Online Data Stream Mining on Interactive Trajectories in Soccer Games

Christopher Mutschler, Gabriella Kőkai, Thorsten Edelhäußer

Abstract—In sports, especially soccer and other ball-sports, the interaction between trajectories of players take a major role in purposes of game analysis. Existing approaches follow the way of mining spatio-temporal, sequential patterns or spatial patterns, but do not incorporate interactions between multiple, interactive trajectories. This paper proposes an online data mining method that is able to recognize repeating patterns of trajectory movements regardless of how they are scaled, rotated or translated. The result is an efficient data stream mining application, which provides meaningful information within the current game. The framework is designed in a modular manner in order to enhance the capabilities easily. The evaluation of the algorithms proves the efficiency of the provided concepts.

I. INTRODUCTION

Online data stream mining is a new field of study, but has already drawn much interest in many kinds of research areas. Conventional data mining techniques like clustering, synopsis or indexing have been efficiently modified for volatile data in data streams. In contrast to static data, where the information is normally stored in relational databases in terms of tuples that can be queried, data streams need a more sophisticated approach due to the volatility of the data. However, these problems have already been encountered in many different ways by transferring the conventional data mining methods to work on stream data [1]. In contrast to static data, stream oriented data emerges theoretically indefinitely, but in spatio-temporal order. With high data rates, it is usually not possible to randomly access the data because it cannot be stored. The methods presented in this paper efficiently extract re-occurring patterns of interactive trajectories from soccer players and the ball. The positioning system is working radio-based with each soccer player having at least one transmitter, e.g. within his shin pad; therefore describing movements by a sequence of spatio-temporal points. The ball and the referees are equipped with radio transmitters as well. The key idea of our approach is to extract tactical movement information of soccer teams by finding similar trajectory patterns and incorporating their interactions.

As opposed to most of the conventional data mining problems, the trajectories of the objects and their interactions are not pre-defined. Conspicuous patterns cannot be determined without receiving at least one similar pattern.

II. REQUIREMENTS

The requirements on such a data stream mining system are widely varied:
- The tracking system is working erroneous. This can be for instance the accuracy of the Euclidean coordinates, which is perturbated with noise [9]. Non-valid TDoA (Time Difference of Arrival) and extremely differing values
are detected and corrected at earlier stages (by phase measurements, Kalman filtering etc.). The trajectories are smoothed by the application of a Gaussian distribution filter onto the incoming positions.

- The system should analyze ball-driven tactical patterns which have been practiced by the opponent team. Offensive or defensive analysis or guarding behaviour are not discussed in this paper.
- The data needs to be processed online in order to give coaches the ability to intervene the game process of their team. Applied algorithms have to run in short time intervals with no exhaustive computations. Good approximations should be prior to exact value calculations.

For instance, for a pattern, played three different ways like depicted in Figure 2, a similarity measure has to fulfill a variety of demands to classify the patterns as equal. In the given example, the patterns have to be moved in space, have to be rotated and also scaled to be matched.

III. RELATED WORK

Analysis of trajectories and learning their distributions has been of special interest since the end of the last century [6]. Many research teams have already investigated problems in this area, especially for clustering trajectories and therefore, finding a measure for their similarity.

Lee et al. [8] proposed a Partition-and-Group framework, a method that efficiently clusters not only entire trajectories but parts of them. Given a set of trajectories, having a common behaviour in a particular moment of time, they partition the trajectories into a set of line segments and then group their line segments together. The distance, or the dissimilarity of two partitions, is calculated as a weighted sum of the perpendicular distance, the parallel distance and the angular distance. The trajectory will then be sampled at discrete points such that the preciseness is maximal and conciseness is minimal. The distance of two partitions is then the sum of the distances between the discrete points. Unfortunately, the distance measure cannot be directly transferred, because the trajectories are analyzed in their global and not in their local environment. Neither the angle, nor the position of the trajectory is important to aggregate similar movement patterns. The method presented in this paper analyzes the trajectories in relation to their teammates. Hence, tactical patterns are independent of angles and positions.

Yanagisawa et. al [12] presented a similarity distance between trajectories based on shape comparison. Since the complexity of calculating the defined similarity is very high, they developed an indexing method based on Piecewise Aggregate Approximation (PAA), a technique that reduces costs for the comparison of two sets of time series data. The key idea is to approximate the lower limit of the distance between two trajectories by calculating average values. If the lower limit of distance is already too high to determine two trajectories similar, the algorithm stops. If the limit is within the acceptable range, the solution is upgraded and the calculation runs again with an approximation on a higher resolution. However, the presented technique only provides the ability to index a set of defined trajectories and to define a similarity between them. The longer the trajectories, the smaller are the similarities between them. In the problem described in this paper, it is not clear when a pattern (and therefore the trajectories of it) start or end - trajectories cannot be defined accurately. Additionally, the computational complexity is still very high and hinders an online processing of the data streams.

Cao et al. [4] presented a method to recognize reappearing movements by segmentation of the movement area. They counteract the problems that arise with spatio-temporal sequential patterns, from a long spatio-temporal sequence: given an object’s movement as $aabombofg$, where each character $a$, $b$ etc. corresponds to a spatial region. Obviously, the occurrence of the pattern $abc$ is 1, assuming that the object moves from $a$ over $b$ to $c$ exactly once. Due to the definition of support, the pattern $abc$ has support 4 for a window size of 5 because of the contribution of the 4 segments $abc$, $aba$, $aba$, $aba$. This problem is solved by mining a substring tree, proposed by the authors. However, with bigger cellsizes, the position error increases. On the contrary, by shrinking the size, the actual whereabouts will approximate their original position but the classification rate decreases. Although the method cannot be directly applied due to the lacking ability to model interactions, the transformation of positions to a sequences of places can be promising. We adapted this idea and assign the ball to the players to get a sequence of possessions.

Kang et. al [7] propose a method for quantitatively express the strategic performance of soccer players. They preprocess
the given data to get ball possession, kicking and dribbling information. Whenever a player kicks the ball (either to a teammate or somewhere else), the authors compute so-called catchable and competing regions. This is done by incorporating the initial point at the kick and some values like speed, orientation etc. By drawing this in a three dimensional space \((x, y, z)\) as usual, \(time\) as third coordinate), they get cones by which these regions are fully described. For each player and the ball, these catchable regions can be evaluated such that the authors can score each individual ball kick. By averaging these emerging integrals, each player can be scored by his ball kicks. However, the problem solved by the authors does not really match the given problem since they do not try to detect repeating patterns in the game. Additionally, even the authors state, that they did not deal with performance issues of processing in their approach. The calculation of the integrals and the join operations for catchable regions are requiring much computation time.

Perl [10] researches the recognition of movement patterns with the help of Kohonen Feature Maps (KFM), a special type of self-organizing maps in addition to Artificial Neural Networks that simply consists of Neurons (vertices) and Synapses (edges), interconnecting the Neurons. During the learning phase, data will be given to the neurons, which will change their state and that of a predefined neighboring set of neurons. With several improvements, it becomes possible to train such a network with less training examples than before (about 500 instead of 5000 for a feature map of \(20 \times 20\) neurons, depending on the task). The disadvantage of this method is that the system as a whole has to be learned previously to the recognition phase with a still great amount of training data that we cannot provide because we do only have limited information about playing patterns of particular opponents. Another disadvantage is, that the system is not able to recognize movements that have not been learned previously, which results in a bad classification rate.

Beetz et al. developed a system called ASPOGAMO [3] which is short for Automated Sports Game Analysis Models and formerly known as CAESAR (Camera enabled sport game media and viewers). The outcome of the observation system is a set of ball and player trajectories. ASPOGAMO provides a tactical behaviour or game strategies. Indeed, the authors do not present the information provided to the user. The current research seems to focus on the detection of players and the tracking mechanism. Therefore, the work does not fit to the given task but shows the relevance of this topic.

Any of the described research presents very good approaches to measure similarity of trajectories in order to analyze them. Unfortunately, the described methods do not fit to the given problem very well. Firstly, previous work does not consider interactions between moving objects, but only static behaviour. Secondly, the given algorithms are not readily portable on data streams, which means that they cannot cope with volatile data. Thirdly, none of the reviewed work regards time and space complexity of the developed algorithms. In order to run online, the applied methods have to be strictly limited in computation time and memory consumption.

IV. AKIMA APPROXIMATION

The positioning system that has been applied in this work provides more than 50,000 object positions per second, which is significantly more than needed. The data packets contain not only the \(xy\)-positions of the objects but also the velocity and acceleration in each axis. The storage of raw positioning data would overflow the available storage capacity of nowadays computer systems sooner or later when incorporating multiple games. Hence, it is necessary to find a way to efficiently store data, even if the storage would imply lossy compression. In order to analyze movement patterns of soccer players, it is usually not important to have an accuracy of more than a few inches. Spline approximation proved suitable for the given scenario. Depending on the speed of the players and their movements, sampling points can be chosen at proper locations without losing much information. Additionally, with polynomial interpolation methods, the sampling point rate is not fixed. For instance, a system could detect movements with higher variance and increase the sampling rate in order to balance the error, without increasing the complexity of the interpolation algorithm afterwards.

To apply splines for the purpose of trajectories, it is necessary to parameterize the movement in time to get to distinctive sets \((t_i, x_i)\) and \((t_i, y_i)\), so that the polynomials are able to approximate curves rather than only functions. The Akima spline method produces a set of polynomials from the power of three with the property that only the closely neighboring sampling points have an impact on the slope at each point. This causes the spline to be more flexible to changes in direction, which is also depicted by the fact that the akima spline is not necessarily two times steadily differentiable. A polynomial that approximates a curve between \(A_i\) and \(A_{i+1}\) is defined by

\[
y_i = a_i + b_i \cdot (x - x_i) + c_i \cdot (x - x_i)^2 + d_i \cdot (x - x_i)^3
\]

Akima determines the slope \(t_i\) at a point \(i\) by [2]:

\[
t_i = \frac{[m_4 - m_3] \cdot m_2 + [m_2 - m_1] \cdot m_3}{[m_4 - m_3] + [m_2 - m_1]}
\]

where \(m_1, m_2, m_3\) and \(m_4\) are the slopes of the line segments \(12, 23, 34\) and \(15\), respectively. Under this circumstances, the slope at point 3 only depends on the points 1, 2, 4 and 5. The coefficients are calculated by

\[
a_i = y_i
\]

\[
b_i = t_i
\]

\[
c_i = \frac{3(a_{i+1} - a_i)}{x_{i+1} - x_i} - \frac{2(b_i - b_{i+1})}{(x_{i+1} - x_i)}
\]

\[
d_i = \frac{2(a_{i+1} - a_i)}{x_{i+1} - x_i} - \frac{2(b_i + b_{i+1})}{x_{i+1} - x_i}
\]
The interpolation of positioning points is simply done by searching the proper coefficients and calculating the value by the function given above. To avoid logarithmic complexity needed for the binary searching of the coefficients, an index list has been implemented that sets timestamps every second. Hence, with \( r \) sampling points per second, the complexity is \( O(r) \), which is significantly better. An evaluation of spline interpolation algorithms can be found in [5].

V. TRAJECTORY SIMILARITY

Comparing a current sequence to the entire past stream data would not be applicable and would also not be possible with increasing size of the available stream data. Hence, it is necessary to perform a pre-selection process that extracts possible matching patterns and then coordinates the matching process. The matching process needs to normalize the played patterns among each other due to their translation, rotation and scaling.

Firstly, the ball possessioning algorithm is explained. Secondly, the pre-selection process is illustrated and thirdly, the transformations of the patterns are described.

A. Ball Possession

For humans, the assignment of the ball to a single player may be a trivial problem. Nevertheless, the proper assignment incorporates some calculations. Intuitively, at least the following two requirements must be met:

1) The distance between the ball and the possessing player has to be below a given distance. Due to error rates in the wireless transmission system, the player with the smallest distance to the ball is not necessarily the possessor. If some players are moving really fast and the ball is spinning a lot, the measured distance between the player’s leg and the ball can reach a value of more than three feet in our test data.

2) The acceleration of the ball must have a negative peak at the beginning (when the player stops it) and a positive peak at the end of the possession (when the ball gets passed). Anyhow, if a player gains or loses the possession during a tackle, the first or respectively the second peak does not necessarily occur.

Figure 3(a) shows the acceleration from the ball and Figure 3(b) shows a plot of the distance between a player and the ball, both for the same period of time. Obviously, the player does not possess the ball in times when the distance is considerably high. Only periods in which the distance is nearly zero, the player might be a candidate for the possessor.

In this work, different relations and additional requirements have been evaluated. With \( a \) being the acceleration of the ball and \( d \) being the distance between a player and the ball, a reliable measurement function can be expressed by

\[
\vartheta(a, d) = \begin{cases} 
\frac{d}{a}, & \text{if } d < 3 \text{ feet} \\
0, & \text{else}
\end{cases}
\]

Now, for each player, the function can be calculated and the result can be analysed. Players with a distance of more than 3 feet need not further be distinguished due to the 2nd condition of the equation. Consequently, the result is a signal for each player who might be a candidate for the ball possessor. Figure 3(c) shows the resulting signal for the given distance and acceleration plots. Obviously, the player owns the ball at the end of the period, where the peak is really high. In the middle, there are two slight peaks. For this cases, the system evaluates any candidate players and chooses the one with the highest peak. If a threshold for a distinctive assignment between several players cannot be met, the system holds all possible players as potential possessors and performs fuzzy classification later on.

B. Pre-seperation of data stream sequences

Tactical patterns in sports strictly depend on the interactions between the players, i.e. the sequence of passings. Imagine a pattern with players 1, 2 and 3. A possible sequence of interaction with the ball could be (players possessing the ball):

\[ s_1 = 1 \rightarrow 2 \rightarrow 1 \rightarrow 3 \]

Some time later, other players (but from the same team), for example players 3, 4 and 5 perform a similar sequence:

\[ s_2 = 4 \rightarrow 3 \rightarrow 4 \rightarrow 5 \]

Obviously, both sequences \( s_1 \) and \( s_2 \) could eventually realize the same interaction pattern, because only the participating players are changed. Otherwise, if there would be no possible way to arrange the players to form the same patterns such that \( s_1 \equiv s_2 \) (like \( 1 \rightarrow 2 \rightarrow 1 \) onto \( 2 \rightarrow 3 \rightarrow 4 \)) the
proper trajectories itself do not need to be compared, because
the tactical pattern will not match anyway. Consequently,
before any trajectories will be compared, a valid mapping
from sequence s₁ to s₂ has to be found. The result is a
set of possible matching sequences that can be compared
on trajectory level. The mapping algorithm is depicted in
Algorithm 1.
A window is slided over the past stream. Within each position
of the window is the stream, the algorithm tries to find the
longest valid matching. The current position is processed, if a
non-valid matching has been found. With a mapping set M,
a mapping m from x to y is valid, if
\[
m(x, y) = \{(x \to y) | \exists x, y \in M \text{ with } x \to u \land v \to y; u, v \in M\}
\]
holds true. If this expression cannot be achieved, the algorithm
fails for this window position. If the length of the valid
subsequence within that window is long enough (i.e. greater
than a defined minimum, e.g. 4), then the mapping is added
to the list of mappings m.
The search for a suitable matching between a recently played
sequence and a sequence played somewhere in the past,
a linear complexity is inevitable. However, the amount of
necessary comparisons can nevertheless be massively reduced
giving the following three assumptions. A tactical pattern is
either finished or canceled, if
1) the ball leaves the playfield,
2) the referee stops the game (offsides, goals etc.) or
3) the opponent team gets in possession of the ball.

with these three preconditions, the overall sequence gets
pruned, such that mappings do not need to be found be-
{}ond such borders. Imagine an overall played sequence of
121345231 (Length = 9). Finding a mapping of length 4
requires at least 9 - 4 + 1 = 6 necessary searches for
mappings (1213, 2134, 1345, 3452, 4523 and 5231). Certainly,
if the stream is 1213|45231, with | being one of the given
circumstances, only (4 - 4 + 1) + (5 - 4 + 1) = 3 mappings
had to be calculated (1213, 4523 and 5231). Even for this
simple and small example, the necessary computations could
be reduced by a factor of 3.

C. Transformations
Rotation and scaling are both normal linear transformation
that can easily be assured by matrix-vector multiplication.
However, translation is not a linear transformation, because
it needs a vector to be added. The following subsections will
explain the transformations.
1) Rotation: In order to rotate a vector \((x, y)^T\) for \(\alpha\)
degrees counter-clockwise, the following calculation needs to
be performed:
\[
\tilde{a}(t) = \left(\begin{array}{cc}
\cos(\alpha) & \sin(\alpha) \\
-\sin(\alpha) & \cos(\alpha)
\end{array}\right) \cdot \left(\begin{array}{c}
f_x(t) \\
f_y(t)
\end{array}\right)
\]
The angle can be determined by normalizing the pattern acros
the line that connects the start of the pattern sequence and
the end of the pattern sequence. Consequently, the pattern is
aligned with the x-axis and the y-coordinate of the start and
end point is 0.
2) Scaling: In order to scale the whole image of the
 trajectories, the maximum and minimum possible x and y
values need to be found. Remember: these values needs to be
calculated after the trajectories have been rotated and translated.
When the proper values have been calculated, the scaling can
be applied by:
\[
\tilde{a}(t) = \left(\begin{array}{cc}
\frac{x_{\text{max}} - x_{\text{min}}}{2} & 0 \\
0 & \frac{y_{\text{max}} - y_{\text{min}}}{2}
\end{array}\right) \cdot \left(\begin{array}{c}
f_x(t) \\
f_y(t)
\end{array}\right)
\]
This transformation matrix scales the entire pattern to the unit
square \([0; 1]^2\).
3) Translation: As already told, translation is not an affine
linear transformation. To any vector \(\tilde{a}\), another vector \(\vec{x}\)
needs to be added. However, it would be useful to summarize all
transformation steps into one single instruction. With the
help of homogeneous coordinates, vectors can be translated
by a matrix multipication. A third variable is added to the
vector and the transformation matrix as an additional, third
dimension. Then, a translation of \((5, 4)^T\) can be expressed by
\[
\tilde{a}(t) = \left(\begin{array}{ccc}
1 & 0 & 5 \\
0 & 1 & 4 \\
0 & 0 & 1
\end{array}\right) \cdot \left(\begin{array}{c}
f_x(t) \\
f_y(t) \\
1
\end{array}\right)
\]
For a given example vector \((2, 1)^T\), the result would be
\((7, 5, 1)\) which is equivalent to \((7, 5)\) after dehomogenization.¹

¹dehomogenization is being performed by dividing \(x\) and \(y\) by the addi-
tional third coordinate, the homogeneous coordinate

---

**Algorithm 1: Sequence Mapping Algorithm**

**Data:** int `frame`, int `minimum`, PlayballStream `p`

**Result:** list of mappings `m`

sequence ← current window

if Sequence does not support minimum subsequence

∥ return;

∥ for `i = 0` to `p.size() - 2` ∥ frame

∥ initialize first source-target mapping in list ;

∥ Do current frame

∥ for `j = 1` to `frame`

∥ search if target is already in target mappings ;

∥ search if source is already in source mappings ;

∥ if source or target already in list then

∥ check if source-target mapping matches to

∥ current one in frame ;

∥ if not matching then

∥ break ;

∥ else

∥ add source-target mapping to current list ;

∥ if `j ≥ minimum subsequence`

∥ add mapping(mapping from, mapping to) ;

---

**Algorithm 1:** Sequence Mapping Algorithm

**Data:** int `frame`, int `minimum`, PlayballStream `p`

**Result:** list of mappings `m`

sequence ← current window ;

if Sequence does not support `minimum subsequence`

∥ return;

∥ for `i = 0` to `p.size() - 2` ∥ frame

initialize first source-target mapping in list ;

Do current frame

for `j = 1` to `frame`

search if target is already in target mappings ;

search if source is already in source mappings ;

if source or target already in list then

check if source-target mapping matches to

current one in frame ;

if not matching then

break ;

else

add source-target mapping to current list ;

if `j ≥ minimum subsequence`

add mapping(mapping from, mapping to) ;
Let sampled locations at feature points (gray). Figure 4 illustrates a normalized trajectory and its points. The corresponding distance of two equally sampled trajectories has to be compared. Unfortunately, traditional trajectory similarity measures [11] are not suitable for interaction analysis. After the preprocessing is done, two given sets of trajectories have to be compared. Unfortunately, traditional trajectory similarity measures [11] are not suitable for interaction analysis.

VI. CLASSIFICATION

After the preprocessing is done, two given sets of trajectories have to be compared. Unfortunately, traditional trajectory similarity measures [11] are not suitable for interaction analysis. All present trajectory matching models do not incorporate that particular objects perform the movements in a complex form of interaction. Hence, most of the already emerged models are not readily applicable to the given problem. Furthermore, while regarding moving object interactions, the actual trajectory of the particular objects is not very important. Even more important are specific points in the interaction, at which the objects interact. In the soccer scenario, such points are the locations of players while passing and the behaviour of opponents near ball possessors. The way that players move through the field is less important than the start and end points of their movement. Hence, the developed algorithms do not only consider the movements but more their points of interest.

A. Sampled Location Index Distance

After extracting important feature points within a potential pattern, this points can be transformed by the matrices given above and then be sampled to introduce fuzziness (a very high sampling rate would imply an Euclidean distance between the points). Figure 4 illustrates a normalized trajectory and its sampled locations at feature points (gray). Let τ be a trajectory and τ(n) be the sampled location (x_n, y_n) of τ at point number n. There is a total of k sampling points. The corresponding distance of two equally sampled trajectories τ₁ and τ₂ is then be calculated by

\[ D_s = \frac{1}{k \cdot r \sqrt{2}} \sum_{n=0}^{k} ||\tau_1(n) - \tau_2(n)|| \]

B. Object Heading Distance

A second distance measurement is the difference of the object’s heading angles. Given two set of angles A₁ and A₂ for two trajectories τ₁ and τ₂ respectively, the heading distance is defined by the average difference between the angles of both trajectories with k feature points:

\[ D_\alpha = 1 - \frac{1}{k} \sum_{n=0}^{k-1} \cos(|A_1(n) - A_2(n)|) \]

with Aᵢ(n) defined as the counter-clockwise angle between two segments at a feature point within trajectory i:

\[ \cos^{-1} \left( \frac{S_{n+1,x} - S_{n,x}}{S_{n+1,y} - S_{n,y}} \times \frac{S_{n+1,x} - S_{n+1,y}}{S_{n+1,y} - S_{n+1,y}} \right) \]

being the angle of A at position n, between sampling point S₁ and S₂. Figure 5 illustrates two trajectories. The angle distance is the average of the angles between the particular partitions (partitions are denoted by the points that separates the trajectories).

C. Euclidean Trajectory Distance

A third distance measurement is an Euclidean distance between two trajectories τ₁ and τ₂. The Euclidean distance of two trajectories τ₁ and τ₂, sampled at k discrete points is defined by

\[ D_e = \frac{1}{k} \sum_{n=0}^{k} ||\tau_1(n + \lambda_1) - \tau_2(n \cdot \phi + \lambda_2)|| , \phi = \frac{\text{length}(\tau_2)}{\text{length}(\tau_1)} \]

with \( \phi \) being the adaptive factor to sample both trajectories uniformly and \( \lambda_i \) being the offset of the pattern in trajectory i. The average Euclidean distance between two trajectories is given by the distance between sampled points of the trajectories. The sampling rate should be chosen adequate in order to approximate the result clearly.

VII. EXPERIMENTAL RESULTS

The algorithms have been taken for test on both synthetic and real positioning data. This section outlines the results that have been gained on real positioning data.

A. Precision of ball possession

This section analyzes the accuracy of the ball possession coefficient. Tests were run with different threshold values onto the same real data. For each threshold value, 200 randomly chosen ball possessions have been observed and checked on validity individually by examining the situations manually.
Figure 6 shows different threshold values and the corresponding precision and amount of remaining ball possessions. If no threshold value is defined, then no possession will be dropped, because the acceleration is not observed (1163 ball possessions in total). These ball possessions are only determined by the distance between the ball and the player, which did not necessarily touch the ball. With a small value of only 10 ms⁻¹ the number of dropped possessions shrinks to 896. With an increasing threshold value, the remaining valid ball possessions are steadily decreasing. The second curve shows the percentage of valid ball possessions within all remaining possessions. With a value of zero, about 80% of all possessions are valid (the rest is either assigned wrongly or not even assigned). With a threshold of 10, the validity increases rapidly. With a threshold of about 100 ms⁻¹ the precision of the algorithm reaches its maximum of about 91%. The remaining possessions were either sorted out in cause of low acceleration or because of too much noise within the position data.

B. Strength and Weaknesses of the Similarity Measures
Depending on the actual pattern, each similarity measure has its advantages and disadvantages. Recall the example from the beginning of the paper. Obviously, the Heading Distance gained the advantage. Sampled Location Distance could have been better and the Euclidean Distance did fairly bad. Figure 7(b) shows the plot for the normalized trajectory, on which the Heading Distance calculation has been performed. The selected sampling points are marked within the proper trajectory. As one can see, contrary to the beginning and end points, the distances between the inner points of the patterns are fairly high. Hence, the Euclidean Distance calculates low similarity although both patterns seem very similar.

Figure 7(a) shows the sampled feature points for the Sampled Location Distance. Obviously, the issue here is quite similar to the one of the Euclidean Distance. Start and end points of the pattern are shifted in y-direction. The inner points show relatively accurate performance contrary to the outer points. Figure 8 shows the plots for the calculation of the Heading Distance. For both patterns, the first angle is huge, while the second angle is very sharp. In Pattern 1 (8(a)) the first angle is about 320 degrees (which is certainly also very sharp, but not in counter-clockwise orientation) and the second angle is about 30 degrees. In Pattern 2 (8(b)) the first angle is about 340 degrees and the second angle 70 degrees. Since the Heading Distance does not explicitly distinguish the outer points (only a little bit by the angles), the problems from the other two similarity measures are no big issue here. Hence, the Heading Distance can be considered as the most applicable for the given scenarios.

C. Similarity Measures
This section evaluates the performance of the similarity measurement functions. Due to the subjectivity of similar patterns, an automatic analysis is not possible. Hence, the result of a real soccer game is presented and individual patterns are presented.

<table>
<thead>
<tr>
<th>Overall recording time</th>
<th>48 min</th>
</tr>
</thead>
<tbody>
<tr>
<td># patterns found</td>
<td>36</td>
</tr>
<tr>
<td># clusters found</td>
<td>12</td>
</tr>
<tr>
<td>Minimum Subsequence (for trajectory matching)</td>
<td>4</td>
</tr>
<tr>
<td>Maximum Found Mapping</td>
<td>6</td>
</tr>
<tr>
<td>Thresholds (SLD, HD, ED)</td>
<td>(0.8, 0.7, 0.6)</td>
</tr>
<tr>
<td>Minimum Pattern Time</td>
<td>2 seconds</td>
</tr>
<tr>
<td>Maximum Pattern Time</td>
<td>3.5 seconds</td>
</tr>
</tbody>
</table>

TABLE I  
Statistics from soccer game

Fig. 6. Drops from possession stream
and the cluster emerges more and more. Figure 9 shows the roughly reduced result of the analysis process (nodes normally normally time information and are are reduced to simple dots, percentage information on the edges is removed). Figure 10 shows a set of similar patterns. Three players are contributing to the pattern. The first player passes the ball to its teammate, which immediately passes the ball back within a sharp angle to the first player. When the ball comes back, the ball is immediately passed to a player standing further away. Another obvious characteristic is, that the trajectory of the returning ball is not as long as the trajectory where the ball first leaves the player. The similarity between 19:40:03+7 and 19:20:09+7 and 19:32:48+3 the similarity is 87% for Sampled Location Distance, 82% for Heading Distance and 70% for Euclidean Distance.

VIII. CONCLUSIONS AND FUTURE WORK

This paper describes an efficient way to mine frequent interaction patterns from moving objects in data streams. Once the particular positions has been sampled, the spline method interpolates the entire trajectory accurately. Once the particular positions has been sampled, the interaction patterns from moving objects in data streams. This paper describes an efficient way to mine frequent interactions between stock-receipt, forklifts and goods play a key role for analyzing the supply chains. Certainly, the possession of an interaction objects must be modified for more than one object, but the actual idea of this paper remains the same. Hence, the processes within the stock may be optimized for throughput in goods and time.

REFERENCES