Anomaly Detection and Localization for Cyber-Physical Production Systems with Self-Organizing Maps

Alexander von Birgelen and Oliver Niggemann

Ostwestfalen-Lippe University of Applied Sciences, Institute Industrial IT,
Langenbruch 6, 32657 Lemgo, Germany
{alexander.birgelen,oliver.niggemann}@hs-owl.de
http://www.init-owl.de

Abstract. Modern Cyber-Physical Production Systems provide large amounts of data such as sensor and control signals or configuration parameters. The available data enables unsupervised, data-driven solutions for model-based anomaly detection and anomaly localization: models which represent the normal behavior of the system are learned from data. Then, live data from the system can be compared to the predictions of the model to detect anomalies and perform anomaly localization. In this paper we use self-organizing maps for the aforementioned tasks and evaluate the presented methods on real-world systems.

1 Introduction

Modern Cyber-Physical Production Systems (CPPS) evolve rapidly, become modular, can be parameterized and contain increasingly more sensors due to increasing product variety, product complexity and pressure for efficiency in a distributed and globalized production chain they become modular, can be parameterized and contain a growing set of sensors [1]. This also means it becomes more and more difficult to monitor the systems. Human operators often struggle to diagnose faults or anomalous behavior in the system in time, leading to system break down, unexpected downtime or degradation in product quality.

A common approach to detect the aforementioned scenarios is to construct models for a given system and compare the predictions of the model to the real system. Anomalous behavior is detected when the real system's behavior deviates from the model's predictions. Manual construction of system models by experts is usually time consuming, expensive and also difficult in today's evolving complex systems. Experts with the necessary knowledge are usually scarce and often times some of the necessary knowledge is not available at all. Modern CPPS often provide large amounts of data such as control signals, sensor signals and configuration parameters [10]. This allows the use of data-driven methods: models are learned from data and then used for various tasks such as anomaly detection and anomaly localization.

The Author(s) 2018
O. Niggemann and P. Schüller (Eds.), *IMPROVE – Innovative Modelling Approaches for Production Systems to Raise Validatable Efficiency*, Technologien für die intelligente Automation 8, https://doi.org/10.1007/978-3-662-57805-6_4

Live data from the system is compared to the predictions of the learned model. Deviations from the normal behavior are classified as anomalous. Once anomalies are found, the anomalous samples are presented to a reverse model to localize the anomalies. This provides a starting point for plant operators and experts to restore the system to normal working order, ideally before production losses occur.

In this paper we use self-organizing maps (SOM) to learn a systems normal behavior in an unsupervised manner. The learned SOM's are then used for both anomaly detection and anomaly localization. The contents of this paper are structured as follows: First, section 2 explains the general concept of self-organizing maps. Second, section 2.1 presents an approach to detect and localize anomalies within the signal domain of a system. Third, section 2.3 introduces an approach where timed automata are used to track the working point on top of the self-organizing map in order to detect anomalies in the time domain. Furthermore, the aforementioned approaches to anomaly detection and localization are applied and explained on the Institute Industrial IT's OPAK demonstrator [11] in section 3. Section 4 presents the conclusion and future points of research.

2 Self-Organizing Map

The self-organizing map (SOM) [5], also referred to as self-organizing feature map or Kohonen network, is a neural network that can be associated with vector quantization, visualization and clustering but can also be used as an approach for non-linear, implicit dimensionality reduction [17]. A SOM consists of a collection of neurons which are connected in a topological arrangement which is usually a two dimensional rectangular or hexagonal grid. The input data is mapped to the neurons forming the SOM. Each neuron is essentially a weight vector of original dimensionality but provides additional information such as its coordinates within the grid. All experiments in this paper use a two dimensional, non-toroidal rectangular lattice and the Euclidean distance measure as shown in Definition 1.

Definition 1. *The* $SOM = (M, G, d)$ *forms a topological mapping of an input space* $O \subset \mathbb{R}^m$, $m \in \mathbb{N}$ *and consist of*

- *a set of neurons* M.
- *each neuron* $n \in M$ *has a weight vector* $\mathbf{w}_n \in \mathbb{R}^m, m \in \mathbb{N}$.
- G *is a two-dimensional rectangular lattice in which the neurons* $n \in M$ *are arranged.*
- $d(\mathbf{x}, \mathbf{y})$ *is the distance measure to calculate the distance between two vectors* $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *which can for example be weight vectors and/or vectors in the input space. The euclidean distance is used for all models in this paper.*
- *an input sample* $\mathbf{o}_i \in \mathbb{R}^m, i \in \mathbb{N}, m \in \mathbb{N}$ *is mapped to the SOM through its best matching unit (BMU). The BMU is given by* $bmu(\mathbf{o}_i) = \operatorname{argmin}_{n \in M} d(\mathbf{o}_i, \mathbf{w}_n)$

One way to learn a SOM from data is a random batch training approach: the initial values of the neuron's weight vectors for the training can be randomly initialized or sampled from the training data to provide a diverse starting point for the

training process. Training takes place over a chosen amount of epochs. All samples from the training data are presented to the algorithm within one epoch. A best matching unit (BMU) is calculated for each input sample from the training data by finding the neuron which has the smallest distance to the sample. The BMU and all of its neighboring neurons, assigned through the topology and neighborhood radius, are shifted towards the input sample (Figure 1). Both the size of the neighborhood and strength of the shift decrease over time to help with convergence. In the end, each neuron of the SOM represents a part of the training data. Areas in the training input space with few examples are represented by few neurons of the SOM while dense areas in the input space are represented by a larger number of neurons. Usually, the number of neurons is chosen much smaller than the number of samples in the training data, effectively discretizing and reducing the training data to the most important samples.

Usually, the number of neurons is chosen much smaller than the number of samples in the training data, effectively discretizing and reducing the training data to the most important samples. The compact representation of the training data provided by the SOM is used in section 2.1 to detect anomalies by calculating the quantization error.

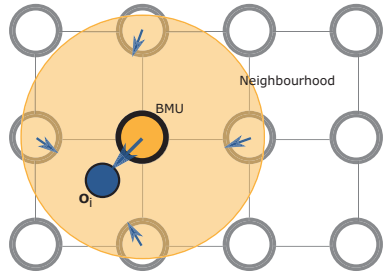

Fig. 1. Illustration of training procedure for a single sample.

The unified distance matrix (u-matrix) [14] of the SOM is great for visual identification of clusters in high-dimensional data. The distance to neighboring neurons according to the SOM's topology is computed and plotted as an image: the X and Y coordinates of the neurons represent the first two dimensions. The third dimension is given by the sum of distances to neighboring neurons as in definition 2. It calculates the sum of distances to neighbouring neurons according to the SOM's topology and visualizes clusters contained in the, usually high dimensional, training data. The neurons located on the borders of the non-toroidal SOM have fewer neighbours than the remaining neurons. Therefore, the summed distance is divided by the number of neighbours of the corresponding neuron to account for the different amounts of neighbouring neurons. A color gradient can be used instead of a third dimension as shown in Figure 3: valleys are represented by the color yellow, indicating a low distance between neighboring neurons. Ridges are represented by

the color red, indicating a high distance between neighboring neurons. The u-matrix can be defined as follows in Definition 2.

Definition 2. *for each neuron $n \in M$ and its associated weight vector $\mathbf{w}_n$, the u-matrix height is given by $U(n) = \sum_{k \in NN(n,G)} d(\mathbf{w}_n, \mathbf{w}_k)$, where $NN(n, G)$ is the set of neighboring neurons of n defined by grid G and $d(x, y)$ is the distance used in the SOM algorithm.*

The u-matrix representation illustrates why SOM's were chosen: SOM's tend to keep neurons with similar signal weights closely together, which results in a topographic landscape with valleys, where weights of neighbors are similar, and ridges, where weights of neighbors are not similar. Valleys represent regions where the contained neurons weight vectors are very similar. These valleys are separated by ridges which mark transitions between the different feature spaces. In section 2.3 we further explore this matter to detect anomalies within the time domain.

2.1 Anomaly detection with quantization error

The SOM can be used to detect anomalies by calculating the quantization error: small errors below a threshold are considered normal, while errors above are considered anomalous. Quantization error based approaches were already used for tasks such as network monitoring [6] and anomaly detection in industrial processes [12][3][13]. These works however, did not perform an anomaly localization and only [13] used the quantization error as a measure for system degradation.

The quantization error (Definition 3) of each sample is calculated by mapping it to the SOM to get its BMU. The distance of the sample to the BMU's weight vector is the quantization error.

Definition 3. *Using the notation from definition 1, the quantization error* **qe** *of an input sample $\mathbf{o}_i \in \mathbb{R}^m, i \in \mathbb{N}$ is given by the distance of the input sample to its BMU of the SOM:* $\mathbf{qe} = d(\mathbf{o}_i, bmu(\mathbf{o}_i))$.

The quantization errors for data that is not anomalous are usually greater than 0 due to the discretization of the SOM. A threshold for the quantization error above which an input sample is classified as anomalous is required. Manual selection of the threshold works but is usually unfeasible for practical applications. It is far more convenient to estimate the threshold from data: the quantization errors of the training data can be seen as a probability distribution and quantiles can be used to retrieve the threshold for the anomaly detection. The quantile can be adjusted and we will use the parameter τ with $\tau \in \mathbb{R}$ and $0.0 \leq \tau \leq 1.0$ within this paper. This can be adjusted to optimize the outcome of the anomaly detection: when labels are present τ can be used to fine tune the anomaly detection score. When the training data is perfect, meaning it contains only normal behavior, no sampling errors, no glitches in the sensors and no noise, then a τ of 1.0 is fine as this results in the maximum error for the threshold. However, training data is never perfect when working with real systems and data might contain a small portion of samples affected by noise and/or other effects. The maximum error might be too

large to effectively find anomalies. Setting τ to a value slightly smaller than 1.0 can increase the true positive rate of the anomaly detection at the cost of some false positives, depending on the use case and desired outcome.

2.2 Localization of anomalies

An additional step after the anomaly detection is to calculate the signal or sensor which is most likely to cause the anomaly. Anomaly localization is performed after an anomaly is detected: the observation found to be anomalous is fed through a reverse model to obtain the expected values for the signals. The deviations from the expected values can then be used to identify the signals related to the anomaly. The weight vector of each neuron of the SOM has the same dimensions as the input data and each element of the weight vector contains the value of its corresponding signal. Once an anomaly is detected, the input sample is again mapped to the SOM to retrieve the BMU. Now, the distance of each signal to the weight vector is calculated and the resulting signals and their distances are sorted in a descending order according to their distance. Since real world systems usually provide a large number of signals it is necessary to reduce the number of displayed signals. Therefore only the first n signals are displayed giving plant experts and operators a starting point to locate the anomaly and possible fault in the system and ultimately restore the system's normal working order. For the experiments in this paper we only consider the signal with the largest deviation (n=1).

2.3 SOM trajectory tracking with timed automata

Another way to utilize self-organizing maps for anomaly detection in industrial production processes is to track the trajectory of the working point on top of the SOM. Other works such as [7], [12] and [2] already performed a visual anomaly detection on different processes by tracking the trajectory of the working point on the SOM: the observations are mapped to their corresponding best matching units as soon as they are recorded. Over time, the path or trajectory of the BMU can be observed and deviations from the known path indicate anomalous behavior.

However, these works do not attempt to track the trajectory through the use of a mathematical model and only pursue a visual anomaly detection by plotting the trajectory on top of the SOM's u-matrix. In this section we use discrete timed automata to learn the trajectory during normal production and detect deviations from it afterwards. This provides explicit modeling of time which the SOM is unable to do alone.

Timed automata have proven to be a great tool to learn the normal behavior of a system and detect deviations from it. Discrete events are required to learn an automaton. They often cause mode changes within the system and the timing of these events is an important indicator for the health of the system. Timed automata are used to separate the system's modes and model the transitions and timing between the identified modes.

Discrete events can be directly extracted from changes in the binary control and sensor signals of the system. It is also possible to obtain discrete events through

thresholds for real-valued signals such as *temperature* $< 19^{\circ} C$ [4]. However, setting the thresholds and combinations of conditions for the continuous signals requires expert knowledge which is usually not available for real world automation systems. For unsupervised learning of these automata only binary control signals are used to obtain the discrete events, such as *HeaterOn* = *true*. Algorithms such as the bottom-up timed learning algorithm (BUTLA) [8] work in an unsupervised manner, and do not require additional expert knowledge.

A timed automaton generated by the aforementioned algorithm can be defined as described in Definition 4.

Definition 4. *a timed, probabilistic automaton is a tuple* $A = (S, s_0, \Sigma, T, \delta, P)$, *where*

- S *is a finite set of states where* $s \in S$.
- s_0 *is the initial state which can be given by the systems state at the start of the training.*
- Σ *is the set of discrete events. Events* $a \in \Sigma$ *is linked to the transitions of the automaton.*
- T *is the set of transitions with* $t \in T$ *and* $t = (s, a, s'))$, $s, s' \in S$ *are source and destination state,* $a \in \Sigma$ *is the trigger event of the transition.*
- *The timing constraint* $\delta : T \to I$ *assigns a time interval to a transition* $t \in T$, *where* I *is a set of time intervals. The time here usually refers to the elapsed time since the last event occurred.*
- P *is a set of probabilities: for each transition* $t \in T$ *probability* $p \in P$ *is calculated.*

The learned automaton can then be used to detect a variety of different classes of anomalies. This can for example be done using the anomaly detection algorithm (ANODA) [8] which can detect the following types of anomalies:

- **Unknown event / Wrong event sequence**: an event occurred which was not observed in the current state.
- **Timing error**: a transition occurred outside of the learned time bounds.
- **State remaining error**: when more time passed than for the latest event and the state is not a final state, then we have a state remaining error.
- **Probability error**: the probabilities of transitions for the new data are calculated and compared to the previously learned probabilities and an error is generated when deviations are too large.

As mentioned before, discrete events are needed to learn an automaton. One way to obtain these from the SOM is to interpret each neuron of the SOM as a binary signal: a neuron is active (or true) when the observation is mapped to it and false otherwise. This mapping can result in a large number of signals, depending on the size of the SOM. This usually leads to a large number of states in the automaton. Also, some neurons might never be activated by the training data leading to an unknown state detection in the automaton. Again, this can lead to a large number of false positives when new data is mapped to neurons which were previously not active but are direct neighbors to neurons previously activated by the training data.

To counteract these effects and ultimately reduce the number of states we group the neurons into a smaller number of clusters. The transitions between the clusters are then learned using a timed automaton.

SOM's tend to keep neurons with similar signal weights closely together, which results in a topographic landscape with valleys, where weights of neighbors are similar, and ridges, where weights of neighbors are not similar. This landscape can be visualized through the aforementioned u-matrix representation. Valleys represent regions where the contained neurons weight vectors are very similar. These valleys are separated by ridges which mark transitions between the different feature spaces. The valleys can be interpreted as stationary process phases while the ridges represent transient process phases [3].

Clustering algorithms from the image processing domain, such as the watershed transformation [9], can be used on the u-matrix representation of a SOM to identify the clusters in a mathematical way. This works analogous to rain falling on top of the u-matrix. The water runs from higher regions to the lower regions and consequently flooding the basins. When the water level gets high enough so two basins merge, a ridge forms which separates them. The watershed transformation dissects the u-matrix into different clusters, separated by the so-called watershed lines. Watershed lines separate the different basins and do not belong to any of the clusters. The implementation used here is the Vincent-Soille watershed algorithm which performs the watershed transformation in a non-recursive manner [15].

Subsequently, the samples of the training data are mapped to the SOM to get the corresponding cluster. The clusters are encoded using a one-hot encoding resulting in a binary vector with one element for each cluster. The value of the active cluster is set to true, while all other values are set to false. The time-stamps of the original samples and the binary vectors are then used to learn an automaton with the aforementioned BUTLA.

3 Experiments

In this section we apply the aforementioned approaches for anomaly detection and anomaly localization to one of our demonstrators. The Genesis demonstrator of the Institute Industrial IT sorts two different materials (conductive and non-conductive) from a magazine into their corresponding target locations (Figure 2). It is portable and uses an air tank to supply all the gripping and storage units. The 4 different modules can switch places and the program for the programmable logic controller (PLC) automatically adjusts for the change in location. A linear drive with a pneumatic gripper transports the materials between the different stations. Five real-valued signals are available from the demonstrator: current, position, speed, acceleration and force. Data samples were taken through an OPC connection with a resolution of 50 milliseconds for a total of 42 production cycles. The first 38 production cycles contain only normal behavior and were used to train the self-organizing map for both experiments shown in this section. Two of the 4 remaining cycles contain anomalous behavior and are used for the anomaly detection.

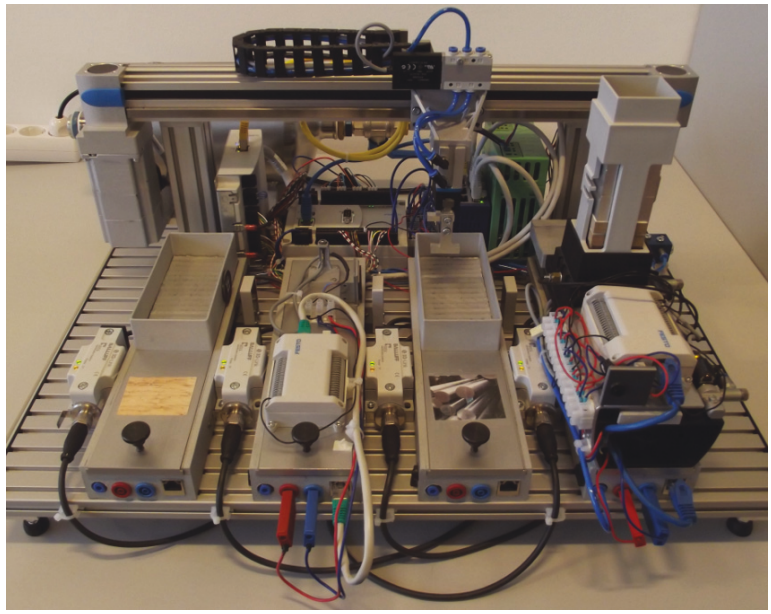

Fig. 2. The Genesis Demonstrator

3.1 Quantization error anomaly detection and anomaly localization

A self-organizing map is trained over 100 Epochs using a 60x60 square, non toroidal topology on the training data using the Eulidean distance measure. Its unified distance matrix representation can be seen in Figure 3. The training data only contains only normal behavior. The anomaly detection is performed in an unsupervised way.

To estimate the threshold for the anomaly detection, tau was set to 1.0 which gives a value of ~0.274 as threshold for the anomaly detection as shown in Figure 4. The observations of the evaluation data are then mapped to the SOM and the distance is calculated as seen in Figure 5. The observation is marked as an anomaly when the distance is larger than the previously estimated threshold.

The final result of the anomaly detection is shown in Figure 6. Anomalies in this data set are labeled which allows to calculate the quality of the anomaly detection: in this example anomalies were detected with an accuracy of 99.63% and F1 score of 94.34%. Sensitivity was 100% which means all anomalies were identified correctly as true positive. Detailed results are shown in Table 1. Figure 7 shows an excerpt of the anomaly localization of the anomalies detected above. Only the two most likely signals estimated to be the cause of the anomaly are shown. The localization results match the predictions made by the experts on the related signals of the system perfectly. The two anomalous cycles shown in Figure 6 contain the same anomalies: the first part of each anomaly, labeled '1' in the data set, is a jam in the linear drive resulting in a standstill and a higher than usual motor current. The second phase of the anomaly, labeled '2' in the data set, is the linear drive

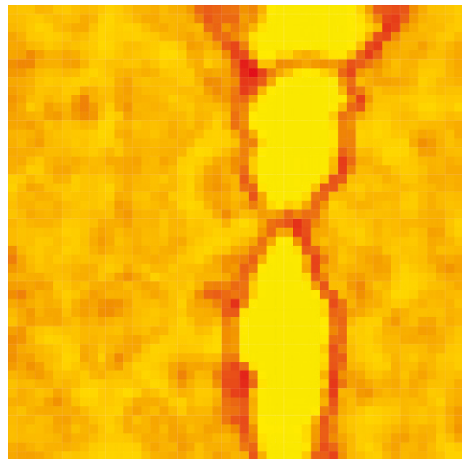

Fig. 3. U-matrix of self-organizing map

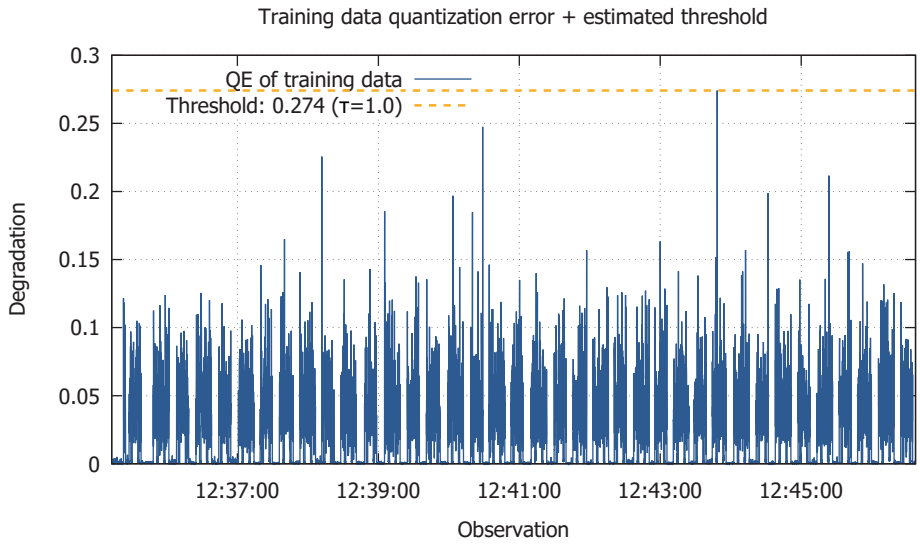

Fig. 4. Quantization error on training data and estimated threshold

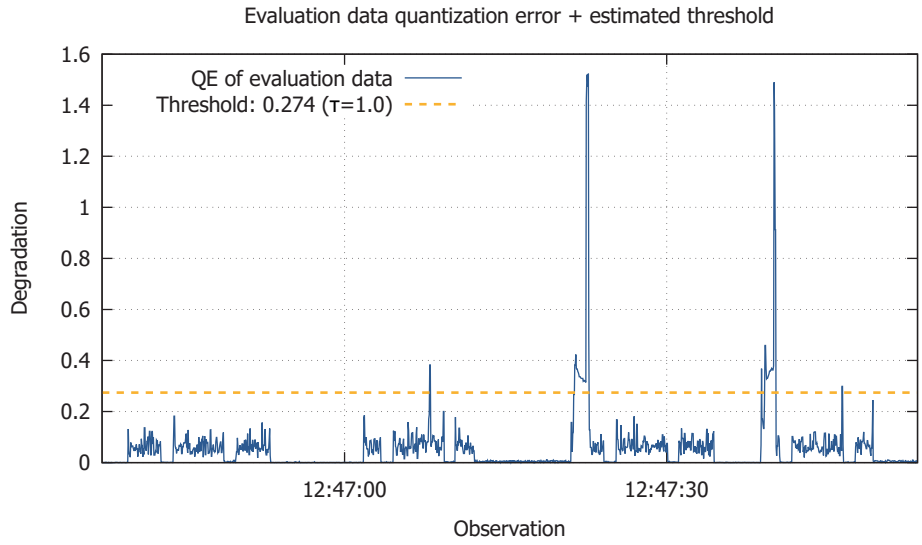

Fig. 5. Quantization error on evaluation data and estimated threshold

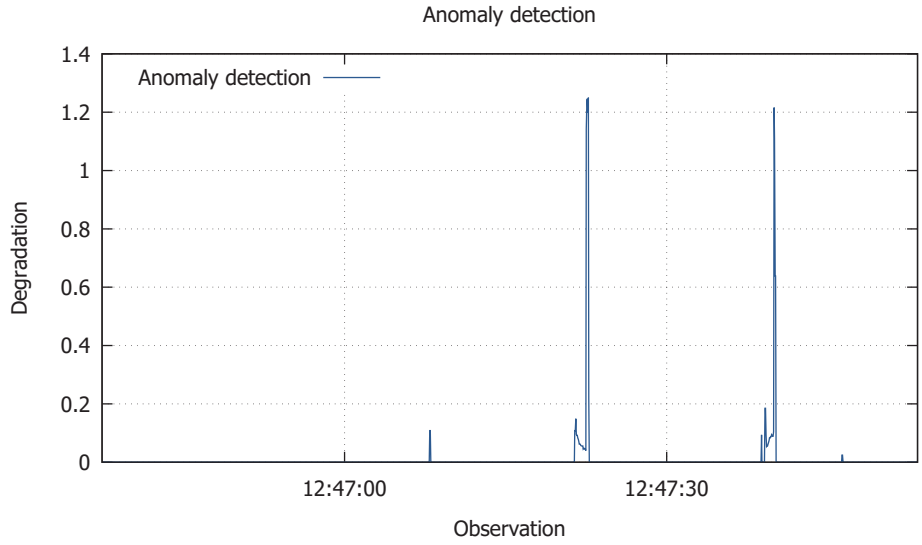

Fig. 6. Final result of the anomaly detection

overcoming the jam, trying to correct its lag error. This results in a much higher than usual speed of the linear drive which is also correctly identified by the anomaly localization.

Table 1. Anomaly detection score

TP	TN	FP	FN	Accuracy	Balanced Accuracy	F1
50	1565	6	0	99.63%	99.81%	94.34%

3.2 Trajectory tracking with automata

Again, a SOM is trained on 38 production cycles containing only normal behavior. The SOM has a size of 60x60 neurons. The resulting u-matrix shown in Figure 8 is then dissected into 6 clusters by the watershed transformation in Figure 9.

Subsequently, the samples of the training data are mapped to the SOM to get the corresponding cluster. The automaton which represents the trajectory across the different clusters during normal operation of the system is learned and shown in Figure 10. The one-hot encoding is easy to read the cluster transitions from the automatons transitions: $C0 = 1; C5 = 0; (5.25 - 10.36)(7.11s)$ describes a transition from cluster 5 to cluster 0 with a timing of 5.25-10.36 seconds after entering the state. The mean time for the transition was 7.11 seconds.

Other encodings than the one-hot can be used so less binary signals are needed for the amount of clusters to describe but they might be harder to read and follow when looking at the automaton.

An example mapping for a single production cycle from the evaluation data is shown in Table 2. Not all states from the state machine can be found in the SOM, as the SOM uses only real-valued signals. Binary signals, such as the storage ejecting material and the gripper closing are not known. The linear drive which provides the real-valued signals does not move during these operations and therefore, these internal states appear in the same cluster of the SOM.

Table 2. Mapping of states to internal state machine and clusters

Automaton state	State machine	State machine description	Cluster
S4	SM0	Idle (standstill)	C0
S5	SM4	Move to storage	C4
S6	SM4	Move to storage (stopping)	C0
S7	SM5	Close gripper (standstill)	C3
S8	SM6	Move to sensor	C2
S9	SM7	Move to target container	C1

```
[...]
20.04.16 12:47:21.438:
MotorData.ActSpeed (0.27)
MotorData.ActCurrent (0.18)
[...]
20.04.16 12:47:22.095:
MotorData.ActCurrent (0.22)
MotorData.ActSpeed (0.21)
[...]
20.04.16 12:47:22.562:
MotorData.ActSpeed (1.50)
MotorData.ActCurrent (0.17)
[...]
20.04.16 12:47:22.703:
MotorData.ActSpeed (1.50)
MotorData.IsForce (0.17)
[...]
[...]
20.04.16 12:47:38.828:
MotorData.IsForce (0.33)
MotorData.ActCurrent (0.14)
[...]
20.04.16 12:47:39.479:
MotorData.ActCurrent (0.21)
MotorData.ActSpeed (0.21)
[...]
20.04.16 12:47:39.998:
MotorData.ActSpeed (1.49)
MotorData.ActCurrent (0.08)
[...]
20.04.16 12:47:40.096:
MotorData.ActSpeed (0.78)
MotorData.IsAcceleration (0.45)
[...]
```

Fig. 7. Excerpt of the anomaly localization algorithm

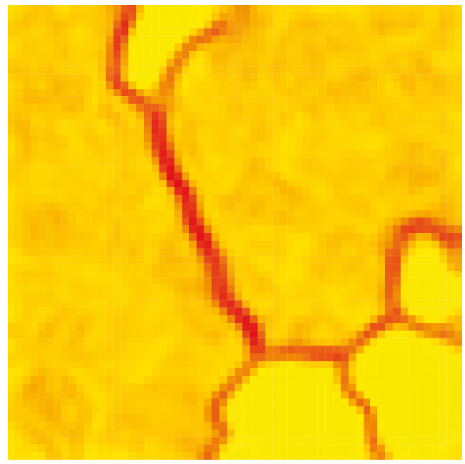

Fig. 8. U-matrix of the SOM

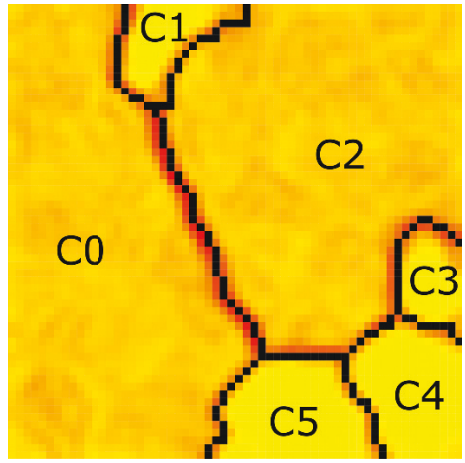

Fig. 9. Clustering after applying watershed transformation

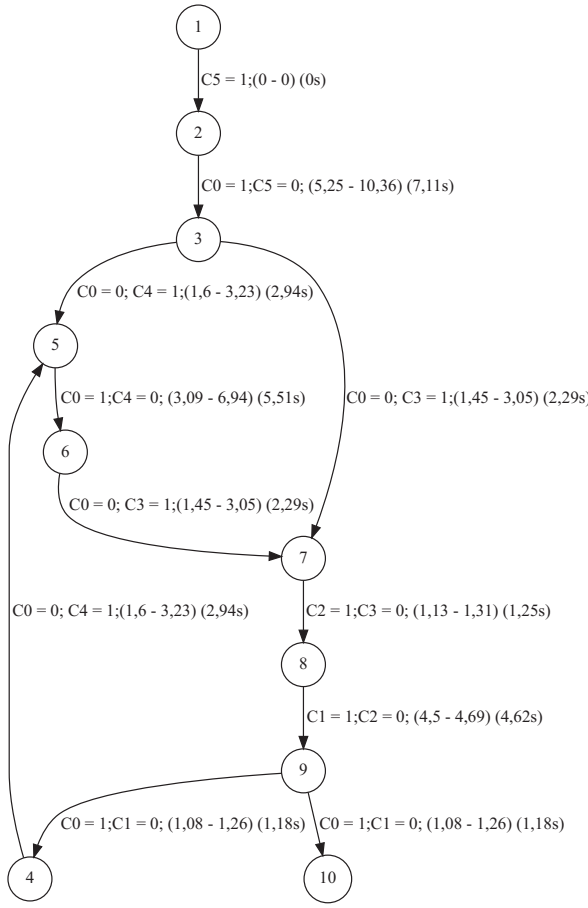

Fig. 10. Automaton tracking the transitions between clusters

Figure 11 shows examples from the output of the ANODA: first, on observation 482 a state remaining error occurs. During the production cycle it took more time to move the gripper to the storage position. This triggers a timing error when the transition finally occurs and normal operation continues. Second, at the end of the data set it was detected that the demonstrator remains in its idle state longer than in the training cycles. Both of these errors are not detectable using the quantization error method presented in the previous section because the SOM itself does not model time in an explicit manner. The automaton adds explicit modeling of time by modeling and tracking the trajectory of the SOM's working point.

```
O481|S5[Normal;]
O482|S5[StateRemainingError;Time is 6.98s but max time in state 5 is 6.94s.]
[...]
O519|S5[StateRemainingError;Time is 8.72s but max time in state 5 is 6.94s.]
O520|S5->S6[TimingError;[8.76s;3.094s to 6.94s with mean 5.50s        ]]
O521|S6[Normal;]
[...]
[...]
O1600|S4[Normal;]
O1601|S4[StateRemainingError;Time is 3.23s but max time in state 4 is 3.22s.]
[...]
O1620|S4[StateRemainingError;Time is 4.12s but max time in state 4 is 3.22s.]
```

Fig. 11. Anomaly detection with the automaton and ANODA

4 Conclusion

This paper presented approaches to data-driven anomaly detection and localization in Cyber-Physical Production Systems. Data provided by the system is used train a self-organizing map to represent the systems normal behavior.

The first option shown in this paper uses the quantization error to detect anomalies in the systems signal domain. Manual adjustment of a threshold above which an anomaly detected is not easy so we estimate the threshold from the data. When an anomaly is found, the SOM is used as a reverse model to compute the differences between expected and actual value of each signal. The signals are sorted from largest to smallest deviation and the first n signals are provided as a starting point for experts to restore the system to a normal working order.

This anomaly detection and localization can be applied to a wide variety of systems and produces good results across the board. However, time is not modeled and deviations in the timing can not be found.

The second option shown in this paper adds modeling of time: discrete timed automata are used to learn the trajectory of the SOM's working point. The automaton keeps track of the timing between the different process phases. This approach

detects anomalies in the systems timing and observed sequence, both of which can not be detected by the SOM alone.

The second approach can be extended to use hybrid timed automata instead of discrete timed automata: a separate model is learned for each state of the automaton to model the different stationary process phases and detect anomalies within them [16]. Yet another approach could replace the discrete timed automata with variable order Markov models. The automaton only uses its current state and long term deviations from the trajectory might not be detectable, especially when the automaton contains cycles. In general, higher order Markov models also use a number of previous states to predict the following state and might be able to better deal with cycles and deviations which happen over a long period of time.

Acknowledgments. This project has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No. 678867.

References

1. Factories of the Future: MultiAnnual Roadmap for the contractual PPP under HORIZON 2020. European Union, Luxembourg (2013)
2. Alhoniemi, E., Hollmn, J., Simula, O., Vesanto, J.: Process monitoring and modeling using the self-organizing map. Integrated Computer Aided Engineering 6, 3–14 (1999), http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.33.573
3. Frey, C.: Monitoring of complex industrial processes based on self-organizing maps and watershed transformations. In: Industrial Technology (ICIT), 2012 IEEE International Conference on (Mar 2012)
4. Henzinger, T.A.: The theory of hybrid automata. In: Proceedings 11th Annual IEEE Symposium on Logic in Computer Science. pp. 278–292 (Jul 1996)
5. Kohonen, T.: The self-organizing map. In: Proceedings of the IEEE. vol. 78 (Sep 1990)
6. Kumpulainen, P., Hätönen, K.: Local anomaly detection for mobile network monitoring. Inf. Sci. 178(20), 3840–3859 (2008)
7. Liukkonen, M., Hiltunen, Y., Laakso, I.: Advanced monitoring and diagnosis of industrial processes. In: 2013 8th EUROSIM Congress on Modelling and Simulation. pp. 112–117 (Sept 2013)
8. Maier, A.: Identification of timed behavior models for diagnosis in production systems. Ph.D. thesis, Paderborn, Univ. (2015)
9. Meyer, F.: Topographic distance and watershed lines. Signal Processing 38(1), 113–125 (jul 1994), http://dx.doi.org/10.1016/0165-1684(94)90060-4
10. Niggemann, O., Lohweg, V.: On the diagnosis of cyber-physical production systems: State-of-the-art and research agenda. In: Twenty-Ninth AAAI Conference on Artificial Intelligence (2015)
11. OPAK-Consortium: Offene engineering-plattform fr autonome, mechatronische automatisierungskomponenten in funktionsorientierter architektur (2017), http://www.opak-projekt.de/
12. Simula, O., Kangas, J.: Process monitoring and visualisation using self-organizing maps (1995)

13. Tian, J., Azarian, M.H., Pecht, M.: Anomaly detection using self-organizing maps-based k-nearest neighbor algorithm. Second European Conference of the Prognostics and Health Management Society 2014 (2014)
14. Ultsch, A., Siemon, H.P.: Kohonen's self-organizing feature maps for exploratory data analysis. In: Proceedings of the International Neural Network Conference (INNC'90 (1990), http://www.uni-marburg.de/fb12/datenbionik/pdf/pubs/1990/UltschSiemon90
15. Vincent, L., Soille, P.: Watersheds in digital spaces: an efficient algorithm based on immersion simulations. IEEE Transactions on Pattern Analysis and Machine Intelligence 13(6), 583–598 (Jun 1991)
16. von Birgelen, A., Niggemann, O.: Using self-organizing maps to learn hybrid timed automata in absence of discrete events. In: 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2017) (Sep 2017)
17. Yin, H.: The Self-Organizing Maps: Background, Theories, Extensions and Applications, pp. 715–762. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.
The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.