Recognizing Compound Events in Spatio-Temporal Football Data

Keywords: football analytics, event recognition, spatio-temporal data analysis, supervised machine learning

Abstract: In the world of football, performance analytics about a player's skill level and the overall tactics of a match are supportive for the success of a team. These analytics are based on positional data on the one hand and events about the game on the other hand. The positional data of the ball and players is tracked automatically by cameras or via sensors. However, the events are still captured manually by human, which is time-consuming and error-prone. Therefore, this paper introduces an approach to detect events based on the positional data of football matches. We trained and aggregated the machine learning algorithms Support Vector Machine, K-Nearest Neighbors and Random Forest, based on features, which were calculated on base of the positional data. We evaluated the quality of our approach by comparing the recall and precision of the results. This allows an assessment of how event detection in football matches can be improved by automating this process based on spatio-temporal data. We discovered, that it is possible to detect football events from positional data. Nevertheless, the choice of a specific algorithm has a strong influence on the quality of the predicted results.

1 INTRODUCTION

In recent years the use of spatio-temporal data increased strongly in various areas. Especially in the highly competitive sport sector new insights gained by positional information of players – tracked by different systems and methods during a game – can have a major impact on the training and tactic of a team. For professional football clubs performance analysis is an integral part of the coaching process [Carling et al., 2005]. In the context of performance analysis in football, many analyses are based on manually tracked and chronological ordered lists of game events on the one hand or the positional information of the players on the other hand [Mackenzie and Cushion, 2013]. For that reason, the significance and accuracy of analysis strongly correlates with the quality of the provided data. Detecting events manually is time-intensive and error-prone task. Based on the data of matches of the German Bundesliga, we discovered that the events are not time-synchronized with the positional information and sometimes associated with the wrong player.

Therefore, in this paper we present the implementation and evaluation of algorithms to automatically detect events in the positional data of football matches. We focused on following major events: passes, receptions, shots on target and clearances in this paper, since these ones are basic events, which have a high probability to occur more often during a match. We computed different features from the raw positional data of the ball. Based on these features, we detected event candidates by using different machine learning approaches – the Support Vector Machine (SVM), K-Nearest Neighbors (KNN) and Random Forest (RForest) classification. To be able to train these supervised learning techniques, we also created manually a gold standard based on the positional data and video data of the football matches. Additionally, we evaluated the three machine learning approaches by the recall and precision of the results.

The paper is organized in the following structure. In Section 2 we take a look at related work. Afterwards, we explain the properties of the provided data and introduce the created gold standard. In following section, we describe how the features are computed based on the positional data and Section 5 shows how we used these features to train different machine learning approaches in order to detect events. We also provide an evaluation about the quality of our results (see Section 6). Before we conclude the paper in Section 8, we give an overview about future work.

2 RELATED WORK

In the past, various projects focused on the extraction of spatio-temporal data out of video recordings [Mackenzie and Cushion, 2013, Barnard et al., 2003, Beetz et al., 2009]. Another approach to gather positional information during football games and exercises sessions are sensor-based systems like the RedFir system [von der Grün et al., 2011]. Based on this data Gal et al. [Gal et al., 2013] developed a system to detect shots on target. Also Madsen et al. [Madsen et al., 2013] focused on this event in connection with DEBS 2013 Grand Challenge [Christopher Mutschler et al., 2013]. Jiang and Yin [Jiang and Yin, 2015] presented an algorithms that uses deep convolutional neural networks to recognize events in the data provided by wearable sensors. A classification of human activities based on support vector machines was presented by Anguita et al. [Anguita et al., 2012]. Peterek et al. [Peterek et al., 2014] also focused on the detection of human activities, but they used an approach based on the random forest algorithm.

3 DATA FOUNDATION

As mentioned before, there are various providers of spatio-temporal data for professional football games. The quality, granularity, and accuracy of the data vary between different competitors and also strongly depend on the used tracking technology. The provided data sets typically consist of the positional information of the players and the ball, the manually tracked list of game events as well as some meta data about the teams and players. In this paper, we focus on data of games of the German Bundesliga. The range of the two-dimensional coordinates goes from goes from -52.5 to 52.5 for x and the data range of y goes from -34 to 34 (for pitches of the size of $105\text{m} * 68\text{m}$). Since the pitch size is not exactly defined, these numbers can differ for other stadiums. The center of the pitch has the coordinates $(0,0)$. The position values can exceed these limits. This indicates that the ball went out of bounds. Figure 1 shows a football pitch and the coordinates of its bounds.

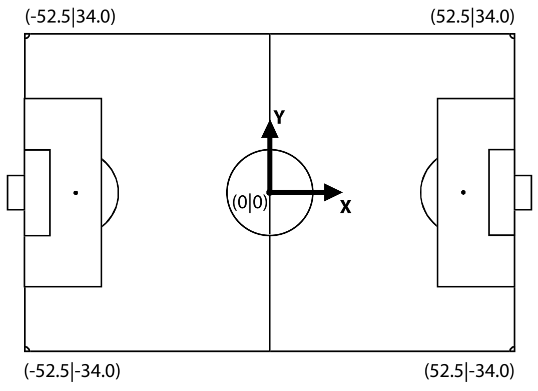


Figure 1: Football pitch with dimensions of bounds

The list of game events includes the timestamp, event type and involved players. All events are classified in the categories pass, shot on target, neutral contact, clearance, duel, foul, offside, caution, and substitution. Several events, such as fouls, cautions or substitutions, can not be detected just by the positional data of the ball and players. They also depend on other information, e.g. the signals of the referee. Additionally, the events are not synchronized with the positional information. The delay can be up to several seconds. To evaluate and train the supervised machine learning algorithms, we created manually a gold standard based on the video recordings of the games and by tacking into consideration the acceleration values of the ball. The gold standard includes the following three game sections:

- **Set A** Berlin vs. Mainz
  Season 2014/15, Time: 00:00 - 03:08
- **Set B** Berlin vs. Mainz
  Season 2014/15, Time: 25:00 - 31:42
- **Set C** Berlin vs. Braunschweig
  Season 2013/14, Time: 70:00 - 73:20

From the selected sections we excluded the times, when the ball was out of bounds or the game was paused. Afterwards we compared the gold standard with the provided event list. We were able to find 121 out of 194 (62.4%) matching events, within a time period of 2 seconds and with the same event type as our event. These events had an average time delay of 0.77 seconds. As a next step we examined the assigned player for these events. For the matched events, 18 out of 121 (14.9%) players were assigned wrong.

Table 1: Tagged events for gold standard

	Set A	Set B	Set C	Total
Pass	49	36	50	135
Reception	17	17	12	46
Clearance	0	5	1	6
Shot on Target	2	3	2	7
Total Events	68	61	65	194
Played Time	3:08 min	6:42 min	3:20 min	13:10 min
Excluded Time	0:58 min	1:49 min	1:36 min	4:23 min
Total Time	2:10 min	4:53 min	1:44 min	8:47 min

4 FEATURE COMPUTATION

Events in football matches are characterized by multiple features of the tracked objects. These objects move on the football pitch and influence each other. Events occur when one or multiple features show a specific value or change at the same time. In this section we present the definition of the features we implemented. All features are computed based on the positional data described in the previous section. The positional data is received per tracked object in an 2-by-n matrix where n is the number of collected data points in a time period. Each column vector represents the position of the object o at time t.

$$Pos_{o,n} = \begin{pmatrix} x_{o,t_1} & x_{o,t_2} & \cdots & x_{o,t_n} \\ y_{o,t_1} & y_{o,t_2} & \cdots & y_{o,t_n} \end{pmatrix} \tag{1}$$

For computing the features we used the Python framework Theano [Bergstra et al., 2010]. It provides several functionalities such as transparent use of the GPU. Theano also offers symbolic differentiation. This allowed us to define the features in a functional way and defer the execution. Due to the shape of the positional data, we were able to use convolution and other matrix operations to compute the features which depend on multiple data points efficiently.

We used the three types of convolution kernels to combine adjacent values. The first one computes the difference of two consecutive values in a row vector (ker_A). The second kernel computes the sum of two consecutive values in a row vector (ker_B) and the third kernel computes the sum of two consecutive values in a column vector (ker_C).

$$ker_A = \begin{pmatrix} 1 & -1 \end{pmatrix} \quad ker_B = \begin{pmatrix} 1 & 1 \end{pmatrix} \quad ker_C = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \tag{2}$$

We can derive the following definitions from the received positional data. The position of object o at time t is defined as $p(o,t)$. Whereas the horizontal position of object o at time t is $p_1(o,t) = p_x(o,t)$ and the vertical position of object o at time t is $p_2(o,t) = p_y(o,t)$. The difference d between two consecutive data points equals the movement of an object in 10^{-1} seconds. This indicates the direction of the object o at time t_1:

$$d(o,t_1) = p(o,t_2) - p(o,t_1) \text{ with } t_2 = t_1 + 1 \tag{3}$$

We used ker_A to compute the direction of an object. The following features will mainly focus on the position of the ball. But they can easily be applied to other objects.

4.1 Velocity

It is possible to reuse the direction of the object in order to determine the velocity of an object. Due to the provided data format we multiply the length of the direction vector by 10 to retrieve the unit $m * s^{-1}$. For velocity computation we used ker_B. The velocity v of object o at time t is defined as followed:

$$v(o,t) = |d(o,t)| * 10 \tag{4}$$

4.2 Acceleration

With the velocity computed, we can now take the difference of two consecutive velocity values to get the acceleration values. The result unit will automatically be $m * s^{-2}$. For computing the acceleration ker_A is used. The acceleration a of object o at time t_2 is defined as followed:

$$a(o,t_1) = v(o,t_2) - v(o,t_1) \text{ with } t_2 = t_1 + 1 \tag{5}$$

4.3 Acceleration Peaks

Due to the sampling rate of 10 Hz it can occur that the acceleration of an object is captured in multiple spatial data point. Therefore, we aggregated two consecutive values in order to find the real acceleration. The aggregation for maximum and minimum values has to be done separately. We ignored negative values for computing maximum peaks and positive peaks for computing minimum peaks by setting them to 0. In this way a not existing acceleration peak is represent with the value 0. We used ker_C to determine acceleration peaks. The maximum and minimum peak value - a_{max} and a_{min} - of object o at time t_2 are defined in the following way:

$$a_{max}(o,t_2) = \sum_{x \in \{t_1,t_2\}} max(0, a(o,x)) \text{ with } t_2 = t_1 + 1 \tag{6}$$

$$a_{min}(o,t_2) = \sum_{x \in \{t_1,t_2\}} min(0, a(o,x)) \text{ with } t_2 = t_1 + 1 \tag{7}$$

Furthermore, we prevented that acceleration peaks are detected at two consecutive timestamps. An acceleration peak is considered as real if there are no higher acceleration peaks at adjacent timestamps. Therefore, real acceleration peaks $a_{P_{real}}$ are defined as followed:

$$a_{max_{real}}(o,t_2) = \begin{cases} a_{max}(o,t_2) & \text{if } a_{max}(o,t_2) > a_{max}(o,t_1) \\ & \wedge a_{max}(o,t_2) > a_{max}(o,t_3) \\ & \text{with } t_1 + 1 = t_2 = t_3 - 1 \\ 0 & \text{else} \end{cases} \tag{8}$$

$$a_{min_{real}}(o,t_2) = \begin{cases} a_{min}(o,t_2) & \text{if } a_{min}(o,t_2) > a_{min}(o,t_1) \\ & \wedge a_{min}(o,t_2) > a_{min}(o,t_3) \\ & \text{with } t_1 + 1 = t_2 = t_3 - 1 \\ 0 & \text{else} \end{cases} \tag{9}$$

4.4 Direction Change

While objects move on the football pitch they will eventually change their direction. A linear movement results in no significant change of the direction feature. Whereas rapid movement tend to have a notable change of direction. We computed the change of direction as visualized in Figure 2.

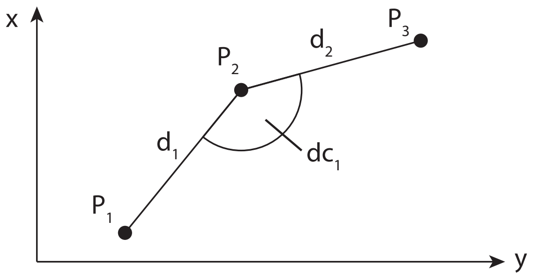


Figure 2: Direction change of object

Given the three position data points $P_1 = p(o,t_1)$, $P_2 = p(o,t_2)$ and $P_3 = p(o,t_3)$, the first direction vectors are defined as $d_1 = d(o,t_1)$ and $d_2 = d(o,t_2)$. The angle created by d_1 and d_2 is the change of direction dc_1. Possible values for direction changes are in the range from 0 to 180.

To determine the direction change value, the arccos function is applied to the quotient of the scalar product of d_1 and d_2 and the product of length of d_1 and d_2. The direction change dc of object o at time t_2 is defined in the following way:

$$dc(o,t_2) = \arccos\left(\frac{d(o,t_1) * d(o,t_2)}{|d(o,t_1)| * |d(o,t_2)|}\right) \tag{10}$$

The direction change is computed by using ker_A and ker_C as well as the Hadamard product.

4.5 Distance to Target

The players try to score in one of the goals on the pitch. These goals are considered as possible targets. While playing the object will move towards one of the targets. The corresponding target is chosen with regards to the horizontal movement of the object. This is independent of the position of the object. If the object moves to the left side, the left goal is assigned as target and vice versa. The reference point g of a target is located middle of the goal line and is defined as followed:

$$g(o,t) = \begin{pmatrix} sign(d_x(o,t)) * 52.5 \\ 0 \end{pmatrix} \tag{11}$$

Figure 3 displays different situation and the distances to the target. The four position data points $P_1 = p(o_1,t_1)$, $P_2 = p(o_2,t_2)$, $P_3 = p(o_3,t_3)$ and $P_4 = p(o_4,t_4)$ and the two targets $T_1 = \binom{-52.5}{0}$ and $T_2 = \binom{52.5}{0}$. The arrow at each position data point represents the approximate direction of the respective object at the same time. The objects at P_1 and P_2 have a positive horizontal movement ($d_x(o,t) > 0$). Therefore the corresponding target for these two point is T_2. The object P_3 has a negative horizontal movement ($d_x(o,t) < 0$). Its target is T_1. The object at P_4 has no horizontal movement ($d_x(o,t) == 0$). This is a special case where no target can be determined. The distance to target feature will return *infinity*.

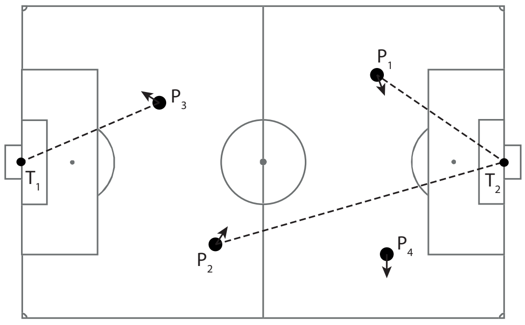


Figure 3: Distance for object to target

In cases where a target can be determined, the distance to target value is equal the length of the vector subtraction of the current point of the object and the position of its target. We used ker_A and ker_C for the distance to target computation. The distance to target value dt for object o at time t is defined in the following way:

$$dt(o,t) = |p(0,t) - g(o,t)| \tag{12}$$

4.6 Cross on Target Line

As discussed in the previous subsection, the objects on the pitch are alternately targeting one of two targets. Beside the distance of the object to the target, another feature is the proximity of the movement to the target. We defined this as the distance from the target to the point where the object will cross the goal line assumed that the object will maintain its directional movement.

Figure 4 shows the position $P_1 = p(o,t_1)$ of object o and its directional movement $d_1 = d(o,t_1)$. If the object continues its movement without changing the direction, it will cross the goal line at C_1. The distance between the target T_2 and C_1 is the measurement for this feature.

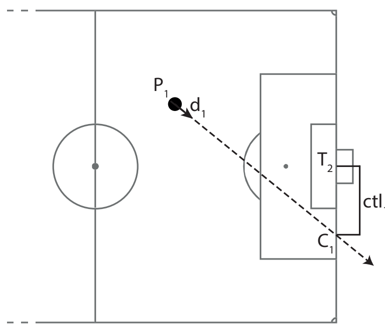


Figure 4: Cross on target line of object

To compute the cross target line feature, we had to solve a linear equation (cf. Equation 13). A multiple of the direction vector is added to the position of the object until it reaches the goal line at any point. The vertical difference to the target point is the desired distance. The cross target line feature *ctl* for object *o* at time *t* is defined as followed:

$$\begin{pmatrix} g_x(o,t) \\ ctl \end{pmatrix} = p(o,t) + s * d(o,t) \tag{13}$$

Repositioned for *ctl*:

$$ctl(o,t) = p_2(o,t) + d_2(o,t)\frac{g_1(o,t) - p_1(o,t)}{d_1(o,t)} \tag{14}$$

5 EVENT DETECTION

The most central object of a football match is the ball. All players try to interact with it. The ball is also the object that shows the most and highly rapid movements on the pitch. Therefore, we computed all features described in Section 4 for the ball. With all features we created a vector for every time *t* containing all corresponding feature values.

Velocity and acceleration describe the current momentum. Acceleration peaks were introduced due to the provided data schema, since they are a strong indicator for interactions with the ball. The direction change feature covers ball interactions with high intensity (e.g. passes) as well as ball interactions with little intensity (e.g. ball touches during dribblings). The distance to target is important to distinguish between shots on target and clearances and is an indicator for the likelihood of a shot on target in comparison to a pass. The cross on target line feature represents a measurement whether a shot will hit the target or not. Each vector describes an instant of the football match and can represent a certain event. Depending on the type of the event, features become more or less important and have characteristic values. A naive approach to classify events would be:

- **Pass:** The ball has an acceleration peak with a minimum value and/or shows a significant direction change.
- **Reception:** The ball shows negative acceleration peak or direction change. Afterwards the ball stays close to a specific player.
- **Shot on target:** The ball is accelerated with a medium to high value and a direction change. Shots on the target occurs most likely within a short distance to target and they are aiming for target.
- **Clearance:** The ball has a high positive acceleration peak and direction change. Clearances mostly happen to prevent risky situations near to the own goal line. Therefore, they have a high distance to target.

To determine a exact differentiation between the events, we selected three supervised classification machine learning algorithms based on related work and common approaches: Support Vector Machine, K-Nearest Neighbors and Random Forest. We used the implementations of the Python package Sklearn [Pedregosa et al., 2011] to define the following output classes: no event, pass, reception, shot on target and clearance.

5.1 Machine Learning Algorithms

In the following section we explain the three machine learning algorithms we used and their configuration.

The *Support Vector Machine* [Cortes and Vapnik, 1995] approach tries to divide the data points in a space into categories based on the provided training data. Thereby the dividing gap has to be as wide as possible. SVMs are effective in a high dimensional space, which is provided by our vector of features values. We used SVM with a linear kernel. This results in a linear divider gap. Furthermore, we used the provided option to determine the weight of each output class automatically. This prevents an over-weighting of classes with a high frequency (e.g. no events). We did not limit the number of iterations. The *K-Nearest Neighbors* algorithm [Altman, 1992] determines the output class by a majority vote of the closest neighbors of a data point. Our results showed that a configuration with $k = 3$ will return the best results. A *Random Forest* approach [Breiman, 2001] consists of multiple decision trees. These decision trees are a very similar implementation to the naive event classification we present earlier in this section. The RForest increases the predictive accuracy and controls over-fitting. In order to add more over-fitting prevention, we limited the number of decision trees to 10. Each decision tree has a maximal depth of 4. As describe for the SVM algorithm, we use the provided option to automatically determine the weight of the output class.

5.2 Event Candidate Aggregation

The three classifier algorithms return their prediction for the test data set. In order to increase the accuracy we allow an aggregation of these results. A customizable weight is assigned to each classifier algorithm for each event type. In addition every event type has a minimum score. If the classifier predicted an event at a timestamp, the weight is added to the score of this event at this timestamp. As a final result only events with a score equal or greater the minimum score are considered as detected events. A possible configuration for this would be the weight 0.4 for SVM and RForest and weight 0.2 for KNN with a minimum score of 0.8 for all event types. With this configuration only events detected by the SVM and the RForest algorithm are accepted.

This allows us to add more prediction algorithms to our implementation and integrate their results, depending on how precise or complete their results are for certain event types.

6 EVALUATION

In the following section we show an evaluation of the three machine learning approaches we described in Section 5. We assessed their quality by precision and recall, which are defined as followed:

$$precision = \frac{true\_positives}{true\_positives + false\_positives} \tag{15}$$

$$recall = \frac{true\_positives}{true\_positives + false\_negatives} \tag{16}$$

For the detailed evaluation of the results we focused on the event types to passes and receptions. We also introduced the overall type of ball touches. Our assumption was, that all four event types have similar aspects for their feature characteristics. Therefore, this event type represents the ability of an algorithm to distinguish between an event occurrence and an event absence. The input set is the union of all four initial event types.

We evaluated different data sets, which are described in Section 3. On the one hand, we trained and tested on the same data set. We learned from 90% of the data and tested on 10% of the data. We split the data set randomly 100 times and calculated the arithmetic mean for the precision and recall of all iterations. The repetition of the random split proceeding should ensure that our results are statistically comprehensible and deviations are mitigated. On the other hand, we trained the events from one set and tested it on another. We wanted to examine if the results change when time has passed during a game or when it is another game with different players. For this variant just one iteration was needed, since no random split was processed.

Figure 5 shows the precision and recall for passes, receptions and ball touches tested within the same data sets. The dashed lines inside the diagrams also indicate the value of the f-measure ($f1$ score), which is the harmonic mean of precision and recall:

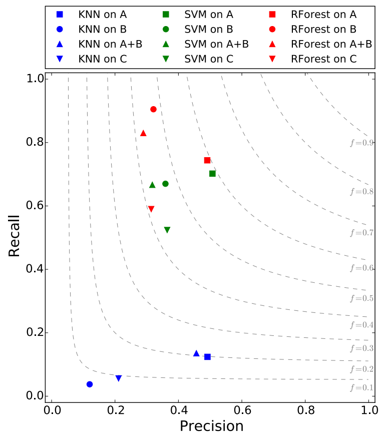
$$f = 2 * \frac{precision * recall}{precision + recall} \tag{17}$$

In Table 2 we compare the algorithms quality by the arithmetic means of the precision and recall for the prediction within the data sets. It turns out that the KNN algorithm offers results with a quite low quality. The recall never exceeds 10%. The precision for passes and ball touches is between 32% and 43.8%. Both precision and recall are 0% for receptions. The SVM and RForest results are close to each other and have a noticeable higher recall than KNN. The recalls are between 59.9% and 76.7%, except for RForest receptions where it is 43.6%. The precisions are between 35.1% and 38.7%. An exception is again receptions, where the precision is between 5.9% and 8.9%.

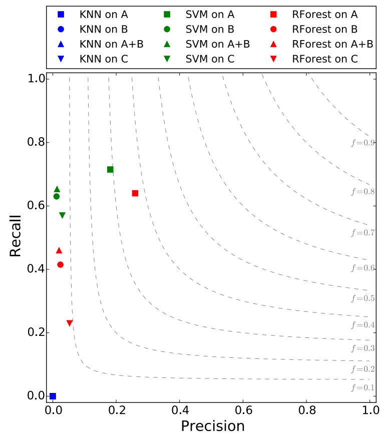
Table 2: Algorithms quality for prediction within data sets

Algorithm	Event Type	Precision	Recall	F-Measure
KNN	pass	0.320	0.088	0.138
	reception	0.000	0.000	0.000
	ball touch	0.438	0.095	0.156
SVM	pass	0.387	0.641	0.483
	reception	0.059	0.642	0.108
	ball touch	0.305	0.716	0.428
RForest	pass	0.354	0.767	0.484
	reception	0.089	0.436	0.148
	ball touch	0.351	0.595	0.442

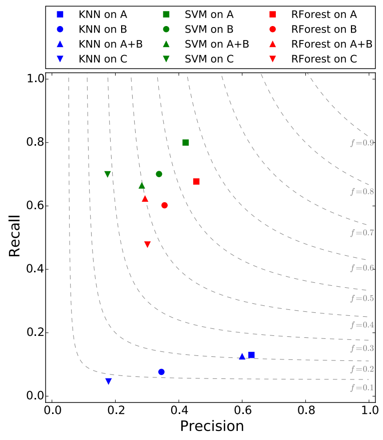
The precision and recall for the prediction across data sets can be seen in Table 3, which is the summarization of Figure 6. The magnitudes and fluctuations of the results are similar to the prediction within data sets results, which were explained before. In summary, the quality is just slightly lower than previously.



(a) Pass

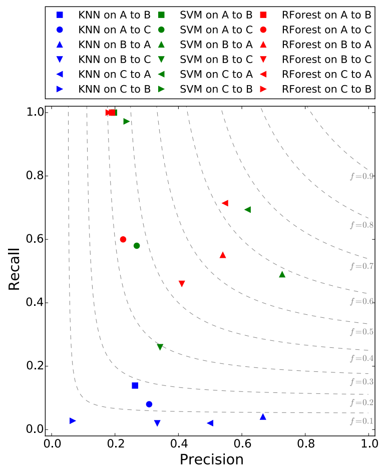


(b) Reception

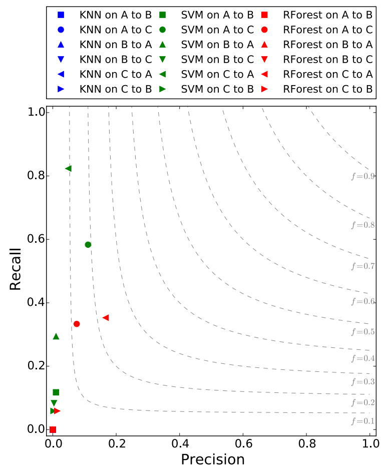


(c) Ball touch

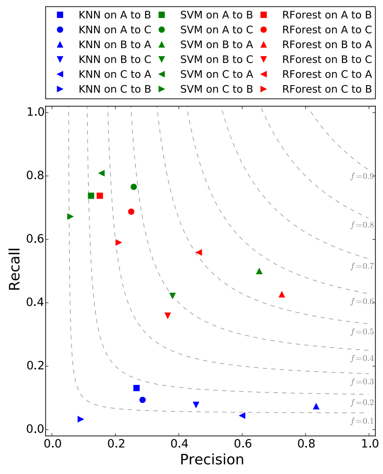
Figure 5: Event recognition within data sets



(a) Pass



(b) Reception



(c) Ball touch

Figure 6: Event recognition across data sets

The overall precision is around 1% lower and the recall 9% lower. This could be caused by the already mentioned effect, that different players show a different skill and players also get exhausted over time.

Table 3: Algorithms quality for prediction across data sets

Algorithm	Event Type	Precision	Recall	F-Measure
KNN	pass	0.356	0.055	0.095
	reception	0.000	0.000	0.000
	ball touch	0.422	0.076	0.128
SVM	pass	0.398	0.666	0.498
	reception	0.031	0.327	0.057
	ball touch	0.272	0.651	0.383
RForest	pass	0.311	0.727	0.435
	reception	0.057	0.124	0.078
	ball touch	0.373	0.544	0.443

Finally, we aggregated the results for the prediction across data sets, as explained in Section 5.2. Since the precisions for the different event types are nearly the same for the three algorithms (cf. Table 3), we used as a configuration for the aggregation a weight of 0.4 for every algorithm and a minimum score of 0.8. Therefore, two out of three algorithms have to find an event at a given timestamp, in order to confirm this event. Table 4 shows that we increased the f-measure for passes and ball touches compared to all three algorithms between 1.6% and 41.9% (17.1% average). We also increased the f-measure for receptions compared to the KNN (6.7%) and SVM (1%) algorithm, but decreased it about 1% for RForest.

To summarize, our results show that it is possible to detect football events from positional data with our approach based on machine learning. Also the choice of a specific algorithm can have an extensive impact on the quality of the predicted results. The

Table 4: Aggregation quality for prediction across data sets

Event Type	Precision	Recall	F-Measure
pass	0.426	0.647	0.514
reception	0.047	0.119	0.067
ball touch	0.372	0.713	0.489

SVM and RForest algorithms showed reasonable results, whereas the KNN algorithm failed to convince us for this use case. With the aggregation of the different algorithms the results could be further improved. Therefore, a valuable configuration of the weights and the minimum score is needed. Nevertheless, there is still potential for optimizations.

7 FUTURE WORK

In this section we face discovered issues and suggest proceedings to further improve and extend our work. The missing z coordinate (height information) is necessary to calculate features such as velocity, acceleration and direction change correctly, since the ball is moving in a three-dimensional space. When the ball is out of bounds or the game is stopped the positional data should be ignored, because the data is inappropriate to draw conclusions about that. We also discovered problems with the distinction of receptions that occur directly before a pass, which happens when a player receives the ball and immediately shoots it to another player.

To extend our approach it would be possible to train the features for every player separately, since every player has a different skill and shows its own behavior. It is also possible that a player changes its behavior between matches or even during a match, when a player gets exhausted. This would make it even more challenging to classify the different event types. To validate an event candidate it would be an extension to integrate what is happening before and after the event time and what the players around the event position are doing. We focused mainly on the moment of an event identified by the features of the ball, but probably more information can be gathered this way.

8 CONCLUSION

In this paper, we proposed to detect events from positional data of football matches, in order to replace the error-prone and time-consuming task of capturing these events manually. We presented a supervised machine learning approach that classifies compound football on base of different features, which are computed from positional data. To calculate those features efficiently for over one million tracking events per match, we used matrix operations, particularly convolution, and optional GPU features. We used the Support Vector Machine, K-Nearest Neighbors and Random Forest classification algorithms to recognize the event classes of passes, receptions, shots on target and clearances in our self-captured gold standard. Additionally, we enhanced the approach with a customizable aggregation algorithm, to be able to weight the outcome of the algorithms for different event types.

We evaluated the three algorithms by their quality of precision and recall. Compared to KNN algorithm, the SVM and RForest algorithms showed reasonable results with a precision of 31.1% up to 39.8% and a recall of 66.6% up to 72.7% for passes. Receptions seemed to be difficult to distinguish from passes for the classification algorithms, which is why we received lower results for them. We further improved the results with the aggregation of the different algorithms and increased the f-measures with an average of 12.1% for all event types compared to the three algorithms. To improve our approach for a practical use, we showed different ideas for future work. In conclusion, our results showed that it is possible to detect football events from positional data, but the choice of a specific algorithm can have an extensive impact on the quality of the predicted results.

REFERENCES

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185.

Anguita, D., Ghio, A., Oneto, L., Parra, X., and Reyes-Ortiz, J. L. (2012). Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *Ambient assisted living and home care*, pages 216–223. Springer.

Barnard, M., Odobez, J.-M., and Bengio, S. (2003). Multi-modal audio-visual event recognition for football analysis. In *Neural Networks for Signal Processing, 2003. NNSP'03. 2003 IEEE 13th Workshop on*, pages 469–478. IEEE.

Beetz, M., von Hoyningen-Huene, N., Kirchlechner, B., Gedikli, S., Siles, F., Durus, M., and Lames, M. (2009). Aspogamo: Automated sports game analysis models. *International Journal of Computer Science in Sport*, 8(1):1–21.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010). Theano: A CPU and GPU Math Compiler in Python. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, 9.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Carling, C., Williams, A. M., and Reilly, T. (2005). *Handbook of soccer match analysis: A systematic approach to improving performance*. Psychology Press.

Christopher Mutschler, Holger Ziekow, and Zbigniew Jerzak (2013). The DEBS 2013 Grand Challenge. In ACM, editor, *Proceedings of the 7th ACM International Conference on Distributed Event-Based Systems*, pages 289–294.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Gal, A., Keren, S., Sondak, M., Weidlich, M., Blom, H., and Bockermann, C. (2013). Grand challenge: The techniball system. In *Proceedings of the 7th ACM international conference on Distributed event-based systems*, pages 319–324. ACM.

Jiang, W. and Yin, Z. (2015). Human activity recognition using wearable sensors by deep convolutional neural networks. In *Proceedings of the 23rd Annual ACM Conference on Multimedia Conference*, pages 1307–1310. ACM.

Mackenzie, R. and Cushion, C. (2013). Performance analysis in football: A critical review and implications for future research. *Journal of Sports Sciences*, 31(6):639–676.

Madsen, K. G. S., Su, L., and Zhou, Y. (2013). Grand challenge: Mapreduce-style processing of fast sensor data. In *Proceedings of the 7th ACM international conference on Distributed event-based systems*, pages 313–318. ACM.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Peterek, T., Penhaker, M., Gajdoš, P., and Dohnálek, P. (2014). Comparison of classification algorithms for physical activity recognition. In *Innovations in Bio-inspired Computing and Applications*, pages 123–131. Springer.

von der Grün, T., Franke, N., Wolf, D., Witt, N., and Eidloth, A. (2011). A real-time tracking system for football match and training analysis. In *Microelectronic systems*, pages 199–212. Springer.