf{m_3}$, $\boldsymbol{\Sigma_3}$ equals to 1 per definition, only c_3 needs to be calculated for each player pair. For each cluster observation combination, the product is calculated, by multiplying c_3 for each probability density function. We end up with a product of probabilities for each cluster. The cluster that has the highest product of probabilities is choosen.

4.1.5 Limitations, flaws and proposed solutions

The article by Shaw and Glickman (2019) has several limitations and flaws. First of all, their iterative algorithm to calculate relative positions a clarified in Figure 4 has a flaw. According to the algorithm, the first step is to calculate the centroid. Assume for this example that in Figure 4 the centroid is center midfielder Davy Klaassen. Then, the relative position of his closest neighbor (Daley Blind) is the relative position to the previous player (Davey Klaassen). This process is continued. The closest neighbor of Daley Blind is Jurrien Timber, so the relative position of Jurrien Timber is the distance to Daley Blind. If this process is continued to the tenth player (recall that the goalkeeper is ignored) the centroid has no relative position. This would result in only nine distributions. As a solution, I propose that the centroid's relative position is

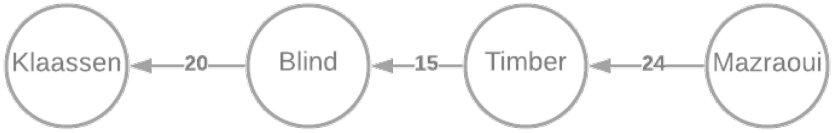


Figure 4: Relative positions example.

the centroid's distance to the center of mass. The center of mass is calculated by averaging all the players' absolute positions.

Another solution is to use the distance to the center of mass as a relative posi-

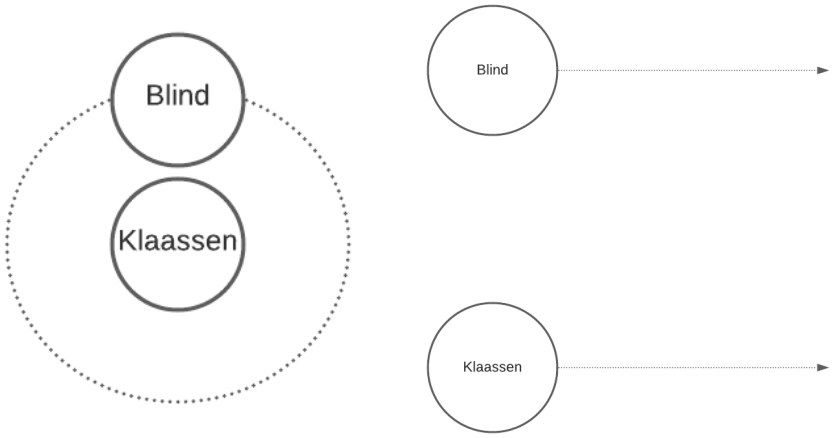
tion

$$
\begin{aligned}
&P(t) = \{\vec{p}_1(t), ..., \vec{p}_{10}(t)\} \\
&\vec{p}_i(t) = [x_i(t), y_i(t)] \\
&c\vec{m}(t) = \frac{1}{N}\sum_{i=1}^{N} \vec{p}_i(t) \\
&\vec{r}_i(t) = \vec{p}_i(t) - c\vec{m}(t) \quad \forall \vec{p}_i(t) \in P(t)
\end{aligned}
\tag{13}
$$

in which $P(t)$ is the set of player vectors at time t. These player vectors consist of an absolute x and y position at time t. The center of mass $c\vec{m}(t)$ is calculated by averaging the absolute positions of each player $\vec{p}_i$ at time t. For each player, the relative position is then obtained by subtracting $c\vec{m}(t)$ of $\vec{p}_i(t)$ for each player vector in the set $P(t)$. Note that center of mass is calculated each frame, since possession observations are merged. Thus, $c\vec{m}(t)$ could be totally different from $c\vec{m}(t+1)$.

Another scientific limitation is that Shaw and Glickman (2019) use Ward's linkage on distances that are not Euclidean but Wasserstein.

The mean and co-variances in the Wasserstein distance can also cause some problems. This is because relative positions give a possibility of the mean being 0 even if the player has moved a lot. Then a lot of valuable information is lost. For example, see Figure 5a, suppose Blind has it relative position to Klaassen and his trajectory is given by the dotted line. He will move a lot, yet his mean relative position is 0. A solution would be to use the squared relative distance, but this would disregard of a lot of interesting information in other situations. For example, in Figure 5b, Blind and Klaassen their trajectories are next to each other. If the squared relative position would be used we would lose the information whether Blind was moving left or right to Klaassen.



(a) Example about the mean being 0. (b) Problems of squaring distance.

Figure 5: Problems that can arise in Wasserstein.

Another limitation of the Wasserstein distance has to do with co-variances. They also play a key role in calculating the distances between two distributions. The authors make 20 clusters based on 100 games. This means that one cluster will be from many games and will thus have large co-variances. The Hungarian matching algorithm is bound to make some mistakes and that will only increase the co-variances. Whereas the observations that will be classified afterwards consist only of a few minutes, and thus their co-variances will be a lot smaller. Therefore, I propose a simple scaling method in which first we calculate the average covariance matrix of the cluster over all the players (*acc*). The same thing is done for the observation (*aco*). Then, a scalar matrix is calculated that divides both average covariances element-wise

$$\alpha = acc \oslash aco \tag{14}$$

the co-variances of the observations are multiplied elementwise by this scalar matrix

$$co_i = co_i \odot \alpha \tag{15}$$

in which co_i refers to a covariance matrix of player i. This operation is applied for each player in the observation. This could lead to a smaller discrepancy between cluster and observation co-variances. One of the dangers is that the values of the of *aco* on the diagonal might be close to zero. This would cause the diagonal of the scalar matrix to become very large. For the Bayesian classification, rather than using the product of the probabilities, I propose to use log as multiplying 10 probabilities would return a number close to 0. Small probabilities could cause to underflow the numerical precision of the compiler Jurafsky and Manning (2012). Furthermore, I suggest using an *argmax* over

two scaling factors, so that x and y can be scaled separately rather than a single k proposed by Shaw and Glickman (2019)). Shaw and Glickman (2019) explain little about this scaling factor, but I presume a grid search is performed over a finite number of k. Then the one with the highest probability is chosen. I propose to scale both x and y by two individual k as a 4-4-2 might be wide but narrow, whilst another 4-4-2 is not wide nor narrow.

A downside of this Bayesian classification method, is that it is rather slow. Let say there are 10 observations need to be classified. Then these 10 observations need to be compared to all the clusters (e.g. 20), then a grid search is performed over the combination of x and y' $k - values$. Then the Hungarian matching algorithm is run, and for each player and double integral needs to be run. One option to decrease the complexity is to get rid of the k-values. Since there is an argmax over the combination a k for y and k for x the complexity is $O(n^2)$. We can simplify this by a priori calculating a K-scaling factor based on the length and in which

$$\vec{k} = \frac{max(O) - min(O)}{max(C) - min(C)} \tag{16}$$

$\vec{k}$ refers to the scaling vector in x and y. This only has to be calculated once, instead of the number of k to the power of two. Another possibility is to use the normalizing as proposed by Müller-Budack et al. (2019), which is explained in section 4.3 or to take positions as a standard deviation to the center of mass as explained in 4.2 (Narizuka & Yamazaki, 2019).

As explained in section 4.1.2 clusters cannot simply be merged. A left-midfielder in observation 1 does not necessarily sit at the same index as the left-midfielder in observation 2. The authors proposed to use Ward's linkage for clustering. But this only returns a list with all observations and to what cluster they belong, not how they should be merged. However, as shown in equation 9 we end up with some final cluster set C. Thus, all the observations should be merged into a limited amount of clusters (20 in the paper by Shaw and Glickman (2019)). Thus, if observation 1 and observation 2 belong to the same cluster, they have to be merged according to the Hungarian algorithm.

There are two possible approaches. First of all, there can be an update function as explained in algorithm 2. For this method, first a distance matrix of $(n \times n)$ is created in which n is the number of observations, with the distance between each observation. Then, two observations are merged with the smallest distance between them according to the Hungarian algorithm. The distance between the merged observation and the other observations is recalculated. This process is continued until the preferred number of clusters is reached. The main disadvantage of this method is that it is time-consuming. For each iteration, the distance between the new observation and all the other clusters needs to be recalculated, which means that the Hungarian algorithm is applied to compare the new observation to all the other observations. In the beginning

of the process this will take a lot of time as almost each observation still forms its own cluster and thus the new observation needs to be compared to many other observations. As the number of clusters decreases, so will the time for an iteration.

Another approach is to use the children returned by the Sklearn agglomarative clustering library for Python [3]. For example, using Ward's linkage the package returns the order in which observations can be merged. Rather than merging two observations and then recalculating the distance between the merged observations and the other observations, the tree returned by the Sklearn is used. The Hungarian algorithm is only used to decide how to merge two observations. The update function then will not be needed, as the tree is followed to check what observations need to be merged. The most considerable advantage is that this saves a lot of time. The other advantage is that different merging methods can easily be compared, since the execution time is very low. However, the disadvantage is that Ward's linkage assumptions work for Euclidean distances and not for Wasserstein distances. Thus it is unknown how well this would work for non-Euclidean distances such as the Wasserstein distance.

Lastly, I propose a different classification method as an alternative to the Bayesian classifier.

Algorithm 3 Distance classification algorithm.

```
int bestDistance = ∞
int bestCluster
obsMeans = []
obsCovariances = []
for i ∈ observation do
    obsMeans[i] = CalcMean(player_i)
    obsCovariances[i] = CalcCoviarances(player_i)
for cluster ∈ Clusters do
    cMeans = []
    cCovariances = []
    for i ∈ cluster do
        cMeans[i] = CalcMean(player_i)
        cCovariances[i] = CalcCoviarances(player_i)
    matrix = WasserteinMatrix(obsMeans, obsCovariances, cMeans, cCovariances)
    tempDistance = HungarianAlgorithm(matrix)
    if tempDistance < bestDistance then
        bestDistance = tempDistance
        bestCluster = cluster
return bestCluster
```

[3] Available at https://scikit-learn.org/stable/install.html (version 0.23 Scikit-learn).

In this algorithm for a new observation, the distance between an observation and all the clusters is calculated by using the mean and co-variance for each player in both the observation and cluster. Based on these arrays, the Wasserstein distance can calculate a distance between each player pair. The Hungarian algorithm optimizes and returns the lowest possible distance. The observation is compared to each cluster, and the one with the lowest distance is returned as the best cluster. This method is almost identical to the way observations were clustered and thus the extra implementation time is limited.

Concluding, the authors of this paper propose a decent method with some interesting aspects that allows for both clustering and classification. However, a lot of important elements in the research by Shaw and Glickman (2019) are implied without proper explanation. This section proposed certain solutions for the implicit knowledge and thus made all of the implicit explicit.

4.2 Method proposed by Narizuka and Yamazaki (2019)

Another method to cluster football formations was proposed by Narizuka and Yamazaki (2019). This research was applied on the J1 League (the Japanese football league). Their method was used on 45 football games. The method bears similarities and differences to the aforementioned method by (Shaw & Glickman, 2019). Again, I will explain the method and note the weaknesses of the paper. This section will be concluded with some suggestions for the main limitations.

4.2.1 Normalizing positions

First, the authors propose to normalize the positions. This is achieved by using the center of mass and its standard deviation calculated with the following formulas:

$$\vec{cm}(t) = \frac{1}{N} \sum_{i=1}^{N} \vec{p}_i(t) \tag{17}$$

in which the position of the j-th player at time t can be denoted as

$$\vec{p}_i(t) = [x_i(t), y_i(t)]$$

This is similar to the center of mass equation as explained in the previous article section. They then calculate the standard deviation of the team

$$\vec{\sigma}(t) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} |\vec{cm}(t) - \vec{p}_i(t)|^2} \tag{18}$$

in which $\vec{\sigma}(t)$ consists of a standard deviation with regard to both x and y. Combining these two formulas and the normalized coordinates $\vec{r}_j(t)$ for player j can be calculated

$$\vec{r}_i(t) = (\vec{p}_i(t) - \vec{cm}(t)) \oslash \vec{\sigma}(t) \tag{19}$$

by first subtracting the center of mass of the player's position and then dividing this by the team's standard deviation. This is done element wise since there is a standard deviation with regard to x as well as y. In reality this means that, again, relative positions are used. This time they are relative to the center of mass and then normalized by dividing them by the standard deviation of the team. If a formation is very wide or very narrow it then does not matter much as the standard deviation serves as a scaling function. The authors then propose to use the adjacency matrix of a Delaunay network. This is a matrix in which for each player it is determined whether the Voronoi regions are adjacent. A Voronoi diagram is a partition of a plane into regions close to each of a given set of objects (Aurenhammer & Klein, 2000). The formula of this Adjacency matrix $A_{ij}(t)$ is as follows:

$$A_{ij}(t) = \begin{cases} 1 & \text{if the } v \text{ regions of players } i \text{ and } j \text{ are adjacent with each other at } t \\ 0 & \text{otherwise.} \end{cases} \tag{20}$$

in which v refers to Voronoi. Narizuka and Yamazaki (2019) note that Delaunay networks are easy for interpretation because of their visualization aspect. This leads to the most important formula of this section: a dissimilarity measure between formations at different timestamps.

$$D_{tt'} = \left\| A(t) - A\left(t'\right) \right\|^2 = \sum_{i=1}^{N} \sum_{j=1}^{N} \left[A_{ij}(t) - A_{ij}\left(t'\right) \right]^2 \tag{21}$$

Their reasoning for using this method is that if a formation changes, a player (or multiple) move towards a different position and thus the Voronoi regions will be different. Therefore, the distance $D_{tt'}$ will increase. Once the distance matrix is obtained, this can be used in the Hierarchical clustering method. Based on the dendrogram returned by the Hierarchical clustering the number of clusters is chosen.

4.2.2 Limitations and flaws

Contrary to Shaw and Glickman (2019), Narizuka and Yamazaki (2019) do not use distinct observations of possession (recall that Shaw and Glickman (2019) used possession of two minutes). In fact, the Delaunay networks are calculated for each frame f. Their tracking data is 25 frames per second. This might make the algorithm a bit too precise, as large positional changes do not occur every $\frac{1}{25th}$ second. Of course, the frames could be down sampled, but this does not get rid of the core problem. As the distance algorithm is $O(n^2)$ it will be very expensive to not preemptively merge observations of possession or dispossession.

Just like Shaw and Glickman (2019), these authors propose to use the Hierarchical clustering with Ward's method. They then plot the dendrogram to determine the number of clusters. This means that the user of this algorithm has to determine the appropriate number of clusters for each game, which is a

substantial limitation.

A more fundamental problem is that this algorithm only clusters one game without substitutions. Once substitutions occur or a different game starts, it is no longer clear if player j at time t is the same as player j at time t'. If we start comparing different players, the whole algorithm ceases to make sense. To compare two formations at a different time t one would need to compare players that play in a similar role. The authors propose a solution for this based on the idea of role presentation introduced by (Bialkowski et al., 2016; Bialkowski et al., 2014). This solution is presented in three steps. In part one, the authors propose to assign index i of the Delaunay network $A(t)$ of players with an approximately similar position. The similarity is approximated by visualizing the average normalized positions in a heat map. For each of the ten players, there is a heat map that represents the covariance matrix. It remains unclear to the reader if the average positions do take into account the clusters calculated in the previous section, or whether they just use the average positions of the game. By using these average formations they find 5 formational patterns: '4-4-2', '4-1-4-1', '4-3-3', '5-4-1' and '3-4-3'. For example, a 4-4-2 formation has a role named the left-striker. Whenever the 4-4-2 pattern occurs the left striker is for instance always stored in $I = 10$ in $A(t)$. There will thus be a direct one-to-one correspondence between a player's role and its index in the adjacency matrix $A(t)$. This is a fundamental issue of this paper. These formational patterns are determined by the authors themselves based on their own interpretation. One would have to manually check different frames and determine which player sits at which index to ensure a good role matching. If a substitution occurs, one would have to visualize the formation again, since the average formation then is not logical. If player j plays left-winger and is substituted for player k who plays center-back in the array they will have the same index, but it would not be appropriate to average these positions as they represent different positions/roles. Moreover, the role correspondence between formations is still a difficult task. For example, the left-winger from 4-3-3 is absent in the formation 4-4-2. Does one put the left-winger at the same index as the left midfielder from 4-4-2 or the left striker? And if it is decided that it receives the same index as the left midfielder from 4-4-2, that leaves a problem for the left center midfielder of the 4-3-3. That role is also absent in 4-4-2. A solution that seems to be proposed by the authors is that the authors manually cluster formations into one of these five formations, and then they further cluster inside such a cluster using the one-to-one role correspondence. However, that would mean that it is only possible to cluster inside a formation, which means there is an assumption that a formation is stable. Ultimately, this proposed method of Narizuka and Yamazaki (2019) can then only detect players switching roles within a formation. However, then it is no longer feasible to see switches in formation automatically.

These problems all arise from the manual role representation introduced to cluster multiple games. A solution for this would be to implement the role rep-

resentation matching proposed by Shaw and Glickman (2019). They also faced the same problem, where one cannot simply compare player i from time stamp t with player i from time stamp $t + 1$. Shaw and Glickman (2019) proposed to use measure the Wasserstein distance between players and use that as an input for the Hungarian algorithm to come to a proper role assignment. However, calculating the Wasserstein distance is expensive: it uses both the co-variances and the means. These co-variances and means are not used in the calculation of the Delaunay network, allowing the algorithm to be quicker. Müller-Budack et al. (2019), on the other hand, propose Hungarian matching based on Euclidean distances. This is a lot cheaper, since only the Euclidean distances would have to be calculated between each player (equation 3, Müller-Budack et al. (2019)) instead of Wasserstein distances (equation 4, Shaw and Glickman (2019)). The procedure is listed in the algorithm down below.

Algorithm 4 Delaunay with player matching

```
DistanceMatrix = [[]]
for i ∈ observations do
    NormalizedObservation1 = NormalizeObservation(observation1[i])
    for j ∈ observations do
        if observation1! = observation2 then
            NormalizedObservation2 = NormalizeObservation(observation2[j])
            Matching1, Matching2 = Hungarian(NormalizedObservation1, NormalizedObservation2)
            Delaunay1 = CalculateDelaunayNetwork(NormalizedObservation1)
            Delaunay2 = CalculateDelaunayNetwork(NormalizedObservation2)
            int TotalDistance = 0
            for z ∈ 1,2,..., 10 do
                TotalDistance += PlayerDifference(Delaunay1, Delaunay2, Matching1[z], Matching2[z])
            DistanceMatrix[i, j] = TotalDistance
Clusters = AgglomarativeClustering(NumberOfClusters, DistanceMatrix)
```

Each observation is normalized and then the Delaunay Network is calculated according to section 4.2.1. To compare two observations, the Hungarian matching is applied based on Müller-Budack et al. (2019)[4]. This is done based on the Euclidean distance between the normalized positions. Then, the two Delaunay Networks can be compared, since the Hungarian algorithm gives the matching for which players need to be compared. Thus, a total distance between observation 1 and observation 2 can be calculated. This process is repeated between each observation pair as proposed in the original algorithm. In the last step, one can give the DistanceMatrix and NumberOfClusters as input for the AgglomarativeClustering method provided by Sci-kit.

Another solution would be to use a specific algorithm to match $A(t)$ and $A(t')$ where $t \neq t'$. For this I propose a local search algorithm that tries to find the best matching between A_1 and A_2. For the Hungary algorithm Shaw and Glickman (2019) argue that the most optimal distance between two observations would be the minimal distance of player pairs. In this situation, I would argue a similar thing for the adjacency matrices. If $D_{tt'}$ is minimal, the player pairing between $A(t)$ and $A(t')$ could be optimal. However, trying all the pos-

[4]This equation is simplified to just calculate the Euclidean distance between the players

sible swaps would be very computationally expensive. Therefore, I propose the best improvement search with a swap. This works in the following manner:

Algorithm 5 Player matching local search

1: Calculate $D(tt')$ for $A(t)$ and $A(t')$.
2: Swap all player pairs once in $A(t')$.
3: Choose the swap that has the steepest decrease Δ in $D(tt')$ and replace the current solution.
4: If no improvement was found, return the best solution at that moment. This could be the original $A(t')$.
5: Continue step 2 - 4 until no improvement is made.

To ensure the quality of this solution, this Player Matching algorithm is only used when comparing two solutions where the ordered player set of $A(t) \neq$ to the ordered player set of $A(t')$. For example, when $A(t)$ and $A(t')$ are from different matches. When $A(t)$ and $A(t')$ are from the same match and team, but the substitution occurs, we could limit the number of swaps to the number of position/player changes. Table 1 and 2 express such a situation. They are pseudo columns from one game.

Index	0	1	2	3	4	5	6	7	8	9
Player name	Blind	Pasveer	Timber	Martinez	Mazraoui	Haller	Anthony	Gravenberch	Tadic	Alvarez
Player number	17	1	2	21	12	22	11	8	10	4

Index	0	1	2	3	4	5	6	7	8	9
Player name	Blind	Pasveer	Timber	Martinez	Danilo	Anthony	Mazraoui	Gravenberch	Tadic	Alvarez
Player number	17	1	2	21	9	11	12	8	10	4

In table 1 Ajax plays with their first ten. At some point, Sebastian Haller is substituted for Danilo (both striker). Normally, one would argue that this would allow for one swap. But now, Noussair Mazraoui (index four) is in index six in table 2. Anthony is in index five instead of six. Therefore, to achieve the most optimal comparison, two swaps are needed. However, if Danilo had been a center-back, he might have swapped position with someone else. Thus, the number of indices where table 1 and table 2 do not match are allowed as swaps. In this case three.

4.3 Method proposed by Müller-Budack et al. (2019)

The last algorithm in this section is the one proposed by Müller-Budack et al. (2019). Contrary to the two other proposed methods, this is a classification algorithm. Classification algorithms have some benefits over cluster algorithms, as the latter can be difficult to test on metrics like accuracy and recall. Also, the user does not have to interpret the different clusters. The method of Müller-Budack et al. (2019) is based on the idea that each formation played by a team

has, in theory, a perfect formation counterpart. The authors call these formations the Ground-Truth formations. A formation is then compared to these Ground-Truth formations. This gives a similarity measure between the observation and the Ground-Truth formations. The formation that has the highest similarity is likely to be the formation that is played in that observation. A detailed account of this algorithm will now follow.

First, all the coordinates of an observation are normalized in regard to the width ($x \in [0.0, 0.7]$) and length ($y \in [0.0, 1.0]$). This 0.7 range was chosen to preserve the aspect ratio of the field. The length of a Dutch pitch is 105 meters, and the width 69 meters, in some leagues, however, the aspect ratio is slightly different. The authors do not specify from which league their data is, but it is important to note the aspect ratio of the field the tracking data is from. All the coordinates are transformed s.t. every game is played from top to bottom. These coordinates are put together into different observations of possession or dispossession and are called a game segment, $S = \{f_i, \ldots, f_{i+m}\}$ in which f is the formation played in an observation of a game of length m. To normalize the players' positions, the team center of all individual player positions is subtracted from the players' positions at each time frame. This then gives a mean position for every role $\bar{r}$ during the observed match segment and leads to a formation being defined as $F = \{\bar{r}_1, \ldots, \bar{r}_{10}\}$. This is referred to as Visual Formation Summary (VFS). It is important to note that this process is done for the Ground Truth as well as for a game segment. Formations can have a different compactness but considered the same formation. E.g., a 4-4-2 played narrowly or very wide and stretched. To deal with this variety in compactness the coordinates of a role are once again normalized with regard to the minimum and maximum x and y coordinate within a formation F:

$$\tilde{r} = \frac{\bar{r} - \min(F)}{\max(F) - \min(F)} \quad \forall \bar{r} \in F \tag{22}$$

From this equation it does not become apparent that we are dealing with vectors of x, y but this seems the most logical interpretation. Thus, for both x and y the minimum and maximum are found. Now that each role has a normalized x, y, two formations can be compared. This leads to a similarity matrix $M(F_1, F_2) \in \mathbf{R}^{10\times 10}$ between two formations. The distance metric in the similarity matrix is the Euclidean squared distance. The distance is squared in order to penalize smaller distances between different roles less severely. This is because the algorithm is dealing with normalized coordinates that will be within the range of $0 - 1$. For each entry, $m_{i,j}$ the Euclidean distance is then calculated by comparing $\tilde{r}_i = (\tilde{x}_i, \tilde{y}_i)$ of F_1 to $\tilde{r}_j = (\tilde{x}_j, \tilde{y}_j)$ of F_2.

$$M_{i,j}(F_1, F_2) = \max\left(1 - \frac{\left\|\tilde{r}_i - \tilde{r}_j\right\|_2^2}{\delta}, 0\right) \quad \forall \tilde{r}_i \in F_1; \tilde{r}_j, \in F_2. \tag{23}$$

Note that if two roles are very similar, $\left\|\tilde{r}_i - \tilde{r}_j\right\|_2^2$ will be close to zero. Therefore, $max(1 - \left\|\tilde{r}_i - \tilde{r}_j\right\|_2^2, 0)$ will approximate 1. $\delta \in [0, 1)$ is added as a tolerance radius. It ensures that the similarity between two players decreases and thus 0 will be approached sooner. According to Müller-Budack et al. (2019) a pitch can generally be divided into three horizontal and vertical groups: left, center, right and defender, midfielder, attacker, respectively. According to Müller-Budack et al. (2019) a normalization factor of $\delta = \frac{1}{3}$ would ensure that similarity of wingers to central players would be 0. Since a winger would have a one-third (of the normalized formations) to the central midfielder, as well as horizontally (from left to center e.g.) and one-third vertically (from attacker to central midfielder). However, it is easy to note that if $x = \frac{\frac{1}{3}^2 + \frac{1}{3}^2}{\frac{1}{3}}$, then $x < 1$ and the max of $(1 - x, 0)$ can then never be 0. The distance is a little larger than 0, still the point stands: δ makes sure that similarities higher than 0 are only appreciated when two players have similar positions. Now that $M_{i,j}(F_1, F_2)$ is calculated, there is a similar problem as mentioned in Shaw and Glickman (2019). Which $\tilde{r}$ from F_1 is compared to which $\tilde{r}$ from F_2, since it has little use to compare a center-back to a striker. As shown earlier, Müller-Budack et al. (2019) also use the Hungarian algorithm to solve this issue. It is calculated using

$$m_{i,j}^* = \begin{cases} m_{i,j} & \text{, if } \tilde{r}_i \in F_1 \text{ is assigned to } \tilde{r}_j \in F_2, \\ 0 & \text{otherwise.} \end{cases} \tag{24}$$

The total similarity between F_1 and F_2 is then the sum of the assignment where $\tilde{r}_i \in F_1$ is assigned to $\tilde{r}_j \in F_2$. Please note that the Hungarian algorithm here serves to maximize the similarity between F_1 and F_2 contrary to the Shaw and Glickman (2019) algorithm that was trying to minimize the distance between two observations.

In the end, a game segment S can thus be compared to a set of ideal formations called Ground-truth templates, $T = \left\{\hat{F}_1, \ldots, \hat{F}_t\right\}$. This makes it possible to create a ranking of the most probable formations played in a game segment S:

$$F_n^* = \underset{\hat{F} \in T}{\text{argmax}}(\text{FSIM}(F, \hat{F})) \tag{25}$$

Note that the Ground-Truth formations T were set up by domain experts. These templates are listed in Figure 6.

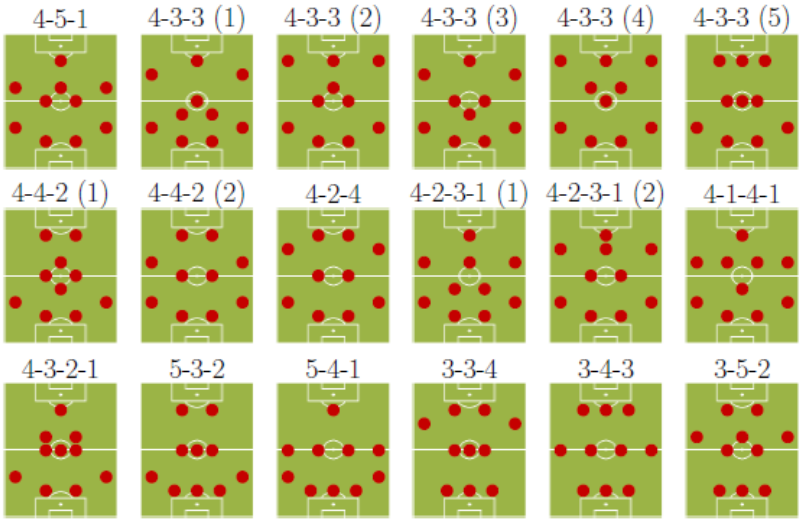


Figure 6: The templates with idealized role positions for twelve formations as defined by Müller-Budack et al. (2019). It is assumed that Müller-Budack et al. (2019) probably take the different versions of 4-3-3, 4-2-3-1 and 4-4-2 together as otherwise it would result in eighteen formations.

4.3.1 Limitations

There are a few limitations to this algorithm. First, the whole algorithm is build around the Ground-Truth formations. However, as mentioned in the introduction, football is a dynamic and fluid game and teams rarely play as the coach planned before. Furthermore, in their results section, the authors claim a top-1 accuracy of 'only' 20%. Their top-3 accuracy already sits at 71% and this could be an indication that their algorithm does not work as bad as it may seem at first. Since formations are very ambiguous, classifying 4-3-3 as a 4-2-3-1 might not be a considerable issue for the user. The similarity matrix of Figure 7 also shows that some formations are very similar. Furthermore, though equation 23 might be easy for interpretation as it squishes a values between 0 and 1, a downside is that information is lost. Without squishing the values between 0 and 1 the similarity (perhaps rather dissimilarity) could become negative.

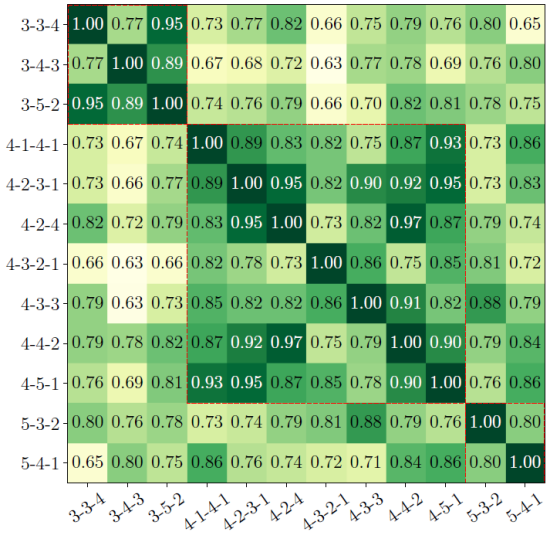


Figure 7: Similarity between formations as calculated by Müller-Budack et al. (2019).

4.4 Short summary of the algorithms

Since this research is comparing three different algorithms and a lot of extension are proposed, I present an overview for the three algorithms, so that this can be used whenever the reader loses sight of the properties of the different algorithms. Table 1 presents the overview in a table. There are seven different properties that are listed of the algorithms. Note that this table has the proposed extensions in them, thus in the case of Shaw and Glickman (2019) and Narizuka and Yamazaki (2019) they do not refer to the base algorithm. For example, the original algorithm by Narizuka and Yamazaki (2019) does not have a role matching algorithm. Moreover, Shaw and Glickman (2019) only propose an argmax k for the scaling. The citations in the column 'Scaling' refer to the scaling method proposed by the other two papers. In the Appendix (7) you can also find an overview with a description of all the important symbols used throughout this research.

	Cluster / Classification	**Cluster method**	**Classification method**	**Distance measure**
Shaw & Glickman (2019)	Both	Hierarchical clustering	Bayesian / smallest distance	Wasserstein
Narizuka & Yamazaki (2019)	Cluster	Hierarchical clustering	NA	Delaunay
Müller-Budack et al. (2019)	Classification	NA	Similarity	Euclidean
	Relative positions	**Role matching**	**Scaling**	**Section**
Shaw & Glickman (2019)	Nearest neighbor	Hungarian	A priori k, argmax k, Narizuka & Yamazaki (2019), Müller-Budack et al. (2019)	Section 4.1
Narizuka & Yamazaki (2019)	Center of mass	Hungarian / local search	Standard deviation of average player position	Section 4.2
Müller-Budack et al. (2019)	Center of mass	Hungarian	Minimum and maximum coordinate within formation	Section 4.3

Table 1: Properties of the different algorithms.

5 Method & Experimental set-up

To determine whether we can accurately classify or cluster formations using geometric pattern algorithms, certain experiments will have to be run. For the algorithms that require a training set, a dataset of 100 Eredivisie games is created. This dataset is obtained from Chreonhego tracking data, which tracks player data at 25Hz, resulting in large raw data files. Observations with less than 10 field players are deleted [5].

All the data is smoothed using the Kalman filter. The Kalman filter uses a series of measurements observed over time. This series of measurements includes statistical noise and other inaccuracies, which the tracking data also has. It then produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone. It achieves this by estimating a joint probability distribution over the variables for each time frame (Welch, Bishop, et al., 1995). All this means that the tracking data is smoothed, by using the expected position based on the previous position and velocity. Schoonderbeek (2021) showed that this decreased unrealistic accelerations from 2.9% to 0.050% on a similar dataset.

For these experiments, a test set was constructed consisting of the formations of 70 Eredivisie games. The Wyscout match reports were used for this as a test set. These match reports contain the formations during certain periods (Figure 8). I wrote a script that reads these pdfs and reads the positional data based on the specific periods of time a certain formation is played. In total, this resulted in 380 observations. The main benefit of using the Wyscout match reports as labels was that this saved a lot of time. Otherwise, these 70 games would have to be watched and annotated. However, using a test set like this has disadvantages. First, their classifications are not always correct. At Ajax AFC, Wyscout is in practice often used as an indication, but not as the actual truth. Secondly, Wyscout does not make a distinction between in-possession and out-of-possession formations. This is a substantial limitation since often times teams play differently when attacking compared to defending. However, in-possession formations are not of a different nature than out-of-possession formations, nor are formations that combine in and out-of-possession. Teams will still play formations like 4-3-3 or 4-4-2. This means that if the algorithms are accurate with the Wyscout test set, they should have little problems with in- and out-of-possession formations. The last issue with the Wyscout match reports is that they tag fewer formation options. Wyscout distinguishes between nine formations, whereas Müller-Budack et al. (2019) distinguishes twelve different formations. Shaw and Glickman (2019) even consider twenty different formations.

[5] Can happen when a player is sent off or if a player is injured and maximum number of substitutions was already made.

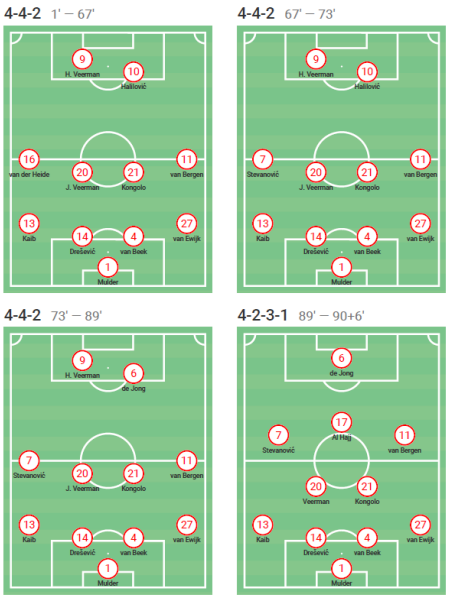


Figure 8: Different formations during a game as presented in a Wyscout file.

The test set will be used to compare the algorithms by Shaw and Glickman (2019) and Müller-Budack et al. (2019) as these both have classification components. The algorithm proposed by Narizuka and Yamazaki (2019) does not have a classification component and can only cluster single games. To compare this algorithm to the other two, I have proposed the local search and Hungarian extension (as explained in 4.2). By comparing the algorithms, I can answer RQ2: 'How do the algorithms compare against each other?'. For the algorithm by Shaw and Glickman (2019) , I have proposed certain modifications in section 4.1 based on methods used by Müller-Budack et al. (2019) and Narizuka and Yamazaki (2019). I will compare these modifications to the base algorithm to answer RQ3: 'How can elements from these algorithms be combined into a better new algorithm?'

I will report on the key metrics: accuracy, precision and recall. However, these metrics are not sufficient for formations. If the algorithm classifies a 4-3-3 as a 4-2-3-1 this is not as wrong as a classifying a 4-3-3 as 5-3-2. Therefore, I will also propose to use the similarity matrix of Müller-Budack et al. (2019) (Figure 7) as a measurement of how wrong the classification was, using the following equation:

$$SA = \frac{1}{N} * \sum_{n=1}^{N} \begin{cases} Sim(P_n, C_n) * \beta & \text{if } P_n \neq C_n \\ 1 & \text{otherwise.} \end{cases} \tag{26}$$

In which SA refers to the smoothed accuracy. For each observation, we calcu-

late a correctness score. If the predicted class is correct, 1 is assigned just like normally for accuracy. However, we do not assign 0 if the prediction is not correct. Rather, the similarity between P_n (prediction for observation n) and C_n (actual class for observation n) is calculated. The similarity scores are rather high (between 0.67 and 0.90) for $Sim(P_n \neq C_n)$, which would always result in a high accuracy. Therefore, the Sim score is multiplied by β with $\beta = 0.5$. Note that the similarity table (Figure 7) for Shaw and Glickman (2019) will be recalculated as the final clusters are different from the formations in Figure 6. This is because Shaw and Glickman (2019) calculate their own clusters, whereas Müller-Budack et al. (2019) use the predefined Ground-truth formations. The last important metric for these algorithms is speed. Two of the three algorithms use a training set. Over this training set a distance matrix needs to be calculated ($O(n^2)$ complexity), which is done on large data files. Therefore, time is a constraint of these algorithms and thus calculation time will be taken into account and reported on.

Since the Wyscout data set has its limitations, I will also look at individual games. Together with the domain experts at Ajax, we will make a case study of a game in which they saw a formation change they would like the algorithms to pick up. Here, we will distinguish between in- and out-of-possession, as well as the phase in which the formation is played. Phases refer to the position of the ball. For the phases, the field will generally be divided into three or four strips.

Lastly, there will be a section that focuses on the implementation and answers RQ4: 'How can this research be implemented in the workflow of domain experts?'. For this, I have created a way to convert the formation predictions into an XML that can be used in Angles (a video tool used by domain experts). It also shows some figures of how these algorithms can be used for further data analysis. This further data analysis explains how these algorithms can be used on multiple games for tactics analysis. For example, the most effective formation in defense can be found. Or what passing option is most used in what formation.

All the experiments were run using Dell XPS (I7-8750H @2.2GHz, 16gb RAM). The code was all written in python, but optimized by heavily relying on Numpy for data manipulation and numba for further optimization.

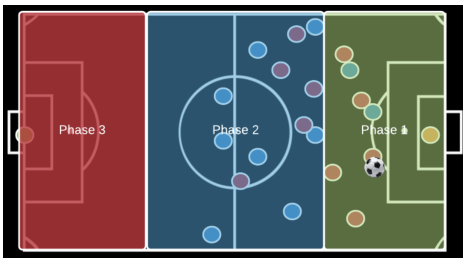


Figure 9: The three phases of play used at Ajax AFC.

6 Results

The result section deals with two parts of this research. The first part goes into detail about how well the various algorithms work, how they compare against each other and how well the proposed extension work (RQ1-RQ3). Secondly it will be discussed how these algorithms can be used by domain experts that work in football (RQ4). Here, I make a distinction between video analysis and data analysis in football. The video analysis shows how these algorithms can be used with the combination of footage. The data analysis hints at how these algorithms can be used to analyze a team's performance and playing style over multiple games.

6.1 Shaw and Glickman (2019)

This section will go into the various different options that can be applied within the framework of the proposed method by Shaw and Glickman (2019). These vary from different cluster methods, to different normalizing methods and distinct classification options. The following paragraph assumes that first a distance matrix between each single observation is calculated. This took 73 hours.

6.1.1 Clustering

In accordance with Shaw and Glickman (2019) the number of clusters is set at twenty. In section 4.1.5, I had argued that Ward's linkage is not a suitable method for clustering Wasserstein distances, as the assumptions expect Euclidean distances instead of Wasserstein distances. In fact, the sklearn package will error if you specify distances are not Euclidean and Ward's linkage is used. However, sklearn can be tricked into 'thinking' the distances are Euclidean, by specifying that the distances are in fact Euclidean. In theory, it makes no sense to use Euclidean distances. Ward's linkage relies on the inertia. This quantifies the sum of squared residuals between the reduced signal and the initial signal: it is a measure of the variance of the error in an l2 (Euclidean) sense. However, there seems to be evidence that Ward's linkage can be applied to other distance measures, such as the Manhattan (Strauss & von Maltitz, 2017). The

most substantial advantage of using Ward's linkage is that it keeps the number of observation within a cluster similar.

Table 2 gives an overview of the different distance measurements statistics on the cluster sizes. Here it becomes clear that single linkage is not usable as it makes one large cluster. The same goes for complete and average linkage, though the largest clusters are smaller. Also, if the custom update function (see Algorithm 2) is used, one cluster becomes too large. To make this method usable and thus comparable to Ward's linkage we apply a penalty based on the cluster size

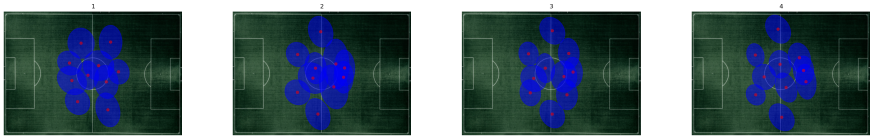
$$d = d * \frac{cs}{to} \tag{27}$$

in which d, the distance, is multiplied by the ratio of the new cluster size (cs): the size of two clusters to be merged and to the total number of observations. From Table 2 it also becomes apparent that the execution time for the custom update function is a lot slower. Even though introducing the distance penalty, helps with a better cluster size distribution it is still worse than Ward's linkage and a lot slower.

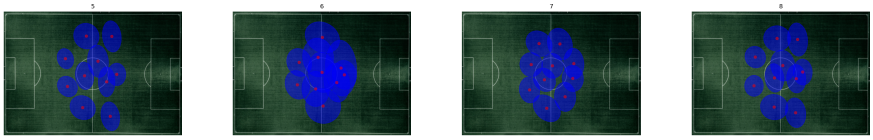
Algorithm	Smallest cluster in %	Median cluster %	Largest cluster in %	Sd	Execution time in m
Ward's linkage	0.32	4.97	11.94	58.75	1.06
Average	0.05	0.21	43.59	227.77	1.06
Maximum / Complete	0.05	1.08	19.90	129.62	1.19
Minimum / Single	0.05	0.05	98.02	420.19	5.09
Update Wasserstein	0.05	0.05	94.02	400.12	2050
Update Wasserstein distance penalty	0.22	3.2	53.3	72.12	2120

Table 2: Cluster size distribution.

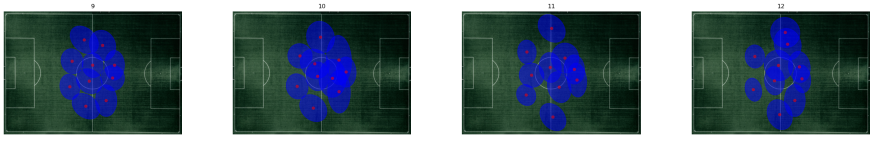
Figure 10 shows the 20 clusters that are obtained when using Ward's linkage. At first, the clusters seem accurate: the Hungarian algorithm seems to have made proper role assignments. Except for cluster 14 there are no clusters in which the merging produced inaccurate clusters. All the other clusters represent valid formations. There are, however, many very similar clusters. For example, cluster 5 and 8 are both 4-2-3-1's with a slight variation. This makes sense, since in accordance with the Wyscout testset a lot of Dutch teams play the 4-2-3-1 formation. To test this algorithm on the test set, some handwork needs to be done, as we need the same nine clusters as provided in the Wyscout test set. Clusters 8, 10, 15, 17, 18, 19 are merged into Cluster 4-2-3-1. Cluster 1 and 5 are merged into 4-3-3. Cluster 7,9, 12 and 16 are merged into 4-4-2. Cluster 11 is used for 5-3-2, 13 for 5-4-1 and 3 for 3-4-3. Cluster 2, 4 and 14 are omitted. Cluster 2 and 4 are formations that do not occur in the dataset (4-2-2-2 and 5-2-3) and Cluster 14 is not an actual formation. 4-2-3-1 was slightly adjusted to obtain a 4-5-1, whereas 5-3-2 was used to obtain a 3-5-2 and 4-4-2 to obtain a 4-1-2-1-2. These are artificial clusters.



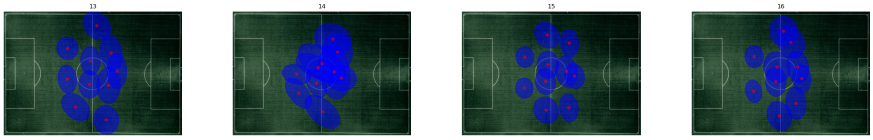
(a) Cluster 1. (b) Cluster 2. (c) Cluster 3. (d) Cluster 4.



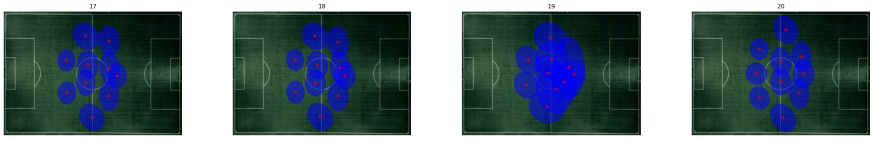
(e) Cluster 5. (f) Cluster 6. (g) Cluster 7. (h) Cluster 8.



(i) Cluster 9. (j) Cluster 10. (k) Cluster 11. (l) Cluster 12.



(m) Cluster 13. (n) Cluster 14. (o) Cluster 15. (p) Cluster 16.



(q) Cluster 17. (r) Cluster 18. (s) Cluster 19. (t) Cluster 20.

Figure 10: Clusters using Ward's linkage: playing from left to right.

6.1.2 Classification and parameter testing

Two classification methods were proposed. One (Bayesian classifier) was proposed in the original paper by Shaw and Glickman (2019). In algorithm 3, I proposed a different classification method based on the distance of a new observation to each cluster.

There were various parameters that were tested in both classification methods. The first parameter tested was the scaling factor. In the original paper, an argmax over the scaling factor k was introduced. I noted that because of the argmax aspect this method was expensive: for each k the relative positions needed to be scaled, the means and co-variances needed to be recalculated and the Hungarian algorithm needed to be reapplied. For the classification methods, I proposed three different scaling options: the one proposed by Müller-Budack et al. (2019), a simple scaling formula that only needs to be calculated once (equation 16) and the option as presented in Narizuka and Yamazaki (2019). Moreover, I proposed a scaling method for the co-variances (equation 14 and 15, algorithm by Shaw and Glickman (2019)).

In section 5, I argued that some formations are very similar to each other. For example, should the algorithm predict a 3-5-2 when the true formation is 5-3-2, the mistake can be considered small. Thus, a smoothed accuracy was proposed based on the similarity score between formations. Table 3 shows the similarity between each formation. δ was set to 0.1 instead of 0.3 to increase the differences between formations. (see equation 23, algorithm by Müller-Budack et al. (2019)). With δ set to 0.3, it was found that the similarity scores were all around 0.9, which would result in a distorted picture, with very high accuracies.

	5-4-1	**5-3-2**	**3-5-2**	**3-4-3**	**4-1-2-1-2**	**4-2-2-2 (4-4-2 variant)**	**4-5-1**	**4-2-3-1**	**4-3-3**
5-4-1	1								
5-3-2	0.90	1							
3-5-2	0.79	0.85	1						
3-4-3	0.77	0.75	0.74	1					
4-1-2-1-2	0.81	0.79	0.79	0.81	1				
4-2-2-2 (4-4-2 variant)	0.80	0.79	0.67	0.81	0.91	1			
4-5-1	0.75	0.76	0.79	0.73	0.88	0.82	1		
4-2-3-1	0.81	0.70	0.77	0.75	0.85	0.87	0.83	1	
4-3-3	0.74	0.71	0.79	0.82	0.83	0.78	0.83	0.85	1

Table 3: The similarity between formations based on the Formula proposed by Müller-Budack et al. (2019).

Table 4 shows the overall results of the distance classification method. The best scaling method was the one proposed by Narizuka and Yamazaki (2019) with 42% accuracy and 65% smoothed accuracy, yet this scaling method was also the slowest (22120 seconds). Scaling the co-variances consistently outperformed not scaling them. The original method for normalizing positions proposed by Shaw and Glickman (2019) was outperformed by both Müller-Budack et al. (2019) and Narizuka and Yamazaki (2019).

	Scaling	Scaling co-variances	Accuracy	Precision	Recall	Smoothed accuracy	Time (in s)
Distance classifier	a priori k	true	14%	14%	13%	45%	266
Distance classifier	a priori k	false	12%	12%	12%	43%	240
Distance classifier	Shaw and Glickman (2019)	true	28%	29%	27%	56%	19715
Distance classifier	Shaw and Glickman (2019)	false	27%	28%	27%	54%	18721
Distance classifier	Müller-Budack et al. (2019)	true	32%	33%	32%	60%	6425
Distance classifier	Müller-Budack et al. (2019)	false	31%	30%	31%	58%	5212
Distance classifier	Narizuka and Yamazaki (2019)	true	42%	42%	42%	65%	22120
Distance classifier	Narizuka and Yamazaki (2019)	false	40%	38%	41%	61%	21980

Table 4: The accuracy of distance classification and the different normalizer methods.

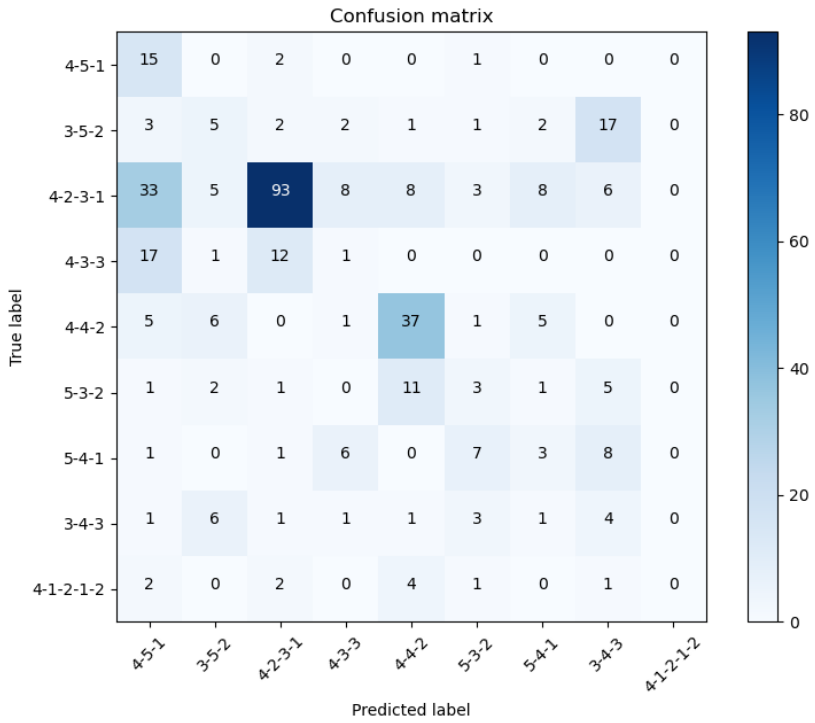


Figure 11: Confusion matrix best performing distance classifier.

Figure 11 shows the confusion matrix of the best performing algorithm. 4-2-3-1 was the dominant observation and it was also predicted the most. Often, formations with a similar counter-part got intertwined. For example, when the true label was 3-5-2, 3-4-3 (a similar formation) was often predicted instead. This was reflected in the higher smoothed accuracy of 65%. Furthermore, it is interesting to note that 4-1-2-1-2 was never predicted.

Table 5 shows the results of the Bayesian classifier. The Bayesian classifier seemed to outperform the distance classifier. The highest accuracy (45%) was achieved when the normalizing method by Narizuka and Yamazaki (2019) was

used, the smoothed accuracy was 66%.

	Scaling	Scaling co-variances	Accuracy	Precision	Recall	Smoothed accuracy	Time (in s)
Bayesian classifier	a priori k	True	15%	15%	15%	45%	380
Bayesian classifier	a priori k	False	15%	15%	15%	45%	280
Bayesian classifier	Shaw and Glickman (2019)	True	40%	39%	41%	62%	18242
Bayesian classifier	Shaw and Glickman (2019)	False	39%	39%	39%	60%	18042
Bayesian classifier	Müller-Budack et al. (2019)	True	42%	42%	42%	64%	4392
Bayesian classifier	Müller-Budack et al. (2019)	False	33%	33%	33%	57%	4300
Bayesian classifier	Narizuka and Yamazaki (2019)	True	45%	44%	44%	66%	21855
Bayesian classifier	Narizuka and Yamazaki (2019)	False	40%	40%	40$	63%	21559

Table 5: Bayesian classifier and different normalizers results.

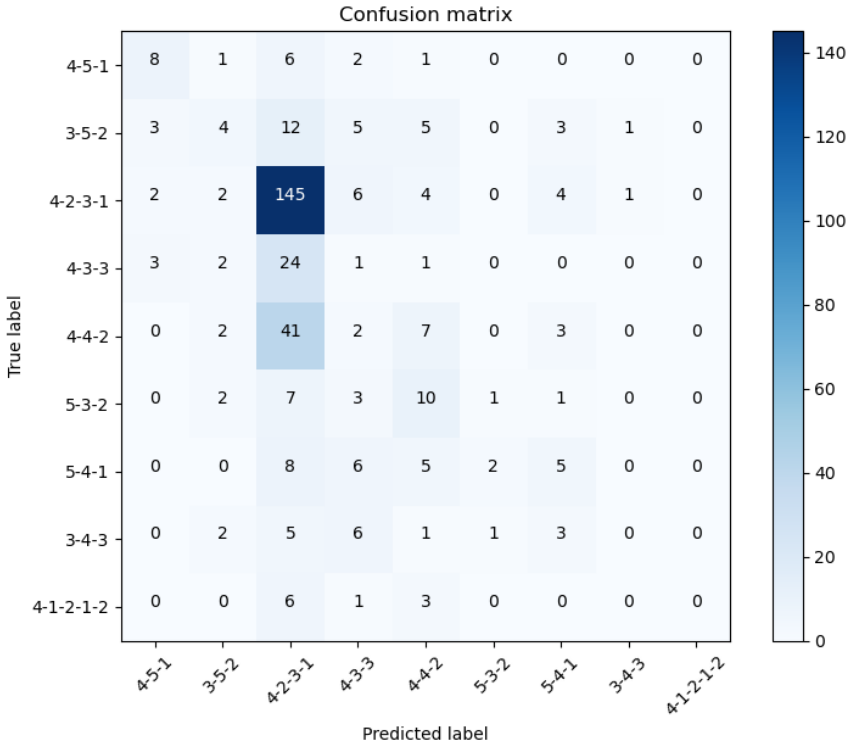


Figure 12: Confusion matrix best performing Bayesian classifier

Figure 12 shows the confusion matrix of the best performing Bayesian classifier. Though the accuracy was higher, it seems that this was merely because it predicted 4-2-3-1 more often. In other categories, it looks like the algorithm made more mistakes. To further examine this, Table 6 shows the recall and precision per class from both algorithms. For precision, the distance classifier was higher in four classes, the Bayesian classifier was higher in four classes as well. When it comes to recall, the distance classifier scored higher in five classes and the Bayesian classifier in two classes. Thus, this somewhat confirmed that the

hypothesis that the high accuracy of the Bayesian classifier is mainly due to it predicting 4-2-3-1 so often.

	4-5-1	3-5-2	4-2-3-1	4-3-3	4-4-2	5-3-2	5-4-1	3-4-3	4-1-2-1-2
Bayesian classifier precision	50%	27%	57%	3%	19%	25%	26%	0%	nan
Distance classifier precision	19%	20%	79%	5%	60%	15%	15%	41%	nan
Bayesian classifier recall	44%	12%	88%	3%	13%	2%	19%	0%	0%
Distance classifier recall	83%	15%	57%	3%	67%	13%	12%	22%	0%

Table 6: Recall and precision per class for best performing algorithms.

6.1.3 Müller-Budack et al. (2019)

Since not all classes from the paper written by Müller-Budack et al. (2019) were used by Wyscout, three of the ground-truth labels were not taken into account (4-1-4-1, 4-2-4, 3-3-4). 4-4-2 (2) is considered as 4-1-2-1-2 in the Wyscout reports. The authors of this method had previously reported a 20% accuracy. The results for this test set were worse, with 13% accuracy (43% smoothed accuracy) and 12 and 14% recall and precision respectively. Figure 13 shows the confusion matrix. The dominant class 4-2-3-1 was not predicted often. The algorithm predicted a wide variety of classes, but not often correct.

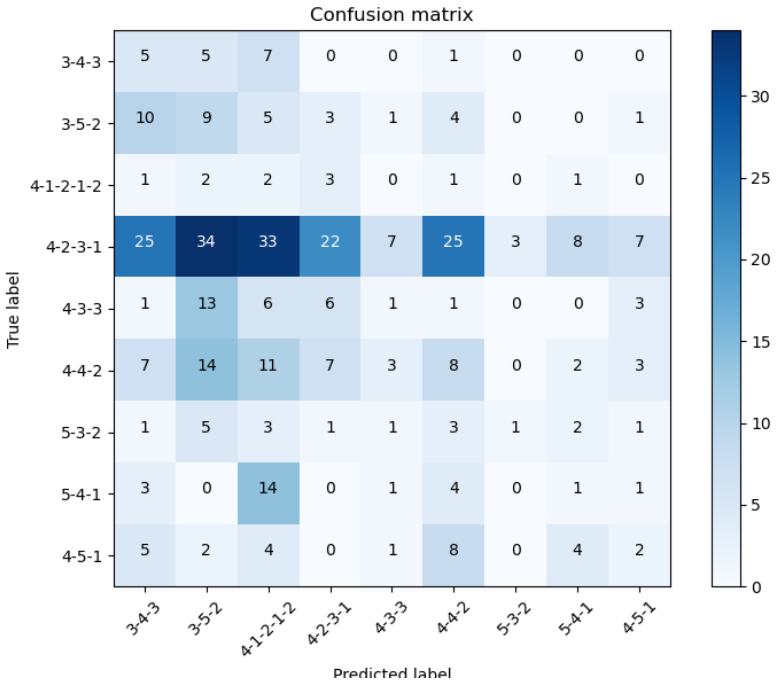


Figure 13: Confusion matrix algorithm by Müller-Budack et al. (2019).

6.1.4 Narizuka and Yamazaki (2019)

For the algorithm proposed by Narizuka and Yamazaki (2019), only one single game was considered. As mentioned in section 4.2 this method could not be used for multiple games. In fact, the base method only allows for a game up until substitutions. In the same section, two extensions were proposed to allow for full-game or even multi-game analysis. Due to the nature of the algorithm, as well as the computationally expensive extensions, only a full-game analysis will be presented in this section. The data was down sampled to 1Hz instead of 25Hz, as otherwise it would become too computationally expensive.

The game for which this was tested is PSV - Ajax (23-01-2022) in which the analysts of Ajax noticed that the defensive midfielder of PSV (Gutiérrez) dropped deep in the first phase attack. Because of the precision of this algorithm (calculating an observation for each frame), I expected it might pick up on these small positional changes.

When the Hungarian algorithm was used, the dendrogram in Figure 14 was obtained. Because the steepest increase occurred after two clusters, this was the number of clusters chosen. Figure 15 shows the two clusters. In cluster 1, Gutiérrez (number 15) dropped deep to become a third defender, in cluster 2 Gutiérrez stayed in his position as a defensive midfielder.

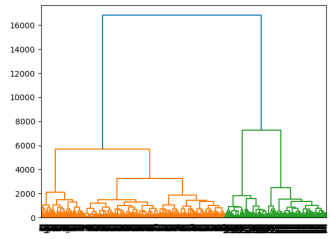
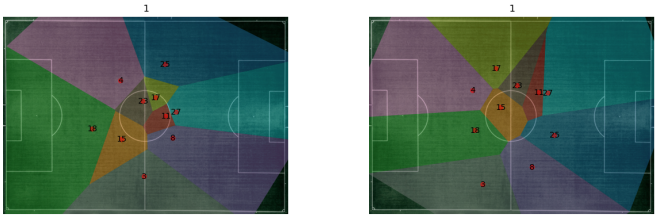


Figure 14: Dendrogram for phase 1 attacking with Hungarian algorithm.



(a) Average positions of cluster 1. (b) Average positions of cluster 2.

Figure 15: Two clusters in phase 1 attacking constructed with the Hungarian algorithm.

The local search extension was also tested, which resulted in a dendrogram very similar to the in Figure 14, as shown in Figure 16. The two clusters are shown in Figure 17. Again, the clusters show that Gutiérrez sometimes played as a holding midfielder and sometimes as a third centre-back. However, in Figure 15a he seemed to be playing more as a right centre-back whereas in Figure 17b he played between the two centre-backs.

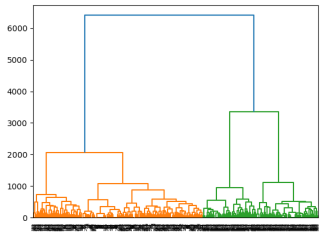
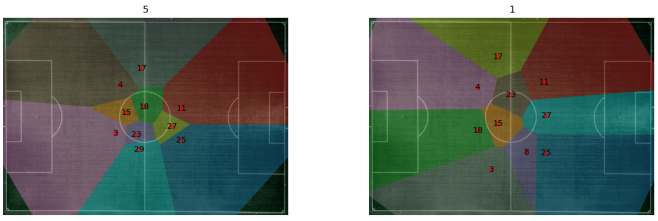


Figure 16: Dendrogram for phase 1 attacking with local search.



(a) Average positions of cluster 1. (b) Average positions of cluster 2.

Figure 17: Two clusters in phase 1 attacking constructed with local search.

Figure 18 shows the heatmap of Gutiérrez in phase 1. The darkest area of the heatmap appears to be around the position of a right centre-back, suggesting that the Hungarian algorithm captured the position change more accurately. However, Figure 19 shows some real images where Gutiérrez played as a centre-back and right centre-back, suggesting that both algorithms captured some correct aspect of his positioning.

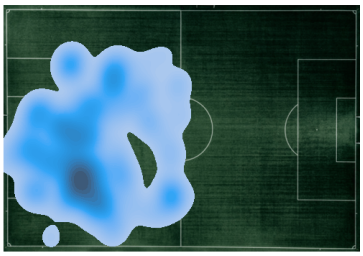


Figure 18: Heatmap of Gutiérrez in phase 1.

(a) Example of Gutiérrez dropping between the two centre-backs.

(b) Example of Gutiérrez dropping between the two centre-backs.

(c) Example of Gutiérrez dropping as a right centre-back.

(d) Example of Gutiérrez dropping as a right centre-back.

Figure 19: Images of Gutiérrez playing in the defense.

6.2 Implementation: how domain experts can use these algortihms

This section discusses how the various algorithms can be implemented. It investigates how this research can be used in video analysis as well as in data analysis. Here, implementation does not refer to how the algorithm should be implemented in a programming language, rather it refers to how domain experts such as video analysts, scouts, data analysts and coaches might make use of these algorithms in their daily jobs.

6.2.1 Case study

In this subsection of the results, a case study was conducted. Together with domain experts, we looked at one individual game, for both the home and away team. The game was FC Groningen - Ajax (02-04-2022) in which FC Groningen played a 5-3-2 or 5-4-1 in defense and a 4-2-2-2 and 4-2-3-1 in attack, whereas Ajax played a wide variation of formations including a 3-4-3, 4-2-3-1 and 4-3-3. This case study was done inside the Angles video tool, since this is the tool used for analysis by video analysts. I have edited the script in such a way that analysts could get formations from all the categories: formations in attack and defense and the three different phases (6 combinations in total). Analysts were able to fill in whether they want formations in phases, defense/attack or both. They could also choose per how many minutes they wanted the formations to be reevaluated (Figure 20).

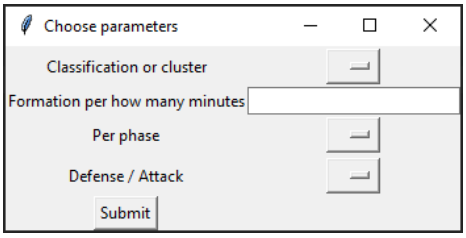


Figure 20: Menu for creating XML.

This tool will output an XML with tags of the formation and what time it was played. This could then be imported in Angles. Figure 21 shows what this looks in Angles. The analysts could scroll through the clips and the movie would jump to the clip where they played that specific formation.

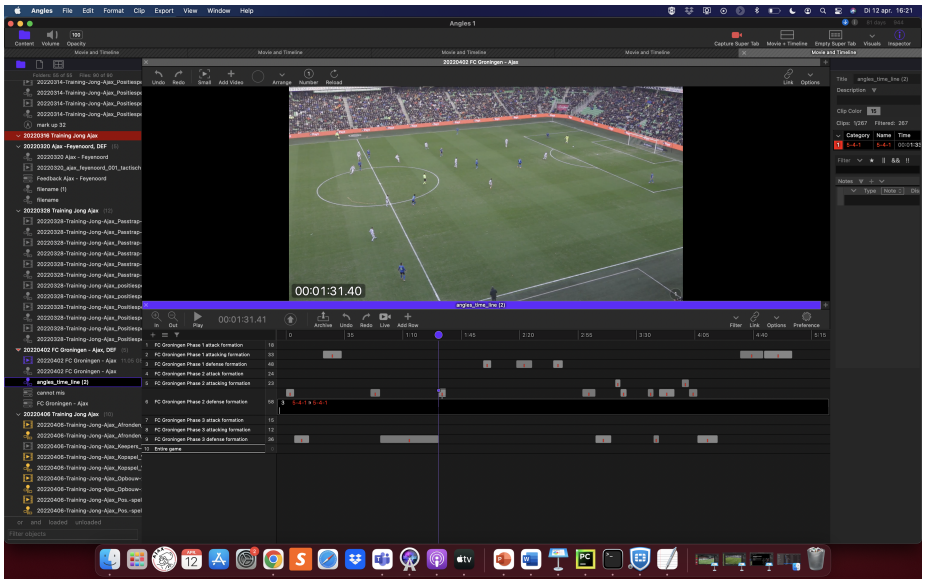


Figure 21: Screenshot from the proposed solution in Angles.

Two example snapshots of clips are shown in Figure 22. These are clips that were correctly marked.

(a) FC Groningen playing a 5-3-2, correctly classified by the Shaw & Glickman algorithm.

(b) FC Groningen playing a 4-2-2-2, correctly classified by the Shaw & Glickman algorithm.

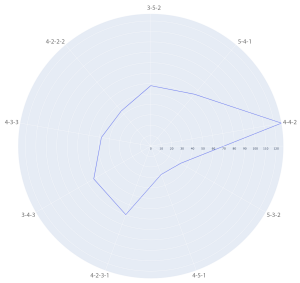
Figure 22: Two snapshots of fragments from FC Groningen vs Ajax (02-04-2022).

For both games, the best performing algorithm from Table 4 was used (Bayesian classifier, with normalization according to Narizuka and Yamazaki (2019) and the co-variances scaled). Formations were reevaluated every 3 minutes and were split up into defense, attack and three phases. If there was no fragment longer than 10 seconds in that category, it was omitted. For the FC Groningen - Ajax, this returned 48 labels for FC Groningen and 36 for Ajax. Out of these

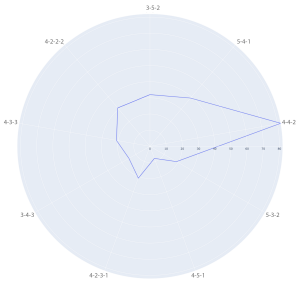
48 labels for FC Groningen, 40 were deemed correct, four were incorrect and for four there was a considerable similarity between the predicted and the true formation. Correctly classifying the formations of Ajax proved to be more difficult. Of the 36 labels for Ajax, 26 were deemed correct. For 5 there was a considerable similarity between the predicted and the true formation. 5 were incorrect.

6.2.2 Deeper analysis of formations

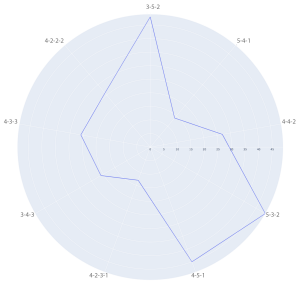
In this section, I will explain how these algorithms can be used for data analysis in football. It is discussed into more detail how domain experts can use these algorithms. Figure 23 shows the formations FC Groningen had been playing over the last 3 games. In defense, they played a wide variation of formations, but mostly relied on either a 3-4-3 or 4-2-3-1. In attack, they mostly relied on a 4-2-2-2 and 4-2-3-1 formation.



(a) General formation FC Groningen.



(b) Attacking formations FC Groningen



(c) Defensive formation FC Groningen.

Figure 23: Overview of formations played by FC Groningen (measured in minutes)

It is also interesting to measure the success of certain formations. Figure 24 shows the different formations FC Groningen applied against Ajax in attack. The blue line shows the formation and the red line shows how often the for-

mation was successful. For phase 1 and 2 the main goal is to get to the next phase, thus success for these phases is measured by whether the play is progressed to the next phase. In phase 3, teams want to score goals, thus success is measured by whether the team has shot on goal. The 5-4-1 formation was the most successful (9 out of 12 attempts), whereas the more frequent formations were less successful (11 out of 19 for 4-2-3-1 and 11 out of 22 for 4-2-2-2).

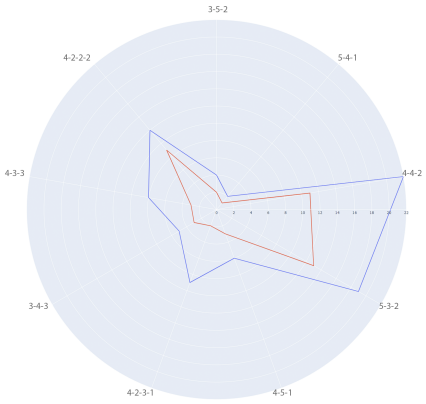
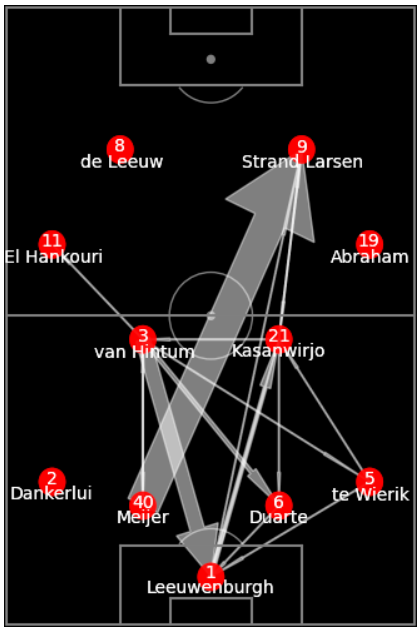
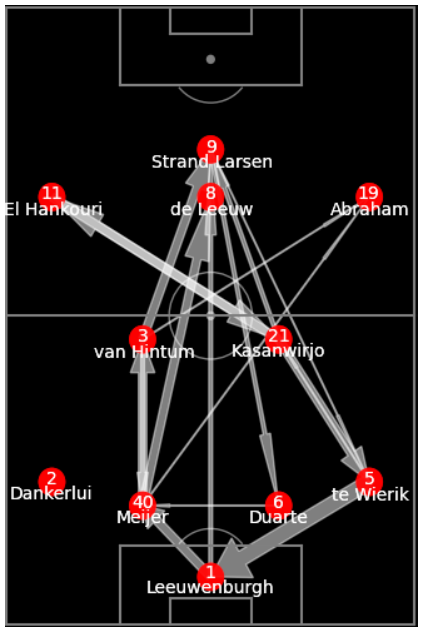


Figure 24: FC Groningen's formation in attack vs Ajax and their relative success.

The formation data can also be combined with other event data. Figure 25 shows the passing map of two different formations. The arrows indicate how often a pass was played, if the arrow is thicker the pass had been played more frequent. This allows coaches to see which player was important in what formation and perhaps on which player they should focus defending. In the 4-2-2-2 (Figure 25a) FC Groningen seem to choose a more direct route by often playing from the left center back (Meijer) straight to the striker (Strand Larsen). In the 4-2-3-1 the wings are used more often (a lot of build-up play via te Wierik and attacks via El Hankouri). The advantage of this type of analysis over the video analysis is that large quantities of matches can be analyzed by the computer. Therefore, the video analysts will not have to watch all the games, annotate whether a formation was successful. With these figures, a video analyst might conclude that FC Groningen often play the long ball in 4-4-2 and if he wants to prepare his team, he only has to search for videos where the long ball is played, rather than first watching countless games identifying the playing style that a team plays in certain formations.



(a) Passing map when FC Groningen attacks in a 4-2-2-2.



(b) Passing map when FC Groningen attacks in a 4-2-3-1

Figure 25: Passing maps for different formations (images are rotated 90 degrees, so names are readable).

7 Discussion

7.1 Findings summarized

The purpose of this study was to find whether existing geometrical pattern algorithms would be able to accurately cluster and classify football formations, how they compared against each other and whether the algorithms could be combined to increase their accuracy. In my introduction, I mentioned how a lot of academic work regarding football analysis, was not made with an eye for the domain expert. Therefore, this research also looked at how these results could be combined into the workflow of domain experts. In this study, I found that accurately clustering football formations is possible with both the method proposed by Shaw and Glickman (2019) and Narizuka and Yamazaki (2019). The clusters produced by the first method were realistic and reflected how teams play in the Eredivisie (often 4-2-3-1 with some other teams playing 3-5-2, 4-4-2 and 5-3-2). The clusters produced by Narizuka and Yamazaki (2019) can accurately model new formations in which position changes occur. Classifying new observations turned out to be more difficult. The base algorithm from Shaw and Glickman (2019) scored an accuracy of 29%, whereas the algorithm

proposed by Müller-Budack et al. (2019) was only 13% accurate. This is inconsistent with Müller-Budack et al. (2019), who reported an accuracy of 20%. Both accuracies are dwarfed by the 75% reported by Bialkowski et al. (2016). Combining elements from the various algorithms increased the accuracy to 45%. The best option turned out to be the Bayesian classifier, with the scaling proposed by Narizuka and Yamazaki (2019) and scaling the co-variances. In the case study, one game was analyzed: 66 out 84 formations were labelled correctly by the best algorithm, indicating that this 45% might not be a good representation of the accuracy.

7.2 Findings analyzed

7.2.1 Results framework by Shaw and Glickman (2019)

In the first section of the results, I showed that the proposed method by Shaw & Glickman (2019) works for clustering football formations. Figure 10 showed the results, in which multiple realistic football formations appeared. I also showed that the Ward's linkage, though not meant for non-Euclidean distances, worked better than other linkages. It was fast, and the clusters were more evenly distributed, which is consistent with findings by Sharma, Batra, et al. (2019). Shaw and Glickman (2019) failed to specify how Ward's linkage could be used to create these actual clusters, as it was not stated how these clusters should be merged. I proposed a viable solution how to obtain these clusters. This cluster method has several limitations, nonetheless. One was already mentioned in section 4.1.5. Since the clusters are obtained from a lot of matches and the Hungarian algorithm is bound to make some mistakes, the co-variances of clusters will be much larger than observations, which becomes a problem when classifying new observations, since there might be a bias towards a cluster with smaller co-variances. Shaw and Glickman (2019) had not yet reported on this problem in their original paper. I proposed to scale the co-variances according to equation 16, which consistently improved the results. There are at least two more limitations concerning this method. First, this clustering method is very slow. For 100 matches, it took over 73 hours to compute a distance matrix between each observation. This is due to the nature of the algorithm which runs in $O(N^2)$. Thus, for each new observation, it needs to be compared to all the other observations. Lastly, the cluster method is limited by the trainingset. Some formations might be less frequent, and this is worsened by the nature of clustering. Smaller clusters tend to be merged in larger clusters (Müllner, 2011). This means that formations that are not very frequent will not end up in the final clusters. One might increase the number of clusters, but as shown in Figure 10 this will result in many very similar clusters. For classifying the Wyscout testset, the highest accuracy was reached by the Bayesian classifier, combining the base algorithm of Shaw and Glickman (2019), replacing the k-scaling by the normalizing method proposed by Narizuka and Yamazaki (2019) and scaling the co-variances. This normalizing method seemed the most promising, as it uses a standard deviation of players

with regard to the center of mass. Using the standard deviation ensured that the distance is properly normalized, and thus it matters little if a formation is wide or narrow.

Some formations are very similar and thus some mistakes should weigh less than others. Therefore, I proposed a smoothed accuracy which better represented the actual accuracy. The best algorithm reached a smoothed accuracy of 62%. I believe that for future research, the smoothed accuracy should be used as a metric. This smoothed accuracy is still lower than the 75% found by Bialkowski et al. (2016). There are several possible explanations for this difference. First, the authors only use five classes compared to the nine used in this research. Moreover, the classes are more evenly distributed in their research. Also, the formations were hand annotated, resulting in a perhaps higher quality dataset. Lastly, Bialkowski et al. (2016) assumed that a formation was stable between one half, whereas some formations from the test set lasted only a few minutes. Sometimes it takes multiple minutes before a team is in a proper formation. I also reported on the time. The time can become important when one would like to analyze formations during the game. Recent developments have made it possible to access the tracking data live and thus time should become an important metric in future research, since this analysis can be done live instead of after a match. The best performing algorithm takes almost 1 minute per observation ($\frac{21855 \text{ seconds}}{380 \text{ fragments}} \approx 60$). That means there is a real possibility to do these analyses live. To combat the possible limitations of the Wyscout dataset mentioned earlier, I also tested a full game (FC Groningen vs Ajax) with domain experts. This had a couple of benefits, foremost the labels were labelled more accurate than the Wyscout dataset. Moreover, testing it with the domain experts gave them a feel of how these algorithms could be used in their workflow, which is an important aspect of this research. The analysts were very positive about the results, and the accuracy seemed much higher than the accuracy for the Wyscout testset (66/84 correct). This can be seen as a first proof that the best algorithm is powerful enough to be used by domain experts. The reasons for the higher accuracy is probably because the data was poorly classified by Wyscout. The formations of FC Groningen were classified more accurately than those of Ajax. This is a consequence of Ajax's players taking on several roles during a game. For example, a left-back will often push forward, resulting in a defense of three man. The clusters were not trained enough on these alternative formations.

7.2.2 Results framework by Narizuka and Yamazaki (2019)

The other existing cluster method that I researched was the one by Narizuka and Yamazaki (2019). In itself it could only be used up until substitutions or one had to manually look at the player roles and shuffle the matrix according. I viewed the last as an undesirable aspect, thus I proposed a role matching based on the Euclidean distances as well as a local search extension to clus-

ter entire games. Down sampling the data is necessary since calculating a role matching every frame is computationally expensive and also unnecessarily accurate: players do not change position every 1/25th second. It was shown that both these adjusted methods can detect certain position changes accurately, though the Hungarian algorithm extension seemed more accurate. The Hungarian algorithm extension also seems the safest way to classify entire or multiple games. When comparing multiple games with the local search algorithm, there needs to be a lot of swapping. First, this will become computationally expensive, as local search algorithms often can be (Cai et al., 2019). Furthermore, there is a possibility that the entire formation is swapped vertically or horizontally to minimize the cost difference between two formations. This is undesirable, since a 4-3-3 and 3-3-4 are not the same, but swapping the latter horizontally might cause the local search algorithm to view them as very similar. The most considerable limitation of this algorithm is the fact that it is a clustering algorithm, which means the dendrogram need to be analyzed every time a game is clustered.

7.2.3 Results framework by Müller-Budack et al. (2019)

I also tested the method proposed by Müller-Budack et al. (2019). As shown in Figure 13 the accuracy was very low. There are several possible factors contributing to this low accuracy. First, recall that the authors normalized the positions to deal with narrow and wide formations (equation 22). Since this equation is based on single players, a player that is an outlier (very high up the pitch compared to the rest of the players, e.g.) could transform the players into distorted positions. This can be visualized with some examples. Figure 26 shows the normalizing process when there are no outliers. It works well by stretching the entire formation. But what if there are outliers? Figure 27 shows a situation in which the left-back and right-winger are fairly out of position. Note that this example is exaggerated, but especially in short observations these kinds of situations are not unlikely. Because of these two outliers, the rest of the formations is not scaled. If it was to be compared to a Ground-truth formation as shown in Figure 26 (recall that the distance measure is the Euclidean distance) it would make sense that the similarity would be very low. In Figure 27, $min_x = 0.1, min_y = 0.1, max_x = 0.95, max_y = 0.95$. What then happens to a properly positioned player such as the striker at $\tilde{r_9} = [x = 0.5, y = 0.75]$? In an ideal situation, his position would be transformed to $[x = 0.5, y = 1]$ (see Figure 26). In reality, this is what happens: $\frac{0.5-0.1}{0.95-0.1} \approx 0.47$ for x and $\frac{0.75-0.1}{0.95-0.1} \approx 0.76$ for y. As min_x approaches 0 and max_x approaches 1, the transformation becomes less and less. The same holds for y, evidently. In the end, because of outliers some formations might have almost not been transformed, whereas other formations without outliers are properly aligned, causing disparity.

Figure 26: Transformation for ground-truth formation.

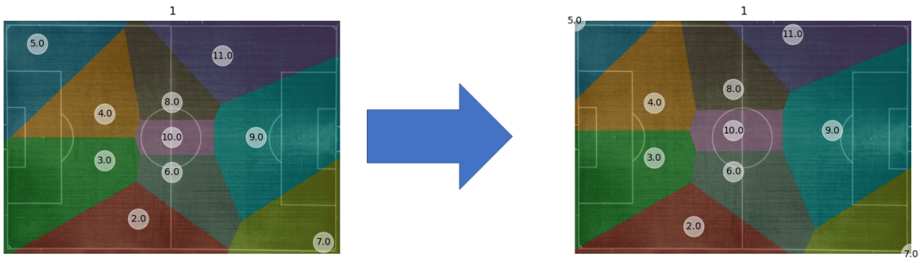


Figure 27: Transformation for an odd formation.

The second possible problem was with the distance function used by the Hungarian algorithm. Recall that the distance matrix $m_{i,j}$ was calculated according to equation 23 after which the Hungarian algorithm was applied for the best possible match. The max part made the function more explainable: if two players are close, their match approached 1, otherwise it would quickly approach 0. But it could be debated whether it works well in practice. Figure 28 shows a normalized observation, as well as the ground-truth labelled as a 4-2-3-1. The two are almost the same, yet the algorithm predicted a 5-4-1. If the scaling was done correctly, then the only fault can lie within the Hungarian algorithm and its distance measurement. Because equation 23 goes to 0 really quickly, it could be that the Hungarian algorithm is choosing a matching in which some players have a good matching and some players have a matching of 0. But since similarity is capped at 0 and approached quickly, there is no difference between a poor matching and a very poor matching. For that reason, I would suggest to not use the maximum function.

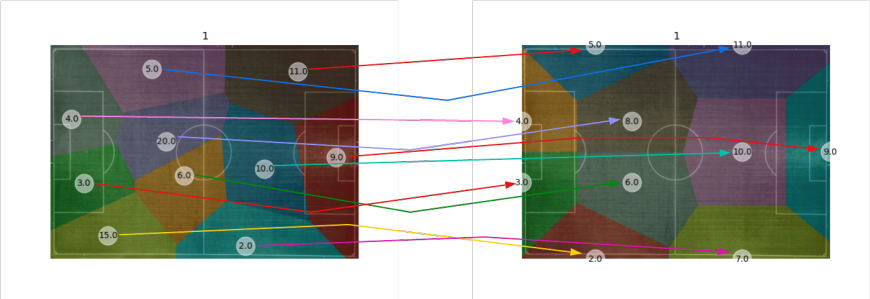


Figure 28: Matching between the ground-truth and an observation.

Figure 28 shows the result of the matching algorithm. All the players were correctly matched, except for the left-back and left-winger. The left-back was matched with the left-winger and vice versa. When equation 23 is changed to

$$M_{i,j}\left(F_1, F_2\right) = \left\|\tilde{r}_i - \tilde{r}_j\right\|_2^2 \quad \forall \tilde{r}_i \in F_1; \tilde{r}_j, \in F_2. \tag{28}$$

a perfect match is obtained (Figure 29) and 4-2-3-1 was predicted. Due to time constraints, this could not be tested on the entire dataset, but it could be a viable improvement.

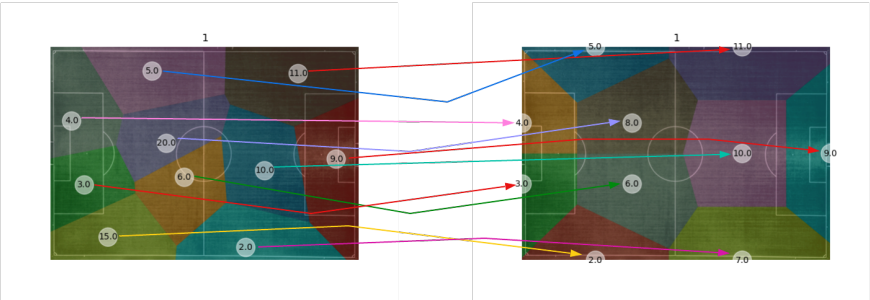


Figure 29: Perfect matching between the ground-truth and an observation

The third possible problem has to do with the nature of formations. Though coaches might like to think their team will play in a perfect formation, in reality the game is always played by two sides and there is always a response to an opponent and the ball. Therefore, formations are often not ground-truth perfect. This is different compared to the algorithm by Shaw and Glickman (2019), whose clusters come from actual games. Furthermore, the Euclidean distance might not be powerful enough to capture the distance between player roles.

7.3 Limitations & Future research

There are several limitations with regard to this research. The experimental set-up for the Wyscout dataset has serious validity issues, as it is not accurate at all. Therefore, it is unsure how well the 45% accuracy holds in real world situations. For this research there was a trade-off between accuracy and time. The Wyscout testset was chosen, as otherwise 70 matches would have to be annotated. Although the case study of the game between FC Groningen and Ajax seemed very promising, it was still only one match. Perhaps the teams played very distinct formations that game. For a proper conclusion on the usability, more games will have to tested. Another limitation is that this research was only tested on Dutch games, who predominantly play the Dutch school marked by the 4-3-3 or 4-2-3-1 formation. Future research should include games from different leagues for the trainingset.

In the previous section of the discussion, I briefly talked about the time it took to classify new observations. Right now, the most accurate algorithm classifies at 60 seconds per fragment, which might be too slow if the game is split up into phases and attack or defense. For future research, one might want to look into training neural networks based on the clusters returned by Shaw and Glickman (2019). Neural networks tend to be much faster at predicting new data, also because libraries like TensorFlow are thoroughly optimized in C (Sanchez et al., 2020). A specific type of Neural Network that could be interesting to look at are Graph Neural Networks. Graph Neural Networks work well with unstructured data: they do not have strict structural requirements, so edges, vertices and variables of the former can change (Ward et al., 2020). As we have seen in this research, formations are also unstructured and thus Graph Neural Networks might be suitable for this problem. Since Graph Neural Networks allow for variables concerning the edges and vertices, a lot of extra information could be added to more accurately describe formations. These could include the passes between players, the speed of players, the relative positions between each players as well as the absolute position. Campos (2021) was able to identify different phases of play using this technique. Stöckl et al. (2022) used Graph Neural Networks to determine the threat of an attack in football.

Another limitation is that the testing with the domain experts could be conducted more extensively. For example, only the clips classified by the algorithm were checked, but not clips that were potentially missed by the algorithm. To further gain insight into how the domain experts feel about the results, a questionnaire could be set-up.

Despite the limitations, this research can be seen as a solid step towards proving the effectiveness as well as the difficulties of automatically recognizing football formations. It is the first research to compare the accuracy of several algorithms and it has suggestions on how to implement these algorithms in the domain of football.

8 Appendix

Symbol	Description	Algorithm
$P(t)$	The set of player vectors $\{\vec{p_1}(t), ..., \vec{p_{10}}(t)\}$ given a time t.	Shaw & Glickman (2019)
$\vec{p_i}(t)$	The absolute position of player i at time t given in x and y.	Shaw & Glickman (2019), Narizuka & Yamazaki (2019)
$\vec{a_i}$	The average position of player i.	Shaw & Glickman (2019)
F	The Euclidean distance between two players.	Shaw & Glickman (2019), Müller-Budack et al. (2019)
W	The Wasserstein distance between two distributions.	Shaw & Glickman (2019)
C	A cluster of observations.	Shaw & Glickman (2019)
k	A scaling factor for the players' positions.	Shaw & Glickman (2019)
$\vec{cm}(t)$	The center of mass of a formation calculated by averaging all \vec{p}(t).	Shaw & Glickman (2019), Narizuka & Yamazaki (2019)
$\vec{r_i}(t)$	The relative positions by subtracting $\vec{cm}(t)$ from $\vec{p_i}(t)$	Shaw & Glickman (2019), Narizuka & Yamazaki (2019)
α	Scaler matrix for scaling the co-variances.	Shaw & Glickman (2019)
$\vec{\sigma}(t)$	Standard deviation of a team's formation at time t.	Narizuka & Yamazaki (2019)
A_{ij}	Adjacency matrix between two observations based on Voronoi regions.	Narizuka & Yamazaki (2019)
$D_{tt'}$	Dissimilarity matrix between two observations based on two A_{ij}.	Narizuka & Yamazaki (2019)
M_{ij}	Distance matrix between two observations based on the Euclidean distance between player pairs, and scaled by δ.	Narizuka & Yamazaki (2019), Müller-Budack et al. (2019)
δ	A tolerance radius that ensures that the distance between two players is made larger.	Shaw & Glickman (2019), Müller-Budack et al. (2019)
$FSIM$	The similarity between two formations.	Shaw & Glickman (2019), Müller-Budack et al. (2019)
SA	Smoothed accuracy where 0 is replaced by the $FSIM$ value when a classification mistake is made.	Shaw & Glickman (2019), Müller-Budack et al. (2019)
d	Distance penalty for a cluster size.	Shaw & Glickman (2019)

Table 7: Notation and description of most important symbols.

References

Aquino, R., Vieira, L. H. P., Carling, C., Martins, G. H., Alves, I. S., & Puggina, E. F. (2017). Effects of competitive standard, team formation and playing position on match running performance of brazilian professional soccer players. *International Journal of Performance Analysis in Sport*, *17*(5), 695–705.

Aurenhammer, F., & Klein, R. (2000). Voronoi diagrams. *Handbook of computational geometry*, *5*(10), 201–290.

Ayanegui-Santiago, H. (2009). Recognizing team formations in multiagent systems: Applications in robotic soccer. *International Conference on Computational Collective Intelligence*, 163–173.

Barra, S., Carta, S. M., Giuliani, A., Pisu, A., Podda, A. S., et al. (2021). Footapp: An ai-powered system for football match annotation. *arXiv preprint arXiv:2103.02938*.

Barris, S., & Button, C. (2008). A review of vision-based motion analysis in sport. *Sports Medicine*, *38*(12), 1025–1043.

Bialkowski, A., Lucey, P., Carr, P., Matthews, I., Sridharan, S., & Fookes, C. (2016). Discovering team structures in soccer from spatiotemporal data. *IEEE Transactions on Knowledge and Data Engineering*, *28*(10), 2596–2605.

Bialkowski, A., Lucey, P., Carr, P., Yue, Y., & Matthews, I. (2014). Win at home and draw away: Automatic formation analysis highlighting the differences in home and away team behaviors. *Proceedings of 8th annual MIT sloan sports analytics conference*, 1–7.

Bromiley, P. (2003). Products and convolutions of gaussian probability density functions. *Tina-Vision Memo*, *3*(4), 1.

Cai, X., Gao, L., Li, X., & Qiu, H. (2019). Surrogate-guided differential evolution algorithm for high dimensional expensive problems. *Swarm and Evolutionary Computation*, *48*, 288–311.

Campos, J. C. (2021). Determining the phases of play using graph neural network embeddings.

Fernandez-Navarro, J., Fradua, L., Zubillaga, A., Ford, P. R., & McRobert, A. P. (2016). Attacking and defensive styles of play in soccer: Analysis of spanish and english elite teams. *Journal of sports sciences*, *34*(24), 2195–2204.

Fradua, L., Zubillaga, A., Caro, Ó., Iván Fernández-Garcıa, Á., Ruiz-Ruiz, C., & Tenga, A. (2013). Designing small-sided games for training tactical aspects in soccer: Extrapolating pitch sizes from full-size professional matches. *Journal of sports sciences*, *31*(6), 573–581.

Graham, K. (2014). World Transfer Records No.14 – Trevor Ford To Sunderland.

Inan, T. (2020). Using poisson model for goal prediction in european football.

Jonker, R., & Volgenant, T. (1986). Improving the hungarian assignment algorithm. *Operations Research Letters*, *5*(4), 171–175.

Jurafsky, D., & Manning, C. (2012). Natural language processing. *Instructor*, *212*(998), 3482.

Larkin, P., & Reeves, M. J. (2018). Junior-elite football: Time to re-position talent identification? *Soccer & Society*, *19*(8), 1183–1192.

League, P. (2017). Premier league. *Premier League*.

Low, B., Rein, R., Schwab, S., & Memmert, D. (2021). Defending in 4-4-2 or 5-3-2 formation? small differences in footballers' collective tactical behaviours. *Journal of sports sciences*, 1–13.

Machado, V., Leite, R., Moura, F., Cunha, S., Sadlo, F., & Comba, J. L. (2017). Visual soccer match analysis using spatiotemporal positions of players. *Computers & Graphics*, *68*, 84–95.

Memmert, D., & Rein, R. (2018). Match analysis, big data and tactics: Current trends in elite soccer. *German Journal of Sports Medicine/Deutsche Zeitschrift fur Sportmedizin*, *69*(3).

Memmert, D., Raabe, D., Schwab, S., & Rein, R. (2019). A tactical comparison of the 4-2-3-1 and 3-5-2 formation in soccer: A theory-oriented, experimental approach based on positional data in an 11 vs. 11 game set-up. *PloS one*, *14*(1), e0210191.

Müller-Budack, E., Theiner, J., Rein, R., & Ewerth, R. (2019). " does 4-4-2 exist?"–an analytics approach to understand and classify football team formations in single match situations. *Proceedings Proceedings of the 2nd International Workshop on Multimedia Content Analysis in Sports*, 25–33.

Müllner, D. (2011). Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*.

Narizuka, T., & Yamazaki, Y. (2019). Clustering algorithm for formations in football games. *Scientific reports*, *9*(1), 1–8.

Olkin, I., & Pukelsheim, F. (1982). The distance between two random vectors with given dispersion matrices. *Linear Algebra and its Applications*, *48*, 257–263.

Ozanian, M. (2021). The World's Most Valuable Soccer Teams: Barcelona Edges Real Madrid To Land At No. 1 For First Timer. https://www.forbes.com/sites/mikeozanian/2021/04/12/the-worlds-most-valuable-soccer-teams-barcelona-on-top-at-48-billion/?sh=4ae1341916ac.

Petersen, K. B., Pedersen, M. S. et al. (2008). The matrix cookbook. *Technical University of Denmark*, *7*(15), 510.

Pettersson, D., & Nyquist, R. (2017). Football match prediction using deep learning. *Psychol. Sport Exerc.*, *15*(5), 538–547.

Rathke, A. (2017). An examination of expected goals and shot efficiency in soccer. *Journal of Human Sport and Exercise*, *12*(2), 514–529.

Sanchez, S., Romero, H., & Morales, A. (2020). A review: Comparison of performance metrics of pretrained models for object detection using the tensorflow framework. *IOP Conference Series: Materials Science and Engineering*, *844*(1), 012024.

Sharma, S., Batra, N. et al. (2019). Comparative study of single linkage, complete linkage, and ward method of agglomerative clustering. *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, 568–573.

Shaw, L., & Glickman, M. (2019). Dynamic analysis of team strategy in professional football. *Barça sports analytics summit*, 1–13.

Sorano, D., Carrara, F., Cintia, P., Falchi, F., & Pappalardo, L. (2020). Automatic pass annotation from soccer videostreams based on object detection and lstm. *arXiv preprint arXiv:2007.06475*.

Stöckl, M., Seidl, T., Marley, D., & Power, P. (2022). Making offensive play predictable-using a graph convolutional network to understand defensive performance in soccer. *Proceedings of the 15th MIT Sloan Sports Analytics Conference*.

Strauss, T., & von Maltitz, M. J. (2017). Generalising ward's method for use with manhattan distances. *PloS one*, *12*(1), e0168288.

Stübinger, J., Mangold, B., & Knoll, J. (2019). Machine learning in football betting: Prediction of match results based on player characteristics. *Applied Sciences*, *10*(1), 46.

Theagarajan, R., & Bhanu, B. (2020). An automated system for generating tactical performance statistics for individual soccer players from videos. *IEEE Transactions on Circuits and Systems for Video Technology*, *31*(2), 632–646.

Wagenaar, M., Okafor, E., Frencken, W., & Wiering, M. A. (2017). Using deep convolutional neural networks to predict goal-scoring opportunities in soccer. *ICPRAM*, 448–455.

Wang, Q., Zhu, H., Hu, W., Shen, Z., & Yao, Y. (2015). Discerning tactical patterns for professional soccer teams: An enhanced topic model with applications. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2197–2206.

Ward, I. R., Joyner, J., Lickfold, C., Rowe, S., Guo, Y., & Bennamoun, M. (2020). A practical guide to graph neural networks. *arXiv preprint arXiv:2010.05234*.

Welch, G., Bishop, G. et al. (1995). An introduction to the kalman filter.

Wu, Y., Xie, X., Wang, J., Deng, D., Liang, H., Zhang, H., Cheng, S., & Chen, W. (2018). Forvizor: Visualizing spatio-temporal team formations in soccer. *IEEE transactions on visualization and computer graphics*, *25*(1), 65–75.

Yuxuan Hu, K. L., & Meng, A. (2018). Agglomertive Hierarchical Clustering using Ward Linkage. https://jbhender.github.io/Stats506/F18/GP/Group10.html.