Behavior Classification with Self-Organizing Maps

Michael Wünstel[1], Daniel Polani[2,1⋆], Thomas Uthmann[1], and Jürgen Perl[1]

[1] Institut für Informatik, Johannes Gutenberg-Universität Mainz, Germany
[2] Institut für Neuro- und Bioinformatik, Universität Lübeck, Germany
{wuen,polani,uthmann,perl}@Informatik.Uni-Mainz.DE

Abstract. We describe a method that applies Self-Organizing Maps for direct clustering of spatio-temporal data. We use the method to evaluate the behavior of RoboCup players. By training the Self-Organizing Map with player data we have the possibility to identify various clusters representing typical agent behavior patterns. Thus we can draw certain conclusions about their tactical behavior, using purely motion data, i.e. logfile information. In addition, we examine the player-ball interaction that give information about the players' technical capabilities.

1 Introduction

The goal of our work is to present a method to detect characteristic features of trajectories. The method is illustrated by analyzing the actions of Robocup players on a purely behavioral level. Only logfile information and no knowledge about implementation or inner states of the players is used.

In our analysis we use *Kohonen's Self-Organizing Map*[1] (Kohonen, 1989). The SOM is a data analysis method inspired by the structure of certain cortex types in the brain. It is able to identify different clusters in high-dimensional data and to project ("map") the data of the different clusters to a two-dimensional grid respecting the topology, i.e. the neighborhood structure of the data. By this a visualization of the data can be easily obtained. The mapping and the visualization capability is an advantage of the SOM as compared to standard statistical methods; in particular, the SOM does not just separate different clusters, but it also resolves the inner structure of the clusters. Mathematically it is related to the principal surface models for data distributions known from statistics (Ritter et al., 1992), though, unlike the SOM, the latter are not designed to handle data sets decomposing into different clusters.

For a trained SOM, each data point from the high-dimensional space is projected onto an element of the two-dimensional SOM grid, a *SOM unit*. Nearby data points are projected to the same unit or a unit close by in the grid. In turn, every SOM unit represents a vector in the high-dimensional data space, such that the SOM can be regarded as an embedding of the two-dimensional SOM grid into the high-dimensional data space. Fig. 1 gives a general impression of the form of such an embedding for RoboCup player trajectories.

⋆ corresponding author
[1] in the following abbreviated by *SOM*

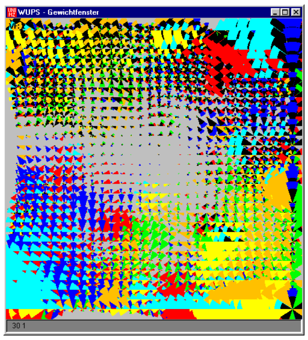


Fig. 1. SOM of 30 × 30 units trained with SFR-vectors in sector representation (see Secs. 2 and 2.2 for the definition of SFR-vectors and Sec. 3.1 for a description of the sector representation)

In this paper, each SOM unit will either represent a trajectory slice (i.e. a short section of a trajectory) of a single player or a trajectory slice of the combined motion of player and ball. Before the SOM can do that, it first has to be trained in order to learn to represent a set of given motion slices in an optimal way. This way we obtain a two-dimensional map of units, each unit encoding a trajectory slice. The trained map can be employed to classify the individual trajectory slices or to examine the behavior of a single player in general. One property of the trained SOM is especially valuable: neighboring units describe similar paths of motion. With the help of a graphical display it is then possible to get an overview over the behavior patterns of a single player. The paper will give an introduction into the SOM algorithm, introduce the representation for player trajectories and present its application to the analysis of player behavior in a game from the RoboCup 1999 competition at Stockholm.

2 Self-Organizing Maps and Trajectory Clustering

The problem of storing trajectories with the help of SOMs can be solved in different ways. One method is to examine the trace of the winning units like the trace of a elementary particle in a cloud chamber. One would then compare the order of activated units with some reference order. This method has been used successfully for example in (Carpinteiro, 1999) to recognize instances of a theme in a piece of music of J.S. Bach. It also can be used in speech recognition to trace the order of phonemes (Mehler, 1994; Kangas, 1994). Another way has been used by Chappell and Taylor (Chappell and Taylor, 1993) who used Leaky Integrator Units. These units can store the units' activations for a while and therefore are able to represent temporal information. The method presented in our paper, however, directly stores a complete trajectory slice itself in each SOM unit. Here we use the *spatially focused representation* (*SFR*), where we consider a trajectory as a simple sequence of spatial data: The time intervals are fixed and discrete and the spatial data are continuous. With the so-called *enhanced spatially focused representation* (*ESFR*) we were able to combine two simultaneous trajectories (of one player and the ball) and thus examine the player-ball interactions. The data representation is similar to SFR. We use these methods to examine the short-term (micro-)behavior of individual RoboCup players.

2.1 The Basic SOM Model

A Self-Organizing Map with m inputs is a set $\mathcal{N}$ of units, where every unit i is assigned a *weight vector* $\mathbf{w}_i \in \mathbb{R}^m$ (Kohonen, 1989; Polani and Uthmann, 1992; Ritter et al., 1992). Furthermore on $\mathcal{N}$ there exists a metric $d_{\mathcal{N}}$ which defines a distance $d_{\mathcal{N}}(i,j)$ for each two units in $\mathcal{N}$. An *input (training) signal* is a vector $\mathbf{x} \in \mathbb{R}^m$. The similarity between an input signal $\mathbf{x}$ and a unit's weight vector $\mathbf{w}_i$ is given by their Euclidean distance $\| . \|$. An input signal $\mathbf{x}$ is said to *activate* a (typically unique) unit i^*, for which

$$\| \mathbf{w}_{i^*} - \mathbf{x} \| \quad \leq \quad \| \mathbf{w}_i - \mathbf{x} \|, \ \forall \ i \in \mathcal{N}$$

holds, i.e. that unit whose weight vector has the smallest Euclidean distance to the input signal. The training of the map involves three steps:

1. **Initialization:** the values of the weights $\mathbf{w}_{il}$, $l = 1, \ldots, m$ of every unit i can be randomly chosen.
2. **Stimulus and Response:** choose an input vector out of a training set according to some probability distribution and detect the activated (*winning*) unit i^*.
3. **Adaption:** modify the weights $\mathbf{w}_i$ of all the units in a certain $\mathcal{N}$-neighborhood of the winning unit i^* towards the winning unit i^*. The degree of the $\mathcal{N}$-neighborhood is determined by the distance function (metric) $d_{\mathcal{N}}$.

Steps 2 and 3 are repeated arbitrarily often. Formally, the learning rule for the adaption of the weights of a unit i at iteration t is given by:

$$\Delta \mathbf{w}_i(t) \ := \epsilon(t) \cdot h_t(i^*, j) \ \big(\mathbf{x}(t) - \mathbf{w}_i(t)\big) \tag{1}$$

$\mathbf{x}(t)$	the training input at iteration t
$i^* = i^*(\mathbf{x}(t))$	the winning unit at iteration t
$\epsilon(t)$	the learning rate at iteration t. The learning rate is a monotonously decreasing function with $0 < \epsilon(t) < 1$.
h_t	the activation profile at iteration t. Unit weights close to the winning unit i^* are more attracted to the the winning unit i^*. An example for h is found in Eq. (2) (Sec. 3.1).

The weights of units are then changed according to $\mathbf{w}_i(t+1) \ := \mathbf{w}_i(t) + \Delta \mathbf{w}_i(t)$. If the units in $\mathcal{N}$ are now arranged as a two-dimensional lattice, above algorithm generates a projection of the high-dimensional input space to the two-dimensional lattice. As neighboring units of a trained SOM represent similar weights, it is possible to identify clusters representing similar data. In Fig. 1 the similarity has been made visible by a direct plot of the units' weights[2]. It can be seen that units with similar weights form certain areas (Kohonen, 1989). If the trained SOM is then fed with data, the frequency of the units' activations can help to draw conclusions about the examined data set.

In Figs. 3(a) and 3(b) a dot is plotted for every activation of a unit. Since the same unit may be activated multiple times, the dot positions are slightly perturbed for each activation event to obtain a visualization of the activation frequencies ("importance") of a certain region in the unit lattice.

[2] Sector representation is used as shown in Fig. 2 and described in detail in Sec. 3.1.

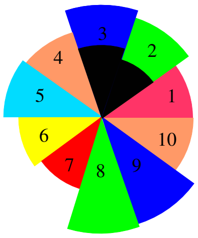
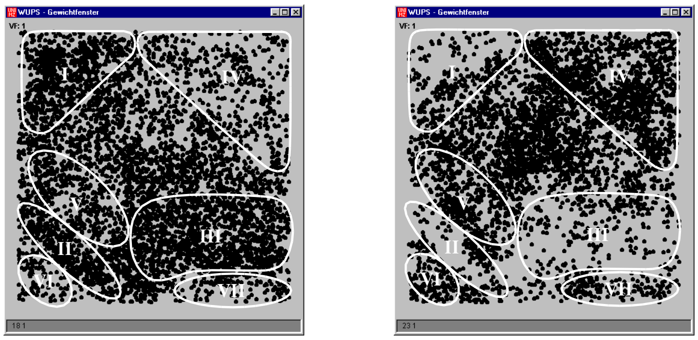


Fig. 2. Weight vector in sector representation



(a) CMU-player 11

(b) MRB-player 7

Fig. 3. SOM activations for SFR-vectors

2.2 Spatially Focused Representation (SFR)

The training input (a trajectory) is given as a sequence of τ difference ("velocity") vectors $\mathbf{u}$. Every difference vector is calculated from two successive position vectors $\mathbf{p}$ of a RoboCup player:

$$\begin{aligned} \mathbf{p} &= (\mathbf{p}_t)_{t=t_0-t', t'=\tau,\ldots,0}, \;\; \mathbf{p} \in \mathbb{R}^2 \\ \mathbf{u} &= (\mathbf{u}_t)_{t=t_0-t', t'=\tau-1,\ldots,0} \\ \mathbf{u}_t &= \mathbf{p}_t - \mathbf{p}_{t-1} \end{aligned}$$

In our experiments we use the sequence length $\tau = 5$. A training input $\mathbf{x}(t)$ at simulation time step $t = t_0$ is then given by

$$\begin{aligned} \mathbf{x}(t = t_0) &= (\mathbf{u}_{t_0-4}, \mathbf{u}_{t_0-3}, \mathbf{u}_{t_0-2}, \mathbf{u}_{t_0-1}, \mathbf{u}_{t_0-0})^T \\ &= \begin{aligned} &(\mathbf{p}_{t_0-4} - \mathbf{p}_{t_0-5}, \\ &\quad \mathbf{p}_{t_0-3} - \mathbf{p}_{t_0-4}, \\ &\quad\quad \mathbf{p}_{t_0-2} - \mathbf{p}_{t_0-3}, \\ &\quad\quad\quad \mathbf{p}_{t_0-1} - \mathbf{p}_{t_0-2}, \\ &\quad\quad\quad\quad \mathbf{p}_{t_0-0} - \mathbf{p}_{t_0-1})^T \end{aligned} \end{aligned}$$

We call such a vector a *SFR-vector*. This method has already been introduced in (Boll, 1999; Wünstel et al., 1999).

2.3 Enhanced Spatially Focused Representation (ESFR)

Whereas in the SFR a training input vector is built exclusively by the difference vectors $\mathbf{u}$ of a RoboCup player, here ball-ball and player-ball difference vectors are used. Thus the trajectory slices of the player and the ball are completely described save a possible translation about the origin. The starting point of a trajectory is the player-ball difference vector at time step t followed by pairs of ball-ball and player-ball difference vectors of the next time steps:

$$\begin{aligned}
\mathbf{p}_{\text{ball}} &= (\mathbf{p}_{\text{ball}_t})_{t=t_0-t',t'=\tau-1,\dots,0},\ \mathbf{p}_{\text{ball}} \in \mathbb{R}^2 \\
\mathbf{p}_{\text{player}} &= (\mathbf{p}_{\text{player}_t})_{t=t_0-t',t'=\tau-1,\dots,0},\ \mathbf{p}_{\text{player}} \in \mathbb{R}^2
\end{aligned}$$

$$\begin{aligned}
\mathbf{u}_{\text{ball}} &= (\mathbf{u}_{\text{ball}_t})_{t=t_0-t',t'=\tau-2,\dots,0} \\
&\quad \mathbf{u}_{\text{ball}_t} = \mathbf{p}_{\text{ball}_t} - \mathbf{p}_{\text{ball}_{t-1}} \\
\mathbf{u}_{\text{player}} &= (\mathbf{u}_{\text{player}_t})_{t=t_0-t',t'=\tau-1,\dots,0} \\
&\quad \mathbf{u}_{\text{player}_t} = \mathbf{p}_{\text{player}_t} - \mathbf{p}_{\text{ball}_t}
\end{aligned}$$

The training input $\mathbf{x}(t)$ at time step $t = t_0$ is given by[3]

$$\mathbf{x}(t=t_0) = \begin{pmatrix} \mathbf{u}_{\text{player}_{t_0-\tau+1}} \\ \mathbf{u}_{\text{ball}_{t_0-\tau+2}} \\ \mathbf{u}_{\text{player}_{t_0-\tau+2}} \\ \vdots \\ \mathbf{u}_{\text{ball}_{t_0-1}} \\ \mathbf{u}_{\text{player}_{t_0-1}} \\ \mathbf{u}_{\text{ball}_{t_0}} \\ \mathbf{u}_{\text{player}_{t_0}} \end{pmatrix} = \begin{pmatrix} \mathbf{p}_{\text{player}_{t_0-\tau+1}} & - \mathbf{p}_{\text{ball}_{t_0-\tau+1}} \\ \mathbf{p}_{\text{ball}_{t_0-\tau+2}} & - \mathbf{p}_{\text{ball}_{t_0-\tau+1}} \\ \mathbf{p}_{\text{player}_{t_0-\tau+2}} & - \mathbf{p}_{\text{ball}_{t_0-\tau+2}} \\ & \vdots \\ \mathbf{p}_{\text{ball}_{t_0-1}} & - \mathbf{p}_{\text{ball}_{t_0-2}} \\ \mathbf{p}_{\text{player}_{t_0-1}} & - \mathbf{p}_{\text{ball}_{t_0-1}} \\ \mathbf{p}_{\text{ball}_{t_0}} & - \mathbf{p}_{\text{ball}_{t_0-1}} \\ \mathbf{p}_{\text{player}_{t_0}} & - \mathbf{p}_{\text{ball}_{t_0}} \end{pmatrix}$$

We call such a vector an *ESFR-vector*.

Example 1. With $T = 49$, $\tau = 4$ we have:

$$\begin{aligned}
\mathbf{p}_{\text{player}_{46}} &= (18,5)^T & \mathbf{p}_{\text{ball}_{46}} &= (22,7)^T \\
\mathbf{p}_{\text{player}_{47}} &= (25,7)^T & \mathbf{p}_{\text{ball}_{47}} &= (34,11)^T \\
\mathbf{p}_{\text{player}_{48}} &= (37,7)^T & \mathbf{p}_{\text{ball}_{48}} &= (41,12)^T \\
\mathbf{p}_{\text{player}_{49}} &= (40,7)^T & \mathbf{p}_{\text{ball}_{49}} &= (46,11)^T
\end{aligned}$$

$$\begin{aligned}
\mathbf{u}_{\text{player}_{46}} &= (-4,-2)^T & & \\
\mathbf{u}_{\text{player}_{47}} &= (-9,-4)^T & \mathbf{u}_{\text{ball}_{47}} &= (12,4)^T \\
\mathbf{u}_{\text{player}_{48}} &= (-4,-5)^T & \mathbf{u}_{\text{ball}_{48}} &= (7,1)^T \\
\mathbf{u}_{\text{player}_{49}} &= (-6,-4)^T & \mathbf{u}_{\text{ball}_{49}} &= (5,-1)^T
\end{aligned}$$

$$\mathbf{x}(t=49) = (-4,-2,12,4,-9,-4,7,1,-4,-5,5,-1,-6,-4)^T$$

[3] Note that there is always one more player-ball entry than ball-ball entry.

3 Experiments and Results

In our experiments we test the SFR method with the isolated data of two opposing RoboCup players. We choose two offensive players with approximately equivalent roles from the game Carnegie Mellon United (CMU) versus Mainz Rolling Brains (MRB) at the RoboCup WorldCup 1999 in Stockholm (simulation league): CMU left wing forward player 11 and MRB left wing forward player 7. For the experiment with the ESFR method we use ESFR-vectors of all players of the same game. Only those ESFR-vectors were used whose player-ball distance vector value at the beginning of the trajectory is smaller than 30 simulator units, since we are only interested in player behavior related to the ball.

In this procedure, the SOM solves the task of organizing the data in a way suitable to projection to 2-dimensional space. This representation has then to be "interpreted", that is, to be translated into a human-readable notion. The units belonging to the different clusters have to be identified; this step can be done in a semi-automated fashion based on distances of unit weight vectors (Boll, 1999), which can not be described here due to space limitations. The different clusters were then tagged by human inspection. This processing step was not automated since that would have require a linguistic concept for the of treatment of geometric notions outside the scope of our work.

Here it is important to emphasize the following point: in principle all possible movement classes could have been enumerated and the vectors falling into those classes according to some suitable criterion could have been counted instead of using the SOM. However, the SOM is able to detect clusters that do not belong to a clear-cut linguistic category, and, in addition, it does not introduce unnecessary classes. Above that, for every class, the SOM maps the data variability to areas on the grid instead of just breaking down the data to a number of events of a certain class.

3.1 Spatially Focused Representation

The SOM consists of a lattice of 30×30 units. The distance between two units i, j is given by their Euclidean distance, i.e.

$$d_{\mathcal{N}}(i,j) = \parallel \mathbf{i} - \mathbf{j} \parallel_2 ,$$

where $\mathbf{i}$,$\mathbf{j}$ are the position vectors of unit i and j in the lattice[4].
The activation profile h_t (see Eq. (1)) is a cone function, i.e.

$$h_t(i,j) = \begin{cases} 1 - \frac{\|\mathbf{i}-\mathbf{j}\|_2}{r_h(t)} & \text{if} \quad \parallel \mathbf{i} - \mathbf{j} \parallel_2 < r_h(t) \\ 0 & \text{else} \end{cases} . \tag{2}$$

The activation radius r_h is given by: $r_h(t) = 40\ (1-0.0006)^t$, with the constraint that $r_h(t) \geq 1.7$.
The learning rate $\epsilon(t)$ is given by $\epsilon(t) = 0.65\ (1 - 0.001)^t$, with the constraint that $\epsilon(t) \geq 0.15$. After the SOM is trained with the SFR-vectors of both players,

[4] Note that the metric here is taken on the two-dimensional grid space which is distinct from that of the vectors $\mathbf{w}$ from Sec. 2.1.

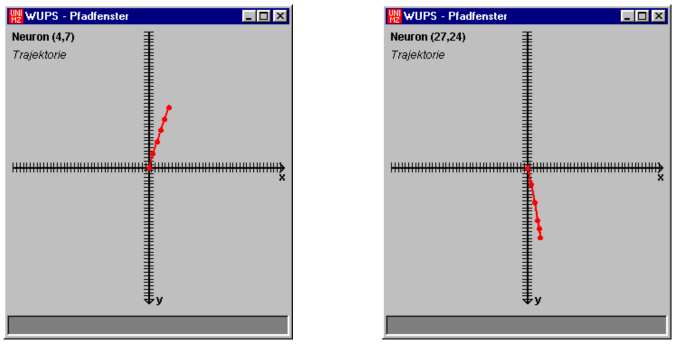


Fig. 4. Player trajectories reconstructed from a weight vector

the weight vector of every unit represents one SFR-vector which describes a trajectory slice. We use the sequence length $\tau = 5$. Shorter sequences do not provide enough structure to distinguish interesting separate behaviors, and in longer sequences several micro-behaviors tend to be combined into sequences that lead to unnecessary multiplication of classes.

In Fig. 1 the weight vectors of the units after the training are plotted as groups of sectors. Every weight vector is represented by 10 sectors (see Fig. 2), one sector for every component of the unit's weight vector. The radius of each sector represents the value of that component, negative values are marked by a black component, showing the self-organization of the SOM clusters.

In Fig. 4 the trajectory slices for two different example weight vectors are shown. The beginning of a trajectory is placed in the origin. A trajectory is hence the translation of a part of the corresponding movement of a RoboCup player to the origin. The dots represent the player position at successive simulator time steps.

In the next step the different behavior of the players will be examined. Therefore the SOM gets activated with all the SFR-vectors of one of the players. Every activation of a unit is plotted as a dot. It is slightly perturbed for easier identification of multiple unit activations. This procedure is repeated for the other player. The Figs. 3(a) and 3(b) show the resulting SOM activations. After the SOM clusters and maps the input data, the tagging of the clusters (as translation into human-understandable notions) has to be done by a human.

If one compares the activation zones with trajectories like in Fig. 4 one can draw some player specific conclusions: a large part of the forward movements (zones II, V, VI) of the CMU player are of intermediate speed (zone II). His sideways movements (zones I & III) are also carried out mainly at intermediate speed. The back movements hardly took place. This can be explained with the superiority of CMU.

The forward movements of MRB player are carried out slowly (zone V) or rapidly (zone VI). Sideways movements are rare (only few activations in the zones I & III). A large part of the right sideways movements are fast (zone VII). Back movements here are frequently happening (zone IV). This can either be explained again with the superiority of CMU or with a reduced action radius (as

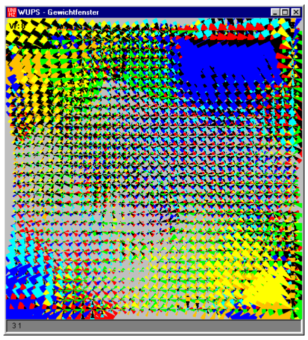


Fig. 5. SOM of 30 × 30 Units trained with ESFR-vectors

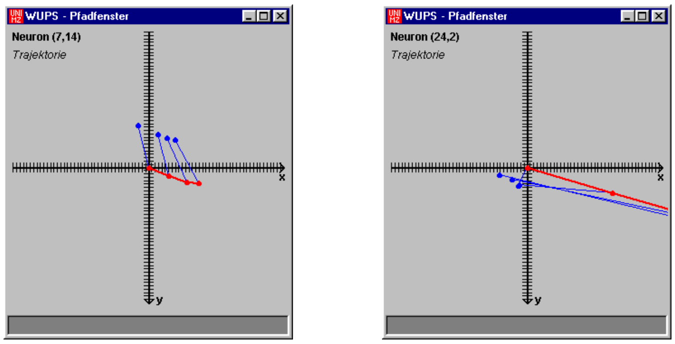
the MRB team plays with 5 offensive players in a row) which let him go back to a waiting position. While the SOM extracted all above information from purely behavioral (i.e. logfile) data, inspection of the agent code confirms that many of the found behavior patterns are indeed implemented in the agent. Our trajectory analysis therefore provides a possibility to reconstruct behavior patterns without knowing implementation details. In addition, it provides a possibility to detect behavior patterns not explicitly implemented but emerging from the collective agent dynamics.

3.2 Enhanced Spatially Focused Representation

The SOM again consists of a lattice of 30×30 units. The distance between two units i, j is given by their Euclidean distance in the lattice. The activation profile is a cone function (Eq. (2)) where the activation radius r_h is given by $r_h(t) = 40\ (1 - 0.00005)^t$, with the constraint that $r_h(t) \geq 1.7$. The learning rate is given by $\epsilon(t) = 0.35\ (1 - 0.00005)^t$, with the constraint that $\epsilon(t) \geq 0.07$. The SOM is trained for $100,000$ iterations with the ESFR-vectors from the same RoboCup match as before. In Fig. 5 the weight vectors of the units after the training are plotted again as groups of sectors. Every weight vector is represented by 14 sectors, one sector for every component of the unit's weight vector. The radius of each sector again represents the value of that component. In Fig. 6 two example trajectories of two weight vectors are presented. The sequence length is $\tau = 4$.

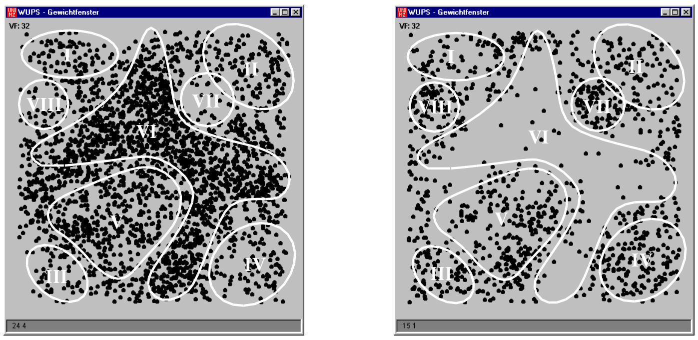
To examine the technical capabilities of the teams, the trained SOM is activated with the activation vectors of the teams. Every activation of a unit is again plotted as a dot. Figs. 7(a) and 7(b) show the activation distributions. Inspection of Fig. 5 and of player-ball trajectories like in Fig. 6 allows again to to draw some conclusions about the players' capabilities:

The trajectory analysis shows very explicitly that the CMU team has much more ball contacts than the MRB team. We distinguish between the techniques dribbling, passes and "near-ball" game. Dribbling trajectories can be identified whenever player and ball are in phase and the distance between ball and player is only few units. In pass trajectories the player only slightly changes his position and the ball leaves the player rapidly. Near-ball game consists of trajectories where the player changes his position for only few units and the ball remains close to the player.



(a) Ball is dribbled (b) Ball is passed

Fig. 6. Player-ball trajectories reconstructed from a SOM weight vector. Note that the ball and not the player is initially placed in the origin. The player is denoted by dark grey, the ball by light grey dots. Player and ball positions corresponding to the same simulator time step are connected by a line.



(a) CMU-player 11 (b) MRB-player 7

Fig. 7. SOM activations for ESFR-vectors

If we examine the dribbling zones (zone VI, VII & VIII) we recognize that CMU controls dribbling in a extraordinary way (zone VI). In contrast to that MRB dribbling occurs seldom and is of lower quality (zones VII and VIII). The activation frequency for the two players differs in zones representing passes (pass right: zone I, pass forward: zone II, pass back: zone III & pass back: zone IV) are examined. The near-ball game (zone V) of CMU is more evenly distributed, at MRB it is more focused on certain patterns.

4 Conclusion and Outlook

The paper introduced and examined the use of SOMs for the examination of the micro-behavior and motion of individual RoboCup players. As opposed to other SOM trajectory analysis methods, in our method each SOM unit encodes not just single states of a trajectory slice, but a complete trajectory slice instead.

Requiring only behavioral (i.e. logfile) data, the method proved very useful to identify short-term behavior patterns explicitly implemented in the agents or emerging during the game. The method shown is easily generalizable to tasks outside of RoboCup or collective agent systems. Its potential covers a wide selection of applications, whereever time-dependent patterns have to be analyzed. In the future, we plan to extend the approach to the analysis of more complex RoboCup scenarios and to other collective agent systems. Further we wish to investigate the potential of the method to uncover causal dependencies between behavior patterns.

4.1 Acknowledgements

We would like to express our gratitude to the anonymous reviewers for their very detailed and helpful comments.

References

Boll, M., (1999). *Analyse von Verhaltensprozessen mit Hilfe Neuronaler Netze*. Master's thesis, Johannes Gutenberg-Universität Mainz.

Carpinteiro, O. A. S., (1999). A hierarchical self-organizing map model for sequence recognition. *Neural Processing Letters*, 9:1–12.

Chappell, G. J., and Taylor, J. G., (1993). The Temporal Kohonen Map. *Neural Networks*, 6:441–445.

Kangas, J., (1994). *On the Analysis of Pattern Sequences by Self-Organizing Maps*. PhD thesis, Helsinki University of Technology, Espoo, Finland.

Kohonen, T., (1989). *Self-Organization and Associative Memory*, vol. 8 of *Springer Series in Information Sciences*. Berlin, Heidelberg, New York: Springer-Verlag. Third edition.

Mehler, F., (1994). *Selbstorganisierende Karten in Spracherkennungssystemen*. Dissertation, Institut für Informatik, Johannes Gutenberg-Universität Mainz.

Polani, D., and Uthmann, T., (1992). Neuronale Netze – Grundlagen und ausgewählte Aspekte der Theorie. Technical Report 2/92, Institut für Informatik, Universität Mainz.

Ritter, H., Martinetz, T., and Schulten, K., (1992). *Neural Computation and Self-Organizing Maps: An Introduction*. Reading, MA: Addison-Wesley.

Wünstel, M., Boll, M., Polani, D., Uthmann, T., and Perl, J., (1999). Trajectory Clustering using Self-Organizing Maps. In Sablatnög, S., and Enderle, S., editors, *Workshop RoboCup at KI'99 in Bonn, Germany*, Report 1999/12 ISSN: 1438-2237, 41–46. SFB 527 Ulm University.