Data evaluation in smart sensor networks using inverse methods and artificial intelligence (AI): Towards real-time capability and enhanced flexibility

Stefan Bosse[1,2], Armin Lechleiter[3], Dirk Lehmhus[1]

[1]ISIS Sensorial Materials Scientific Centre, Univ. of Bremen, Bremen, Germany
[2]University of Bremen, Department of Mathematics & Computer Science, Bremen, Germany
[3]University of Bremen, Center for Industrial Mathematics, Bremen, Germany

Keywords. Load Monitoring, Structure Health Monitoring, Machine Learning, Agents, Inverse Numeric

Abstract. Data evaluation is crucial for gaining information from sensor networks. Main challenges include processing speed and adaptivity to system change, both prerequisites for SHM-based weight reduction via relaxed safety factors. Our study looks at soft real time solutions providing feedback within defined but flexible, application-controlled intervals. These can rely on minimizing computation/communication latencies e.g. by parallel computation. Strategies towards this aim can be model-based, including inverse FEM, or model-free, including machine learning, which in practice bases training on a defined system state, too, hence also facing challenges at state changes. We thus introduce hybrid data evaluation combining multi-agent based systems (MAS) with inverse FEM, mainly relying on matrix operations that can be partially distributed: The MAS perform sensor data acquisition, aggregation, pre-computation, and finally application (the LM/SHM itself and higher information processing and visualization layers, i.e., WEB interfaces). System capabilities are evaluated against a virtual test case, demonstrating enhanced stability and reliability. Besides, we analyze system performance under conditions of in-service change and discuss system layouts suited to improve coverage of this issue.

Introduction

Data evaluation techniques are a crucial issue in gaining information from smart sensor networks. Requirements in this respect are subject to change as new application scenarios for and wider spread as well as increased market penetration of such systems creates new challenges. Among them is the need for fast, and ideally real-time, data evaluation, which can be motivated considering Load and Structural Health Monitoring (LM/SHM) as a use case and the associated aim of saving weight through structure- or material-integrated systems: If safety factors are to be relaxed based on the implementation of monitoring systems, there is a need for information to be available within a certain time interval, either with a strict upper time limit (hard real-time), or a within an average time linked to typical service intervals to guarantee operational safety of the system (soft real-time), in order to affect the necessary changes in usage patterns and thus service conditions allowing component and system to survive despite performance losses. Considering distributed sensing systems affected by possible failures themselves (w.r.t. communication and computation), satisfying soft real-time constraints is feasible, while hard real-time is difficult to reach since by definition, missing a hard deadline is equivalent to a complete system failure. Soft real-time systems can be achieved by minimizing computation and communication latencies, e.g., by using parallelization of computation.

Strategies addressing this aim can be model-based or model-free. The former group encompasses e.g. inverse FEM methods, which can be designed to be extremely fast and reliable, but are "hard-wired" to an FEM model of the monitored system. For that reason, such methods cannot be easily adapted to system change. To address the latter, machine learning techniques are frequently employed. However, these, too, have to base their training on a defined and thus fixed system state.

In contrast, safety-relevant systems need to maintain their predictive capabilities even after damage has been incurred, causing the monitored component's state to change **[7]**.

Realizing such capabilities is closely linked to weight reduction potentials of SHM systems. Achieving them would thus create a major drive towards broader introduction of SHM.

The present study addresses this issue by introducing a hybrid data evaluation approach which combines AI methods, and here specifically Multi-Agent based Systems (MAS) and Machine Learning (ML), with inverse FEM calculations, mainly relying on matrix operations suitable for partial distribution. In our concept a MAS perform the sensor acquisition, aggregation, pre-computation, learning, and finally application (the LM/SHM itself and higher information processing and visualization layers, i.e., WEB interfaces). ML is a suitable approach for the evaluation of sensor data **[4]**.

In an extension of previous investigations of systems having the aforementioned general structure, system capabilities are now investigated based on a virtual test case, which represents a real-world load bearing structure. As stated above, mobile autonomous distributed MAS using self-organizing structures can only satisfy soft real-time constraints, but can be considered as stable in the presence of environmental and technical failures of the processing system and network, introducing enhanced reliability of the entire LM/SHM solution, which can be more critical than satisfying time fixed time constraints.

Besides, the present study analyses the system performance under conditions of in-service change. Questions in this respect include the issue of missing sensor data caused by failure of individual sensors, material property change induced by variations in service conditions, influence of statistical scatter of material properties or component geometry and structural damage caused, e. g., by short time events like impacts temporarily exceeding design loads and thus introducing local damage.

Starting from this general evaluation of the ability to maintain predictive capabilities, system layouts are discussed which incorporate flexibility and adaptability to state changes in their initial/ fundamental design by employing advanced model free data evaluation, or by integrating model-updating features. With respect to the latter aspect, the value of combined active/passive sensor networks is discussed: The nodes of these do not only listen to signals, but can also act as emitters, thus allowing the system to actually test its own state, basing the judgement on reference signals clearly defined in level, time, and space. In specifically addressing state change, the paper goes beyond many other works which consider MAS in an SHM context mainly in terms of damage detection accuracy and speed **[8][9]**.

Distributed Computing in heterogeneous Networks using Mobile Agents

One major goal of the deployment of MAS is overcoming heterogeneous platform and network barriers arising in large scale hierarchical and nested network structures, consisting and connecting, e.g., the Internet, sensor networks, body area networks and/or Cyber-Physical System (CPS) networks in production engineering. Mobile agents are commonly represented by mobile program code that can be transferred between different host platforms and network nodes. An agent is associated with a computational process, and its migration commonly requires the creation of a snapshot of its process state, which is transferred to another node, where its execution is continued.

The deployment of agents can overcome interface barriers and closes the gap arising between platforms and environments differing considerably in computational and communication capabilities, enabling, e.g., the integration of sensor networks in large-scale World-Wide-Web (WWW) applications and providing Internet connectivity, shown in Fig. **1**. This issue is addressed in this work using a unified reactive agent-based programming and interaction model, which is independent of the underlying processing platform, in combination with a highly portable agent processing platform based on common *JavaScript*, the *JavaScript* Agent Machine (*JAM*), shown in the middle of Fig. **1**.

Agent mobility crossing different execution platforms and agent interaction via tuple-space databases and global signal propagation aid solving data distribution and synchronization issues in the design of distributed sensor networks.

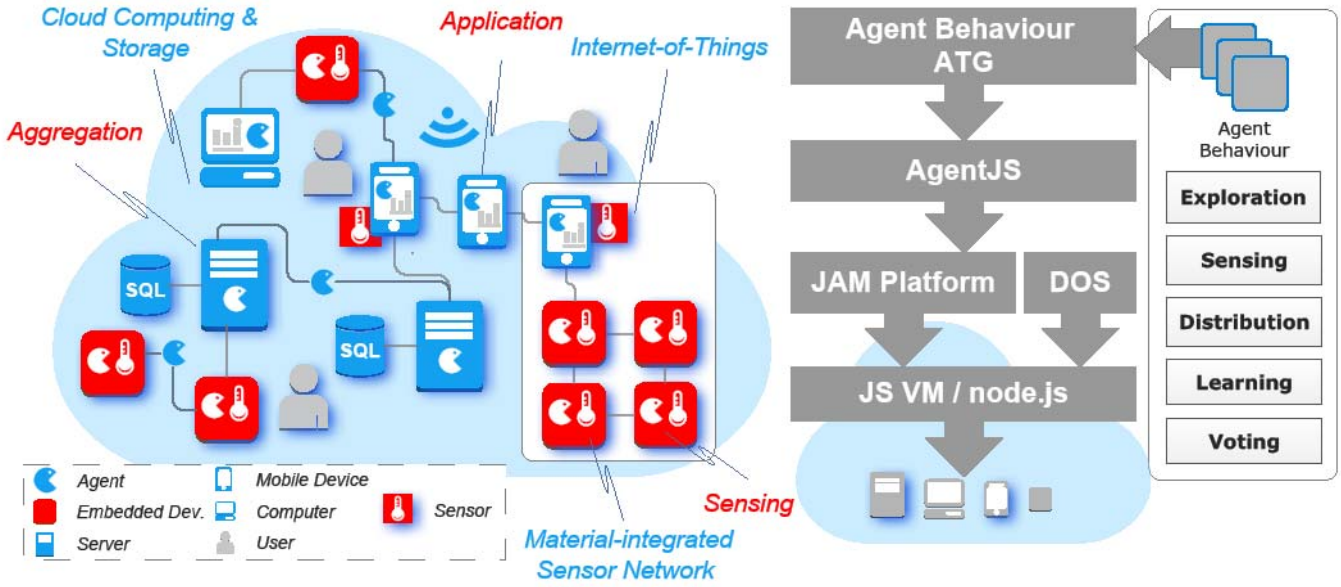

Fig. 1. Deployment of agents in Sensor Clouds and Internet applications (left) using a highly portable Agent Processing Platform (middle) based on *JavaScript* implementing different agent behaviour (right).

A sensor network as part of the Internet-of-Things (IoT) is composed of nodes capable of sensor processing and communication. Smart systems are in turn composed of more complex networks (and networks of networks) differing significantly in computational power and available resources, raising inter-communication barriers.

Usually sensor processing and information computation require defined world models including mechanical models, for example, in load monitoring use cases of technical structures. Self-organizing MAS are useful in unreliable and partially unknown environments, since they can overcome environmental and physical model-related uncertainties successfully.

Adaptation of the agent behaviour, i.e., based on learning, offers a reliable reaction mechanism in the presence of environmental variations, like changes in network connectivity or node failures. This adaptivity is addressed in this work by means of a graph-based behavioural reconfiguration at runtime. Mobility - the ability to migrate a agent processing unit to a different execution platform and node - and autonomy together with a high degree of independence of the processing platform ensure robust data processing in large-scale networks.

It can be shown that agent-based computing can be used to partition complex computations in off-line and on-line (in network and real-time) parts resulting in an increased overall system efficiency (both in terms of performance and energy demands) and a unified programming interface between off- and on-line parts, as outlined for LM/SHM-systems in **[2]**.

MAS solutions provide higher level information processing that maps the raw sensor data to condensed information. They can facilitate, e.g., Internet connectivity of perceptive systems (body area networks...). These smart systems unite sensing, aggregation, and application layers **[6]**, offering a more unified design approach and more generic and unified architectures. Smart systems glue software and hardware components to an extended operational unit, the basic cell of the IoT.

The central approach in this work focuses on mobile agents and the ability to support mobile reconfigurable code embedding the agent behaviour, the agent data, the agent configuration, and the current agent control state, finally encapsulated in portable *JavaScript* code. The mobility is granted by converting the agent program in an extended textual *JSON+* representation, and finally by parsing this text and executing the code again. This agent-specific mobile program code can be executed on a variety of different host platforms including mobile devices, embedded devices, sensor nodes, and servers, using *JAM* and a *JS* VM, closing the gap between the IoT and Cloud infrastructures. Additionally, the agent processing platform provide machine learning as a service.

Among the Internet-of-Things and Ubiquitous Sensorial Perception, one major field of application for the agent-based information processing is LM/SHM of mechanical structures.

A portable and secure Agent Processing Platform

Microchip-capable agent processing platforms were already proposed in **[3]**, but were limited to application-specific designs, well suited for material-integrated sensor networks. To simplify the development and deployment of Self-organizing MAS in the Internet domain and sensor networks it would be desirable to implement agents directly in JavaScript (*JS*). The agent behaviour is modelled with an Activity-Transition Graph (ATG) and implemented entirely in *JavaScript* with a restricted and encapsulated access to the platform API. *JS* Virtual Machines are available on all host platforms including WEB browsers and mobile devices. The proposed novel *JS* Agent Machine (*JAM*) is capable of executing *AgentJS* agents directly in a sandbox environment with full run-time protection, low-resource requirements, and Machine Learning as a service. Hence, agents can carry learned models separating code (ML algorithms) and data. Incremental learning can be used to adapt the learned model during run-time **[5]**, covering system change. Agents can migrate between different *JAM* nodes seamless preserving their data and control state by using on-the-fly code-to-text-to-code transformation. *JAM* thus enables a seamless integration of different host platforms (server, desktop computer, mobile devices, embedded devices, material-integrated sensing systems) with one unified agent model and portable platform architecture.

JAM consists of different modules entirely implemented in *JS* that can be executed by any stand-alone *JS* VM or within WEB browsers, shown on the right hand side of Fig. **1**. The deployment in Internet and client-side applications like browsers and the Internet require a Distributed Co-ordination and Operation System layer (DOS) with a broker service, which will not be discussed here (details can be found in **[1]**).

The Agent Input/Output System (*AIOS*) is the main execution layer of *JAM*. It consists of the sandbox execution environment encapsulating an agent process, with different privileged sub-sets depending on an agent role (level 0,1,2). The *AIOS* module implements the agent process scheduler and provides the API for the logical (virtual) world and node composition. Finally, it provides the *AgentJS* API for agents (agent control, tuple-space access, mobility, and many more utility functions). The sandbox environment grants restricted access to a code dictionary based on the privilege level, enabling code exchange between agents Level 0 agents are not privileged to replicate, create, or kill other agents and to modify their code.

Machine Learning compared with Inverse Numerical Approaches

In this work, different load situations were applied to a metal plate, and the strain of the plate was computed at particular points using FEM simulation, finally mapped on artificial sensor data processed by a MAS in the *SEJAM* MAS simulator (*JAM* node extended with a visualization and control layer) combined with Monte-Carlo simulation techniques creating noisy sensor data sets.

For engineering applications in the context of SHM there exist accurate physical models describing the behaviour of a mechanical structure under space- and time-dependent load. Such models can hence be used to compute the structure's change under surface loading, but of course also to obtain information on surface loads from measurements of structural changes. In the context of this paper, structural change is constituted through strain measurements by sensors forming an embedded sensor network. We denote the mapping that takes surface loads $\boldsymbol{l}$ and evaluates the corresponding strain measurements $\boldsymbol{s}$ by $\boldsymbol{T}$. Assuming a linear model for simplicity, a finite-dimensional load spaces that correspond to some load discretization of dimension N, and M sensor measurements of surface strain, $\boldsymbol{T}$ is a matrix of size M times N.

Typical ways of discretizing surface loads are load representations via finitely many piecewise constant basis functions that cover parts of the surface of the mechanical structure under investiga-

tion. Using such finite representation together with a physical model for the structure load makes it possible to simulate the matrix $\boldsymbol{T}$ numerically via, e.g., finite element methods. Given a measured, noisy strain data set $\boldsymbol{s}^{\delta}$ with noise level $\delta>0$, inverse numerical (IN) methods seek to stably approximate a solution to the linear system $\boldsymbol{Tl}=\boldsymbol{s}^{\delta}$. Typically, this is a challenging problem due to the ill-conditioning of the matrix $\boldsymbol{T}$: For most indirect measurement systems, the amount of information that can be recovered form sensor measurements is intrinsically limited by the measurement set-up and if few sensor measurements are available, the inversion problem is moreover under determined. The solution $\boldsymbol{l}$ to the linear system $\boldsymbol{Tl}=\boldsymbol{s}^{\delta}$ hence needs to be stabilized, which is discussed in detail in **[2]**. The simplest variant is to apply Tikhonov-Phillips regularization. A variant to stabilize the linear system is to apply an iterative regularization method as, e.g., the conjugate gradient iteration (cg iteration), that must then be stopped early for stabilization. The advantage of the cg-iteration with early termination is that the costly solution of the linear system can be avoided, which makes the method potentially fast if few linear systems are to be solved. Moreover, compared to the Tikhonov iteration, the method yields far less over-smoothed reconstructions, which is advantageous if one aims to reconstruct loads concentrated on small domains. On the other hand, this smoothing effect of Tikhonov regularization makes Tikhonov regularization stabler for high noise level.

Originally, the simulated loads are cylindrical weights placed at $\boldsymbol{N}$=400 weight positions forming an equidistant rectangular grid of 400 points. The force on the upper surface of the upper horizontal plate due to the loading hence vanishes outside the circle covered by the weight; inside this circle the force points in direction $-\boldsymbol{z}$ and equals 1 N/cm^2. Since the deformation model used is linear, the actual value assigned to this force is irrelevant if the load-strain matrix is re-scaled such that the magnitude of reconstructed loads matches those of the true load for noise-free data. After computing the deformation field, the surface strain in x and y direction at the sensor points is extracted by computing the deformation field and the extracted surface strain (see Fig. **2**).

There are still many sensing applications operating stream-based, i.e., the sensor information is collected by one or multiple dedicated nodes periodically from all sensor nodes, requiring high-bandwidth communication and consuming a significant amount of power. Frequently, most of the sampled sensor data and information do not contribute to new information about the sensing system, since in a multi-sensor system only a few sensors will change their data beyond a noise margin. In a previous work **[2]** it was shown that different data processing and distribution behaviour can be used and implemented with agents, leading to a significant decrease in network communication activity and a significant increase of the reliability and Quality-of-Service.

The central approach for global self-organization is an event-based sensor processing and distribution behaviour; Extended with an adaptive path finding (self-routing) supporting agent migration in unreliable networks with partially missing connectivity or nodes by using a hybrid approach of random and attractive walk behaviour; And local self-organizing agent systems with divide-and-conquer exploration, distribution, replication, and interval voting behaviour based on feature marking (details in **[2]**).

In this work, decentralized and self-organized learning performed by mobile agents was added and used to predict specific load cases even in the presence of noisy sensor data. This approach do not require a mechanical model of the structure under test. This approach is well suited for structural monitoring with material-integrated sensor networks, typically consisting of low-resource computing devices. Distributed learning divides a spatially distributed data set in local regions and applies learning to limited local regions, based on a divide and conquer approach. Each local learner agent is trained with sensor data from a Region of Interest (ROI), finally learning a decision tree model using robust data variable separation based on an ε-interval entropy and partitioning approach. Each variable represent a sensor in the ROI. Decision trees are simple and compact models derived from learning with training data, and well suited for agent-based learning. Prediction bases on the learned interval tree model and nearest neighbourhood variable selection.

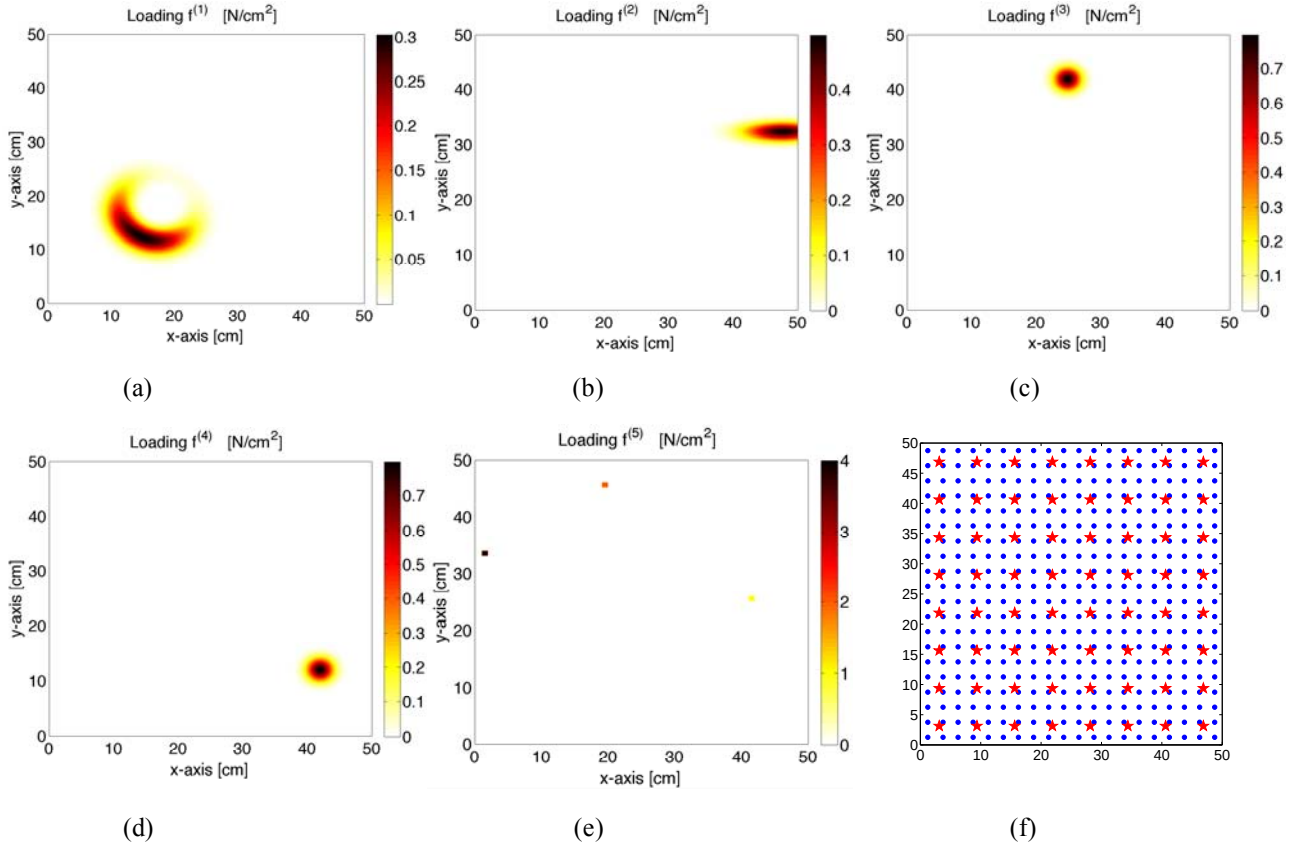

Fig. 2. From (a) to (e): The five loads $\boldsymbol{l}^{(1)}, .., \boldsymbol{l}^{(5)}$. (f) The positions of the 400 weight points $\boldsymbol{w}\{\boldsymbol{i},\boldsymbol{j}\}$ are indicated using blue dots; red crosses indicate the positions of the 64 sensors **[2]**.

Among the distribution of the entire learning problem, event-based activation of learning entities can improve the system efficiency significantly. Commonly the locally sampled sensor values in a ROI are used for an event prediction, waking up the learner agent, which collect neighbourhood data by using a divide-and-conquer system with explorer child agents, and finally sent out voter agents carrying a vote for predicted classification (load case). All votes are collected by election agents performing a majority election.

Table **1** shows the evaluation of load case prediction results achieved with the ML and IN approaches. The mean prediction probability for the correct classification using the ML approach was computed from the vote distribution of multiple experiments using Monte-Carlo simulation techniques (creating noisy sensor data). The mean correlation probability using the IN approach was computed from the peak value of the normalized 2d-cross correlation function of the original (synthetically generated) and the reconstructed load matrix (with noisy sensor data), repeated with multiple experiments using Monte-Carlo simulation techniques, too. The achieved probabilities and cross correlation coefficients show that the ML approach can effectively be used to pre-classify a reconstructed load matrix, even in the case of low correlation between the reconstructed and the original load.

Method/Load Case	**l0**	**l1**	**l2**	**l3**	**l4**	**l5**
Mean ML Prediction Prob.[%]	94	82	82	95	47	58
Mean IN Correlation Coefficient [%]	-	81	61	38	60	32

Table 1. Comparison of load case prediction and reconstruction accuracy achieved with (a) Distributed Machine Learning and (b) Inverse Numeric (l^0: zero load, only recognized by ML)

Conclusion and Outlook

In the present wok, sensor nodes have been deployed as sensing and computational entities providing an agent processing platform suitable to implement distributed sensor processing. The agent processing platform can be deployed in strong heterogeneous networks including the Internet and a large variety of host platforms, ranging from embedded systems to servers. Mobile agents are represented by mobile *JavaScript* code. This agent platform provides Machine Learning as a service, and agents can carry learned models. An decision-tree ML approach was used to predict different load cases of a mechanical structure. The results were compared with an IN approach, showing that the ML approach can be used to pre-classify a reconstructed load matrix, even in the case of low correlation between the reconstructed and the original load. Robustness can be demonstrated for both approaches (IN & ML) concerning sensor noise immunity and drift. Adaptivity on system level is covered by SoS/MAS approaches, adaptivity on structural state level can partly be covered by incremental learning during run-time.

Future work will evaluate the latter issue in more detail. Furthermore, more complex non-linear material models will be considered in IN approaches. With respect to the detection of structural change (e.g., irreversible damage), a probabilistic learner can be deployed, delivering a trust probability over time and following mechanical state change. ML and IN can be merged in a hybrid approach in such a way that (1) ML delivers trust probability and a pre-classification for IN, (2) IN triggers incremental learning of new/unknown load cases.

References

[1] S. Bosse, Unified Distributed Computing and Co-ordination in Pervasive/Ubiquitous Networks with Mobile Multi-Agent Systems using a Modular and Portable Agent Code Processing Platform. In The 6th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN 2015), Procedia Computer Science, 2015.

[2] S. Bosse, A. Lechleiter, A hybrid approach for Structural Monitoring with self-organizing multi-agent systems and inverse numerical methods in material-embedded sensor networks. Mechatronics, 2015, doi:10.1016/j.mechatronics.2015.08.005, in press

[3] S. Bosse, Design and Simulation of a Low- Resource Processing Platform for Mobile Multi-Agent Systems in Distributed Heterogeneous Networks. In Agents and Artificial Intelligence, LNAI 8946, Springer, Béatrice Duval, J. van den Herik, S. Loiseau, and J. Filipe, Eds. Springer, 2015.

[4] A. Moraru, M. Pesko, M. Porcius, D. Mladenic, and C. Fortuna1, Using Machine Learning on Sensor Data. Journal of Computing and Information Technology, vol. 4, pp. 341–347, 2010.

[5] J. Gama and P. Kosina, Learning Decision Rules from Data Streams. In Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, 2011.

[6] V. Di Lecce, M. Calabrese, and C. Martines, From Sensors to Applications: A Proposal to Fill the Gap. Sensors & Transducers, vol. 18, no. Special Isse, pp. 5–13, 2013.

[7] D. Lehmhus, J. Brugger, P. Muralt, S. Pane, O. Ergenemann, M.-A. Dubois, N. Gupta, M. Busse, When nothing is constant but change: Adaptive and sensorial materials and their impact on product design. Journal of Intelligent Material Systems and Structures, vol. 24, doi: 10.1177/1045389X13502855, 2013.

[8] K. Smarsly, K. H. Law, D. Hartmann, Implementation of a multiagent-based paradigm for decentralized real-time structural health monitoring. Proceedings of the Structures Congress 2011, Las Vegas, Nevada, USA, April 14th-16th, 2011, p. 1875-1885, doi: 10.1061/41171(401)163

[9] D. Liang, S. Yuan, Structural health monitoring system based on multi-agent coordination and fusion for large structure. Advances in Engineering Software, vol. 86, pp. 1-12, 2015.