Chapter 14

Use-Cases Environmental Perception, Load Monitoring, and Manufacturing

Load Monitoring with Multi-Agent Systems, Machine Learning, and Inverse Numeric for Structural Monitoring, Environmental Perception, and Manufacturing

This chapter presents several main use-cases for the deployment of MAS and Artificial Intelligence methods by providing perception either of the physical load of the environment acting on a technical structure, i.e., a robot manipulator and its interaction with obstacles and objects, or by providing the internal load of a structure, i.e., the deformation of structure caused by loads. The latter case provides input for Load and Structural Health Monitoring systems (LM/SHM), e.g., used for the monitoring of aircraft wings. A distributed deployment of industrial agents in large-scale and heterogeneous environments is presented with an adaptive and additive manufacturing use case.

The field of structural monitoring has evolved in the past 30 years, and can be classified by the abstraction level of the information derived from the sensor input, shown in Figure *14.1* [LEH13].

Agents are already deployed successfully for scheduling tasks in production and manufacturing processes [CAR00B], and newer trends poses the suitability of distributed agent-based systems for the control of manufacturing processes [PEC08], facing not only manufacturing, but maintenance, evolvable assembly systems, quality control, and energy management aspects, finally introducing the paradigm of industrial agents meeting the requirements of modern industrial applications by integrating sensor networks. Multi-agent systems can be successfully deployed in sensing applications, for example, structural load and health monitoring, with a partitioning in off- and online computations [BOS14C]. Distributed data mining and Map-Reduce algorithms are well suited for self-organizing MAS. Cloud-based computing with MAS, as a base for cloud-based manufacturing, means the virtualization of resources, i.e., storage, processing platforms, sensing data or generic information.

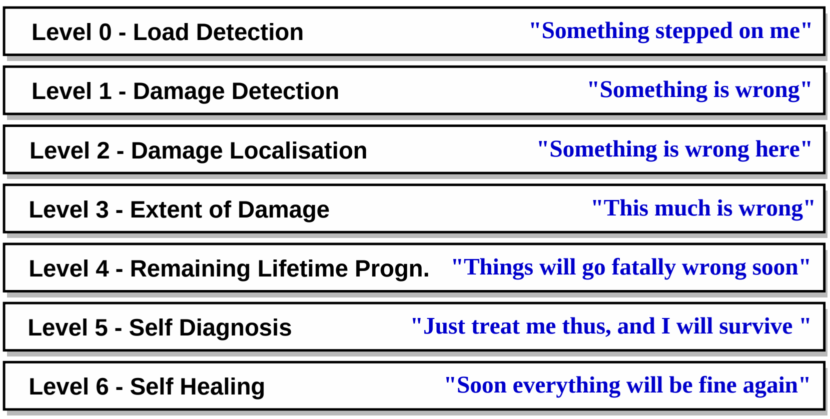


Fig. 14.1 *Algorithmic and information level hierarchy in SHM and LM systems* [LEH13]

epubli, ISBN 9783746752228 (2018)

Load Monitoring

A Load Monitoring system (LM) can be considered as being the lowest part (Levels 0-3) of a full SHM system, which provides spatial resolved information about loads (forces, moments, etc.) applied to a technical structure. When implemented in robot grippers [BOS12D], such systems can improve grasping performance via feedback control. Another application can be found in long distance robot manipulator structures to improve position accuracy [MAV97]. In aerospace industry, load monitoring systems as first step towards structural health monitoring were initially implemented in military aircraft like the Eurofigther TYPHOON [HUN01], but have meanwhile entered the civilian market in parallel to their ongoing maturing into true SHM systems [RUL13][SAN13].

Structural Health Monitoring

Structural Health Monitoring adds the ability to derive not just loads, but also their effects on the structure from sensor data (Levels 3-5). Boller [BOL09] gave a definition of a SHM system: *A SHM is the integration of sensing and possibly also actuation devices to allow the loading and damaging conditions of a structure to be recorded, analysed, localized, and predicted in a way that non-destructive testing (NDT) becomes an integral part of the structure and a material*.

Structural control, adaptive and morphing structures

The term structural control implies a material-inherent capability of changing structural characteristics in response to, and countering the effects of external loads (Level 6). It is thus beyond mere sensing and thus beyond the scope of systems addressed within this text. These however form the necessary basis of farther-reaching adaptive systems.

Besides alleviating structural loads, structural control-like systems are foreseen to facilitate a fly-by-feel approach for autonomous flight of unmanned aerial vehicles (UAVs) [SAN13]. At a second glance, this scenario can be seen as linked to robotic tactile sensing: In both cases, a major extension of perceptive capabilities at the interface between system and environment is foreseen to support interaction with the latter, and based on it, allow for increased autonomy.

Tactile Sensing

Tactile sensing (TS) systems provide extrinsic perception for robots and robotic applications [CAN10], via systems commonly designated as smart or artificial skin. Basically they deliver spatial resolved information on forces applied to an extended but limited surface region, for example, of robot connection elements or finger tips of a robot hand [DAH07] [VID11]. The finger tip example is probably what comes to mind first when thinking of robotic tactile sensing. However, covering areas other than a fingertip with a smart skin capable of interpreting tactile sensor data bears additional advantages: Sensor networks of this kind can provide the robotic system with much finer information about – voluntary as well as involuntary – contacts with its surroundings than state-of-the-art joint-integrated load monitoring ever will. This can be exploited to enhance safety levels in robot-robot or human-robot cooperation, but just as well to endow the robot with the capability to monitor its own actions, profiting from their success or failure via reinforcement learning. The concept has been applied to humanoid robots in the form of tactile sensor arrays to help monitor, control and ultimately improve their strategy to dynamically stand up from lying on their back. Such examples are merely a glimpse at the full potential of material-integrated sensor systems. From the above, many development trends can easily be extrapolated, like the simple change of perspective that turns robotic smart skins into new kinds of tactile user interface for countless purposes.

Industrial 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 networks, production and manufacturing Cyber-Physical System (CPS) networks. The large diversity of execution platforms, network topologies, services provided by network nodes, and the programming environments require a unified and abstract behavioural and structural representation model that can be delivered by industrial agents [PEC08].

14.1 Sensorial Material I: A Flat Perceptive Sheet and Machine Learning

Intelligent behaviour of robot manipulators become important in unknown and changing environments. Emergent behaviour of a machine arises intelligence from the interactions of robots with its environment. Sensorial materials equipped with networks of embedded miniaturized smart sensors can support this behaviour.

Environmental perception can be provided by some elastic material covering extended surface of robotic structures, e.g., intersection elements or body covers. An integrated autonomous decentralized sensor networks can be capable providing perception similar to an electronic skin. Each sensor network is connected to strain gauge sensors mounted on a flexible polymer surface, delivering spatial resolved information of external forces applied to the robot arm, required for example for obstacle avoidance or for manipulation of objects.

The first attempt of a perceptive material was a flat rubber sheet (based on work in [BOS11C]), which was finally bent around a technical structure, presented in Section *14.2*.

Each autonomous sensor node provides communication, data processing, and energy management implemented on microchip level.

Commonly a high number of strain gauge sensors are used to satisfy a high spatial resolution. The chosen approach uses advanced Artificial Intelligence and Machine Learning methods for the mapping of only a few non-calibrated and non-long-term stable noisy strain sensor signals to spatially resolved load information and a decentralized data processing approach to improve robustness. Robustness in the sensor network is provided by 1. Autonomy of sensor nodes; 2. By smart adaptive communication to overcome link failures and to reflect changes in network topology; and 3. By using intelligent adaptive algorithms. It is well-known that robust cooperation and distributed data processing is achieved by using Mobile Agent systems [WAN03]. As already outlined in this book, the agent behaviour and cooperation is implemented on microchip level.

The central aim is to derive useful information constrained by limited computational power and noisy sensor signals unable to be captured by a complete system model. Machine Learning (ML) methods are capable to map an initially unknown n-dimensional set of input signals to a m-dimensional output set of information like the position and strength of applied forces [MIT97].

Without any interaction and material model Machine Learning requires a training phase. Additional material models and FEM simulation can reduce or avoid the training phase [BOS11C].

The training set contains recorded load positions, masses and classification results for different load cases determined via sensor measurement.

The hyper-elastic behaviour of polymers reduces the long-term prediction accuracy of learned models as well as the consistency with FEM output, requiring Machine Learning models that automatically adjust their output to the structure's aging process.

14.1.1 Machine Learning and Multi-Agent Systems

Perception of a robot requires some kind of sensitive skin. The proposed skin consists of a smart strain-gauge sensor network. Machine learning methods with prior training are used to map (classify) a set of (preprocessed and filtered) noisy sensor signals to spatially resolved load information applied to the skin. This approach allows usage of lower sensor density and non-calibrated sensors with unknown electromechanical signal model without loss of spatial resolution. Figure *14.2* shows different machine learning approaches (classification and regression) used for mapping of sensor signals to load information with k-nearest-neighbourhood, decision tree, and neuronal network algorithms enabling agent-based structure monitoring based on trained (condensed) data [MIT97]. K-nearest-neighbourhood algorithms are used for numerical regression of load position, load strength, and displacement vectors, C4.5 decision trees can be used for strength classification, and neuronal networks are suitable for numerical regression of load position and strength.

The training set is either derived by using FEM simulations or by using real test runs and sensor measurement of the structure under test (SUT). This training set contains all correct load positions, masses and classification results for every load case in the load case library along with a respective 8-dimensional strain vector. To answer a query with k-Nearest-Neighbour (k-NN), the k entries closest to the query strain vector are selected and combined into a single estimation for the target variables. Numerical regression of the load position and mass can be achieved by calculating the weighted average, whereas discrete (e.g., Boolean) classification is possible by means of weighted voting. For a fixed k, let $\{E_1, .. , E_k\}$ denote a set of the k nearest points in the training set, (x_i, y_i, m_i) the load coordinates and mass of E_i, and $Q \in \mathbb{R}^8$ the query vector. The estimated load position and mass $(x(Q), y(Q), m(Q)) = (x_Q, y_Q, m_Q)$ for Q is calculated using inverse distance weighting, given by Equation *14.1*.

$$\begin{pmatrix} x_Q \\ y_Q \\ z_Q \end{pmatrix} = \sum_{i=1}^{k} w(E_i) \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix},$$
$$\text{with } w(E_i) = 1/(\text{dist}^2(Q,E_i) w_{sum}),$$
$$\text{and } w_{sum} = \textstyle\sum_{i=1}^{k} 1/\text{dist}^2(Q,E_i) \tag{14.1}$$

The $\text{dist}^2(Q,E_i)$ function computes the squared Euclidean distance between points Q and E_i in the 8-dimensional strain space. Weighted voting is done by adding the weighting factors $w(E_i)$ of all E_i that have the same classification and then assigning to Q the classification, which gained the largest sum. In the first experiments presented in the next section, the C4.5 decision tree learning is employed for classification only as its reliable extension to multivariate numerical regression is more elaborate.

Now Multi-Agent Systems can be deployed for multiple purposes in such a learning system:

1. Local acquisition and preprocessing of sensor data
2. Global Event-based distribution of sensor data and delivery of sensor data to dedicated computational nodes, based on SoS algorithms presented in Chapter 9.
3. Distributed computation of the load vector from sensor data by using distributed ML algorithms, as shown in Figure 14.2.

Applied to the task of monitoring a technical structure equipped with distributed sensors, the agent may be physically situated on locations on the latter, with its perceptions coming from different groups of sensor nodes and communication with other agents. The group defines a region of interest (ROI), which can be formed by sensorial events and correlation of sensor signals happening in this region. If the agent has access to a knowledge base of facts about the structure's geometric and mechanical characteristics as well as cause-effect relations with respect to load introduction, it can conclude about the current state of its environment based on its sensor input, i.e., make sense of the sensor signals. It is the presence of this semantic level that distinguishes intelligent agents from nodes of the sensor network.

The sensor nodes bound to the sensor networks only provide the infrastructure for routing and distribution of measured sensor data across the physical object. It provides initially no way of interpreting this data, but agents are capable of incrementally constructing a mental model of their surroundings over time.

The structure can be partitioned into connected substructures, which can be further partitioned into regions. Each region can be managed by an individual monitor agent that is assigned and that regularly adjusts its internal model of the respective region according to its perceptions. Communication among agents covering directly neighbouring regions enables the construction of a (simplified) distributed global view of the entire structure state. For this purpose, each monitor agent requires an appropriate description of how loads and their effects are transmitted across the boundaries between the agent's region and each of the adjacent regions. That way, an agent can infer from its local sensor input the structural state at these boundaries and send the results to its topological neighbour agents, which in turn update their local models by combining their own sensor data with the boundary state information they received from their respective neighbours, and, again, pass the updated boundary states on to these, and so forth.

A distributed load inference algorithm can be establishing by using a multi-level hierarchy of agents and incorporating equation-based knowledge from the FEM domain into the reasoning process.

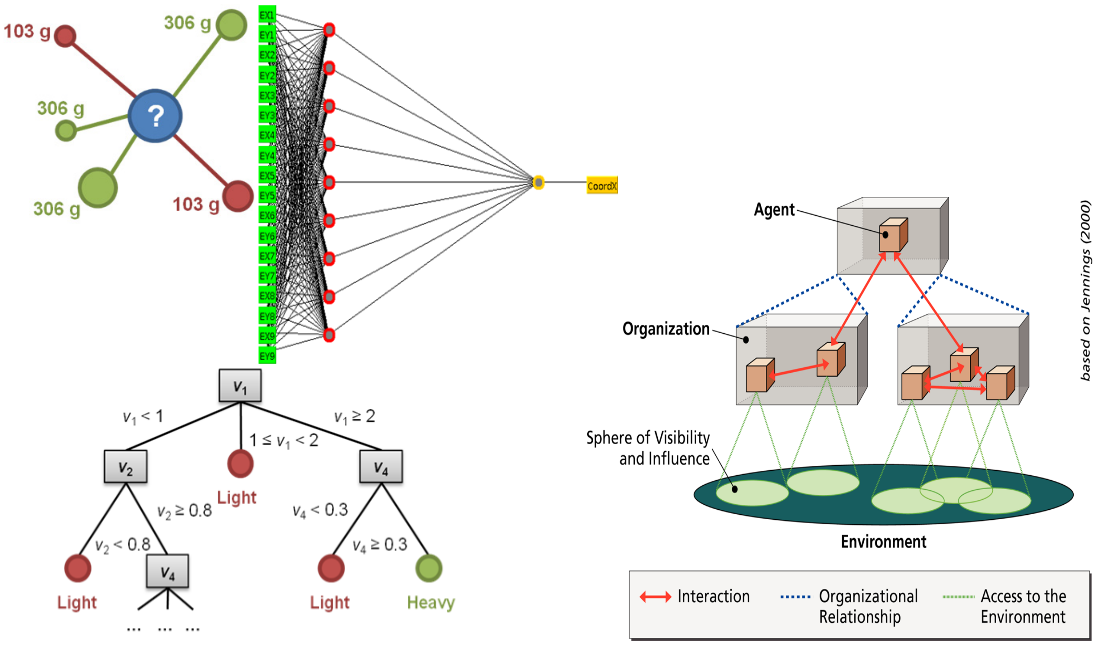


Fig. 14.2 Different machine learning methods (k-nn, c4.5 desc. trees, neuronal networks) used to retrieve load information (position and strength) and agent-based structure monitoring based on trained data.

This inference algorithm offers adaptive behaviour in terms of desired temporal resolution and predictive precision, depending on the number of communication rounds performed per time unit (information exchange between agents).

Experiments investigating the suitability of the previously introduced ML algorithms for the derivation of the spatially resolved and interpolated load vectors from the sensor data vectors were split in FEM-based training with a metal plate model and real sensor measurements using a rubber plate.

14.1.2 Experimental Results using Machine Based Learning with Metal Plate

Completely noise-free strain values from a FEM simulation (performed by F. Pantke, published in [BOS11C]) were used as the input for the machine learning process to gain a first set of reference success rates for the 200 × 300 mm steel plate scenario and the two described machine learning algorithms. In this case, the learned models stay valid as long as all load-induced stresses fall within the elastic range. As the prediction accuracy can be regarded as a measure of the general practicability of our approach, the success rates obtained in this evaluation will be used as benchmarks for further improvement of the hardware prototype as well as development of more sophisticated machine learning approaches.

For the sake of simplicity it was assumed that a metal plate was fixed at one (shorter) side, hanging free with one or multiple point loads $\{F_1(x_1, y_1), F_2(x_2, y_2), ..\}$ applied at different spatial positions (x,y) on the downside of the plate. The plate is initially bent due to the gravitational forces acting on the plate, shown in Figure 14.3.

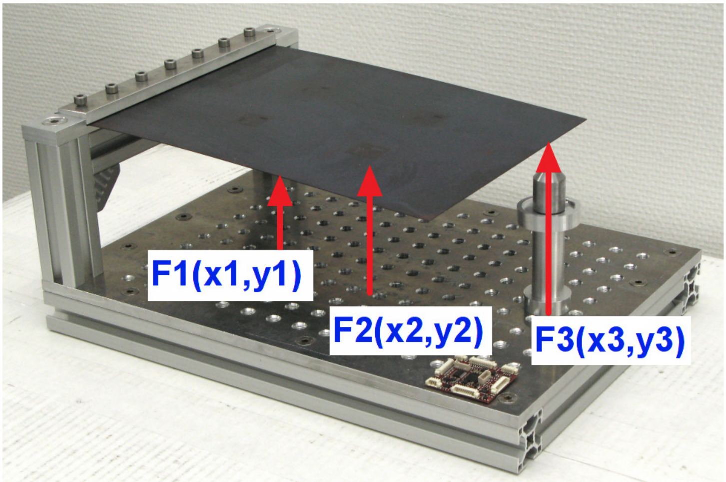


Fig. 14.3 *The model of the experimental setup for the FEM simulation.*

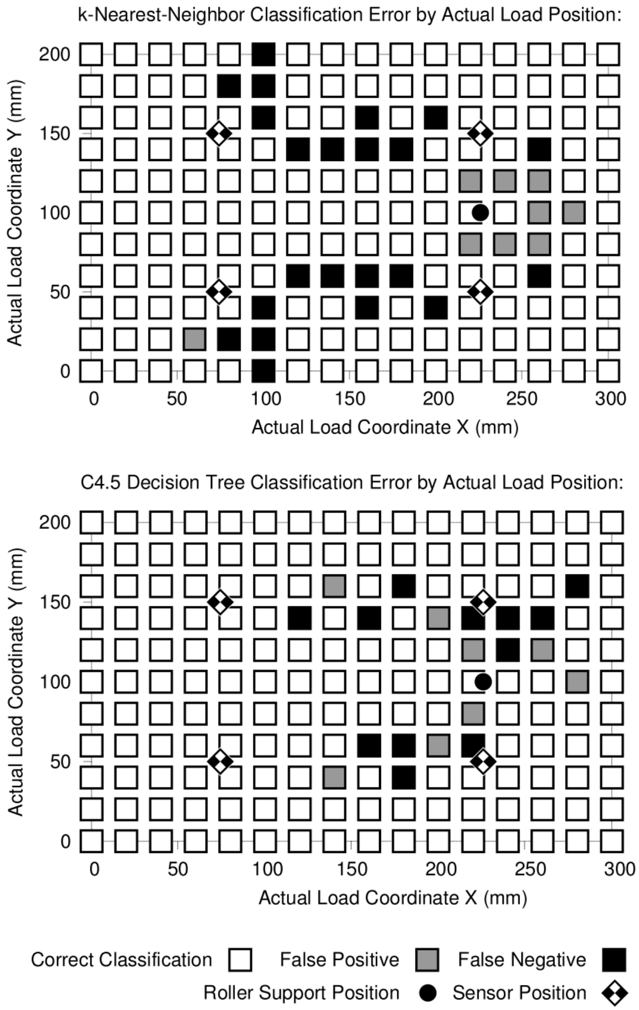


Fig. 14.4 *Classification results obtained with leave-one-out cross-validation on the training set*

Since only values from FEM simulation are used in the evaluation, i.e., no comparison with measured values from the physical prototype was made, the spike placed under the plate's bottom face was modelled as a roller support in the FEM model for sake of simplicity. For C4.5 and k-NN with $1 < k < 10$, a leave-one-out cross-validation (LOOCV) was conducted on the training set: Each singleton subset of the training set was used exactly once for querying the machine learning models constructed from its complementary set and comparing the result with the known correct values. In addition, the models obtained from the entire 352 element training set were queried using an intermediate test set INT consisting of 150 g weights placed exactly at the midpoints between the previously defined grid positions, resulting in further 10 × 15 load cases neither the coordinates nor masses of which appear anywhere in the training set.

Depending on the value chosen for k, the percentage of correct classifications achieved with k- NN varies between 89.77% and 90.91% in LOOCV and between 92% and 93.33% for INT. The C4.5 decision tree attained 94.32% correct classifications in LOOCV and 96.22% for INT. The locations at which false positives and false negatives were returned in LOOCV by C4.5 and k-NN with *k* = 4 are shown in Figure 14.4.

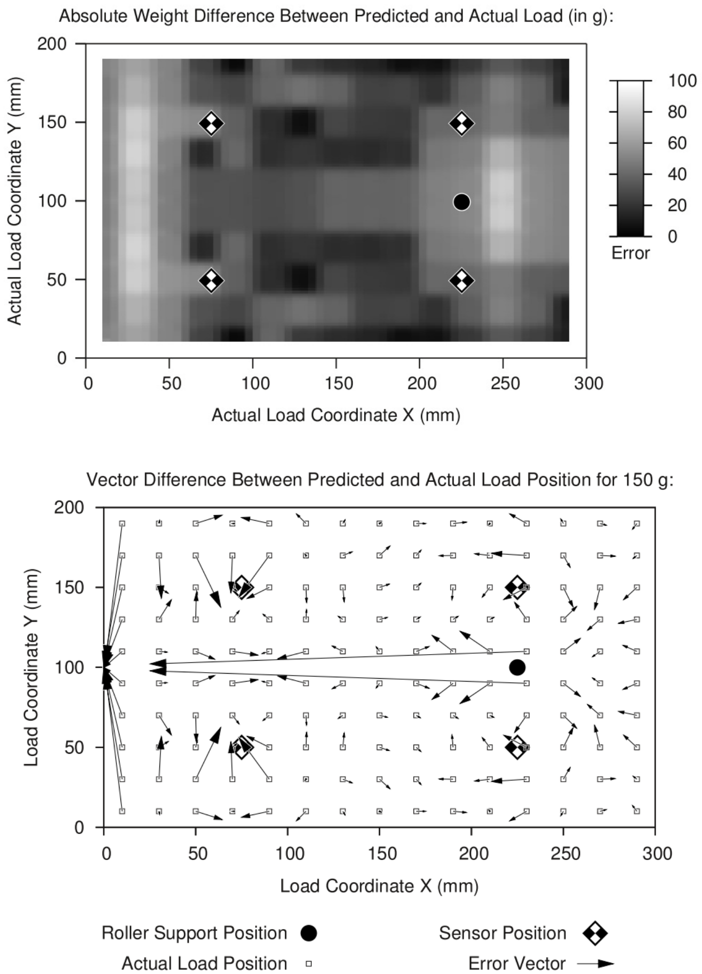


Fig. 14.5 *Regression error for 150 g weights placed at intermediate positions*

While at each position a 100 g and a 200 g load case were tested, no position produced wrong classifications for both masses. The numerical regression results obtained for the test set INT by k-NN with k = 4 are illustrated in Figure 14.5. The difference vectors from the actual to the estimated load positions are shown in the bottom vector field (median length 9.09 mm, average length 15.88 mm), whereas the absolute error in load mass estimation is visualized in the upper plot (median 20.39 g, average 25.38 g). Among all tested values for k, the median difference vector length of the load coordinate regression was smallest with k = 4, while the average was smallest with k = 2. In Figures 14.4 and 14.5, the locations of the strain sensors and the roller support are indicated with diamond and circle symbols, respectively. The fixture is located on the Y coordinate axis. It is clearly noticeable that the regression error is largest in the vicinity of the supports.

14.1.3 Experimental Results using Machine Based Learning with a Rubber Plate

Preliminary experiments were performed with a flat rubber plate (equipped with nine bi-axial strain-gauge sensors and the sensor network previously introduced, 70 mm sensor distance) and an experimental set-up shown in Figure 14.6 with circular weights.

Figures 14.6 and 14.7 show the analysis of the difference between measured and predicted load positions (position accuracy) retrieved by machine learning with two different sensor array configurations. The plots show the spatial vector difference between the predicted and observed position with a mean value below 25 mm and 50 mm, respectively.

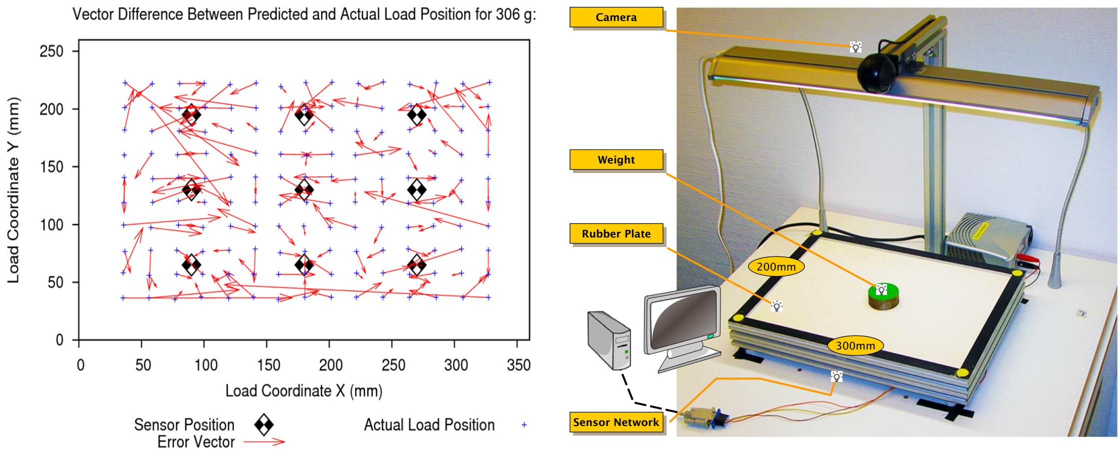


Fig. 14.6 *Experimental results of predicted load positions (306 g weight) with nine strain-gauge sensor pairs mounted on backside of a rubber plate (experimental test set-up shown on right side)* [BOS12F].

epubli, ISBN 9783746752228 (2018)

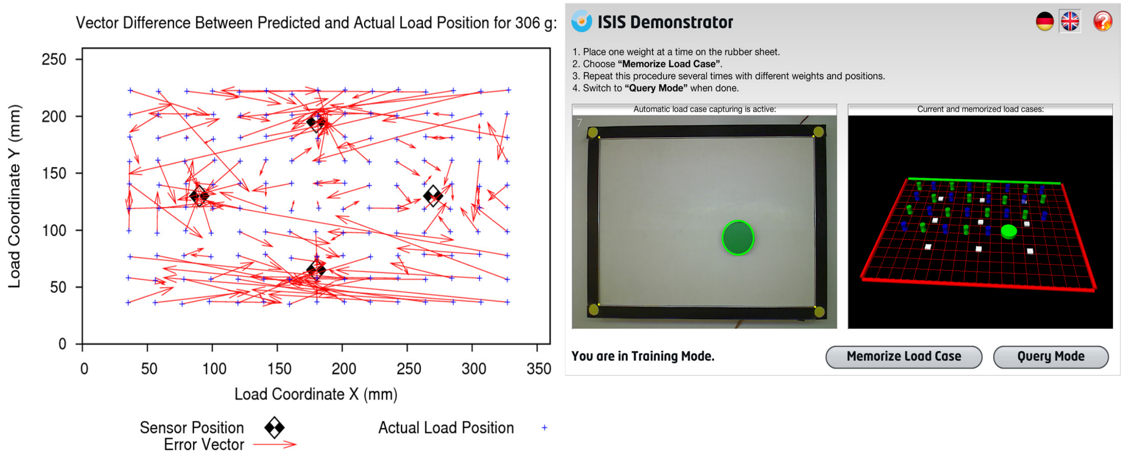


Fig. 14.7 *Experimental results of predicted load positions (306 g weight) with only four strain-gauge sensor pairs and ML graphical user interface (GUI, right side)* [BOS12F].

The training set consisted of two different masses (103 g and 306 g) and 150 positions. During learning mode the position (of the weight) was monitored with a camera mounted above the rubber plate together with acquired sensor data delivered by the sensor network.

A system with reduced number of sensors (Figure 14.7, 150 mm sensor distance) results in a decrease of prediction position accuracy in the boundary region, but is still usable for structure load monitoring especially in the middle area spawned by the edges of sensors.

The sensor network attached below the rubber plate consisted of nine single sensor nodes, each equipped with Analogue-Digital conversions and digital processing units (FPGA). Each sensor node was attached to one bi-axial aligned strain-gauge sensor pair. The sensor nodes were arranged in two-dimensional network, shown in Figure 14.8.

The digital processing unit of each sensor node was composed of communication modules supporting the SLIP routing protocol, introduced in Section 4.3, sensor signal acquisition with a zooming window ADC approach, noise filtering modules, and an RPC layer. All processing and communication modules were integrated in one FPGA SoC design (*Xilinx* Spartan 3, 1000k eq. gates) by using the *ConPro* High-level synthesis approach, introduced in Section 12.5. The nodes were connected with two unidirectional serial links. The adaptive path finding of the SLIP protocol allowed defective connections between nodes up to 20% with a loss of messages carrying the sensor data, which was in this early work collected periodically by an external computer connected to one of the sensor nodes.

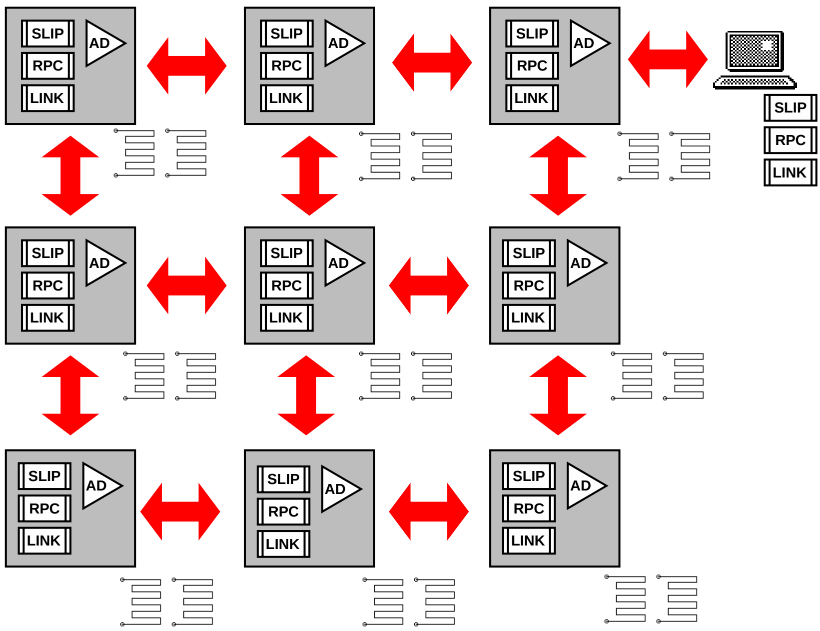


Fig. 14.8 *Two-dimensional Sensor Network with autonomous sensor nodes*

Sensor Signal Processing: Zooming ADC

Resistive sensors, like strain-gauge sensors, provide only a small relative change in resistance in the order of 1% resulting from a change of applied load in the considered operating range of the sensor. Using bridge configuration, providing a differential signal, require compensated sensors with small tolerances in strain and zero-load resistance parameters, actually not applicable to sensorial materials using, for example, printed sensors.

Assuming only one non calibrated and uncompensated resistive sensor, a zooming window approach (see Equation 14.2) can be used to match an initially unknown sensor to the measurement system preserving a high and full-range resolution, shown in Figure 14.9.

$$W(s) = k(s - off) \tag{14.2}$$

The data processing performs an initial (or periodically repeating) auto-calibration finding the centre of the operational window by using fast settling successive approximation, shown in Algorithm 14.1.

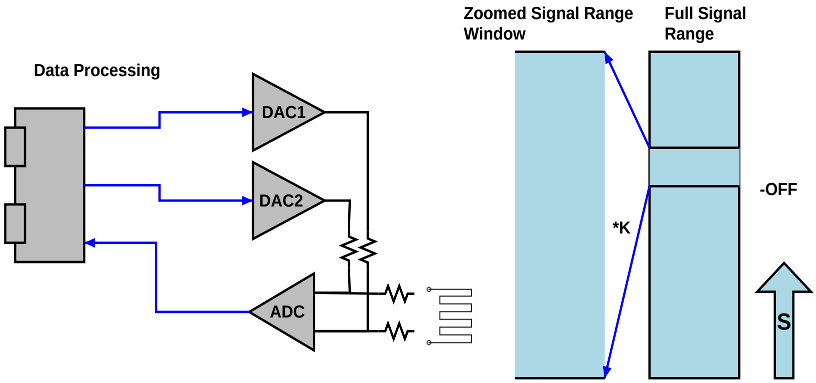


Fig. 14.9 *Zooming ADC system architecture used for signal acquisition of resistive sensors with high relative sensitivity and auto-calibration. (ADC: Analogue-to-Digital Converter with differential input, DAC: Digital-to-Analogue Converter)*

Alg. 14.1 *Auto calibration using successive approximation*

```
sar ← DIGITALRANGE/2;
DAC1 ← GAIN0, DAC2 <- 0;
WHILE sar <> 0 do begin
  IF ADC > DIGITALRANGE/2
    THEN DAC2 ← DAC2 + sar ELSE DAC2 ← DAC2 - sar;
  sar ← shift_right(sar,1); end;
off ← DAC2-DAC1;
```

14.2 Sensorial Material II: A Flat Perceptive Plate and Inverse Numeric

Structural Health Monitoring (SHM) of mechanical structures allows deriving not just loads, but also their effects to the structure, its safety, and its functioning from sensor data. A load monitoring system (LM) can be considered as being a subclass of SHM, which provides spatial resolved information about loads (forces, moments, etc.) applied to a technical structure, e.g., the mechanical structures of a robot manipulator arm..

One of the major challenges in SHM and LM is the derivation of meaningful information from sensor input. The sensor output of a SHM or LM system reflects the lowest level of information. Beside technical aspects of sensor integration the main issue in those applications is the derivation of a mapping function $F_m(S)$ which basically maps the raw sensor data input S, a n-dimensional vector consisting of n sensor values, to the desired information I, an m-dimensional result vector (see Figure 14.10).

Initially unknown external forces acting on a mechanical structure lead to a deformation of the material based on the internal forces. A material-integrated active sensor network integrating sensors, electronics, data processing, and communication, together with mobile agents can be used to monitor relevant sensor changes with an advanced event-based information delivery behaviour. Inverse numerical methods can compute finally the material response.

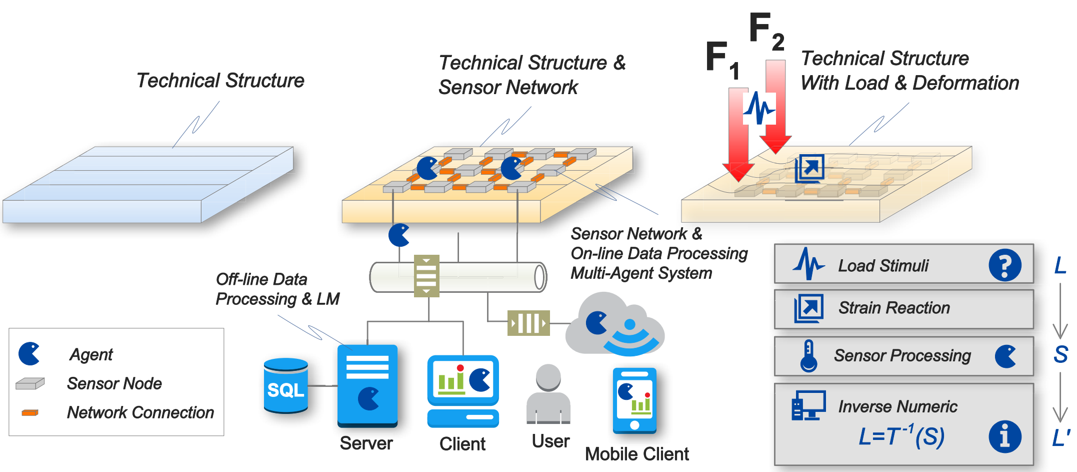


Fig. 14.10 A heterogeneous network environment consisting of a mechanical structure equipped with a sensorial system that reacts on externally applied load and a processing system performing inverse numerical computations.

The unknown system response for externally applied load L is measured by the strain sensor stimuli response S, finally computing an approximation of the response L'.

It can be shown that a hybrid data processing approach for material-integrated SHM and LM systems by using self-organizing and event-driven mobile multi-agent system (MAS) is suitable for this sensing system class, with agent processing platforms scaled to microchip level (PAVM/PCSP) which offer material-integrated real-time sensor systems, and inverse numerical methods providing the spatially resolved load information from a set of sensors embedded in the technical structure. Inverse numerical approaches usually require a large amount of computational power and storage resources, unsuitable for resource constrained sensor node implementations. Instead, off-line computation is performed, with on-line sensor processing by the agent system. Commonly off-line computation operates on a continuous data stream requested by the off-line processing system delivering sensor data continuously in fixed acquisition intervals, resulting in high communication and computational costs. Here the sensor preprocessing MAS delivers sensor data event-based if a change of the load was detected (feature extraction), reducing network activity and energy consumption of the entire system significantly. The basic SOMAS behaviour were introduced in Chapter 9.

One common approach in SHM is the correlation of measured data resulting from an induced stimuli at run-time (system response) with data sets retrieved from an initial (first-hand) observation, which makes it difficult to select damage relevant features from the measurement results. Other variants are based on statistical methods or neural network approaches. Related work in [FRI07], [FRI01], [CAR06], and [HUH11] can be referred for examples illustrating the variety of possible approaches.

Inverse methods generally belong to the first class of approaches since they are based on a mechanical model T of the technical structure mapping loads to sensor signals. Given a sensor signal vector s (serialization of a two-dimensional sensor matrix S), inverse methods try to stably "invert" the mapping T, that is, to find a stable solution x to the problem $Tx = s$. Since measured signals and the underlying physical model always contain numerical and modelling errors, inverse methods do not attempt to find an exact solution to the latter equation. Indeed, inversion problems, in particular those with incomplete data, are usually extremely ill-conditioned, meaning that small errors in the signals or the model lead to huge errors in any "solution" gained by such a naïve approach. Instead, inverse methods try to stabilize the inversion process, using, e.g., one of the following techniques:

- Pick amongst all solutions to $Tx = s$ the one that minimizes a certain functional - the simplest functional to minimize would be the Tikhonov functional

$$l \mapsto \|Tl - s\|_2^2 + \alpha\|l\|_2^2, \tag{14.3}$$

where $\alpha > 0$ and $\|\cdot\|_2$ is the 2-norm of a vector, but different and more complicated variants might also be convenient choices.

- Alternatively, consider any iterative method driving the residual ***Tl-s*** and stabilize the inversion by stopping the iteration when the norm of the residual is about the magnitude of the expected signal and modelling error.

The mechanical model of the structure under investigation allows in particular the pre-computation of a sufficiently accurate discretization of the forward mapping ***T*** linking loads with measured signals. Moreover, this pre-computation allows associating each sensor to an individual signal level that might potentially be critical for the entire structure (details can be found in [BOS14C]).

Hence, when a load change that is potentially critical is detected by one the material-integrated sensors, the signals measured by all sensors are propagated to an exterior CPU. An alternative way is that merely those sensors noting a critical load change start to propagate their signals to the exterior processing system (computer). The propagated signals are then fed into a regularization scheme that is able to stably invert signals into loads. Several algorithms can be used at this point: A classical and well-known inversion method is Tikhonov regularization, minimizing the quadratic Tikhonov functional

$$l \mapsto \|Tx - s\|_2^2 + \alpha\|l - l_0\|_2^2, \tag{14.4}$$

where $\alpha > 0$ is a (small) regularization parameter, $\|\cdot\|_2$ is the 2-norm, and l_0 is some a-priori guess for the exact solution. The minimum l is the unique solution of the linear system

$$(T^*T + \alpha)l = T^*s + \alpha l_0, \tag{14.5}$$

where ***T**** denotes the transpose of ***T***.

The solution to this system is hence computed rapidly with low cost if one is able to pre-compute a singular value decomposition of the matrix ***T***. The disadvantage of this inversion scheme is that reconstructions of discontinuous loads, in particular with small support, are smoothed out which makes the precise location of the support of a load difficult. Several iterative inversion techniques such as the steepest descent method or the conjugate gradient method applied to $T^*Tl= T^*g$ avoid this disadvantage. Further, they merely require the ability to compute matrix-vector products and a (cheap stopping) rule to stabilize the inversion. The class of iterative inversion methods also includes the so-called Landweber iteration and its variant, the so-called iterative soft shrinkage. The disadvantage of the latter two techniques is their slow convergence, and the huge number of iterations are necessary to compute accurate inversions [ENG96][KIR96].

Combining Self-organizing Multi-Agent Systems (SoMAS) and event-based sensor data distribution with inverse numerical methods into a hybrid data processing approach has several advantages: First, the (possibly distinct) critical level for an individual sensor signal can be pre-computed for each sensor position individually. Second, depending on the a-priori knowledge on the expected loads on the structure, a suitable regularization technique can be chosen as inversion method, promoting specific features of the expected loads.

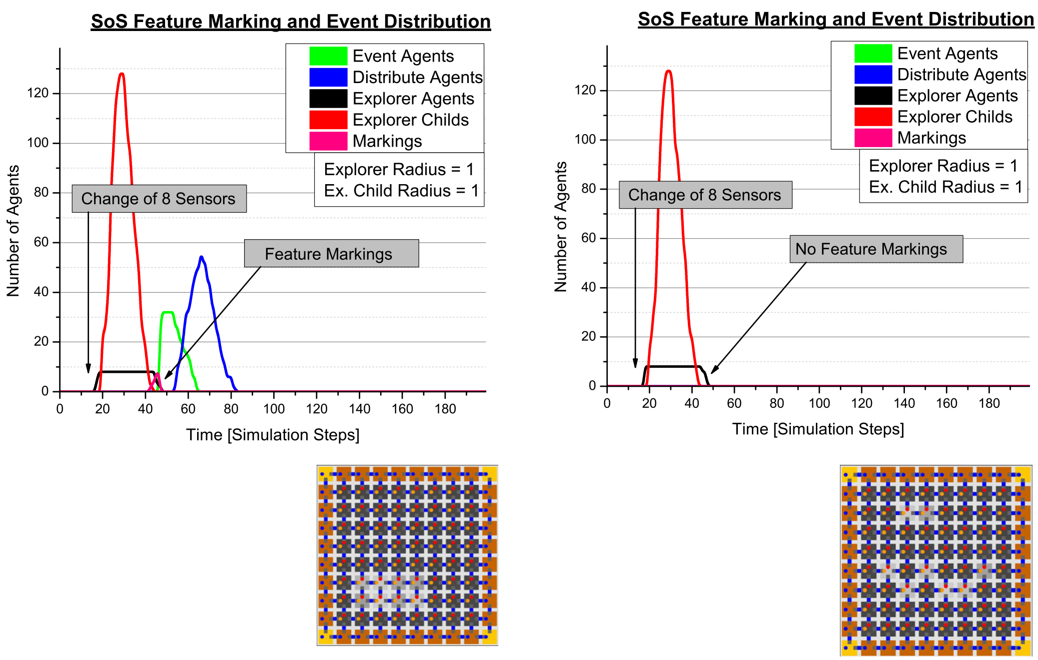


Fig. 14.11 *Analysis results of the agent population obtained from the multi-agent simulation of the feature marking combined with the event-based data distribution. Left: with a contiguous cluster of 8 sensors (correlated), Right: 8 sensor stimulation scattered around in the network (no correlation)*

The simulation of the full sensor processing MAS consisting of the feature detection SoMAS (see Section 9.2) and the adaptive event-based sensor data distribution (see Section 9.3) triggered by the feature detection SoMAS with an exemplary sensor network poses a strong correlation of the sensor stimuli events and the temporal and spatial population with agents.

Figure 14.11 summarizes the analysis of the MAS simulation giving the agent population for two different cases: (1) A correlated cluster of stimulates sensors (left) compared with (2) An uncorrelated cluster (right). Using more realistic sensor stimuli based on FEM simulation (based on experiments shown in Figure 9.5) leads to similar results regarding the spatial and temporal population of agents in the network, shown in Figure 14.12 with the corresponding exemplary load cases of the steel plate (200x300 mm, 1 mm thickness) [BOS14F].

Five pairwise different loads $l^{(1)}, .. , l^{(5)}$ with different characteristics and acting on different parts of the steel plate are used.

The reproduction after successful feature detection leads to a significant increase of the explorer child agent population, which has advantages only in the occurrence of large extended correlated regions (the feature is the boundary of the region).

The uncorrelated case does not trigger the creation of event and distribute agents. The simulation results show that the temporal and spatial exploration does not depend on the presence of a feature (correlation) if the explorer radius is limited to one. Otherwise, either explorer reproduction or diffusion occurs, with a temporal broadening and/or increase in the agent population.

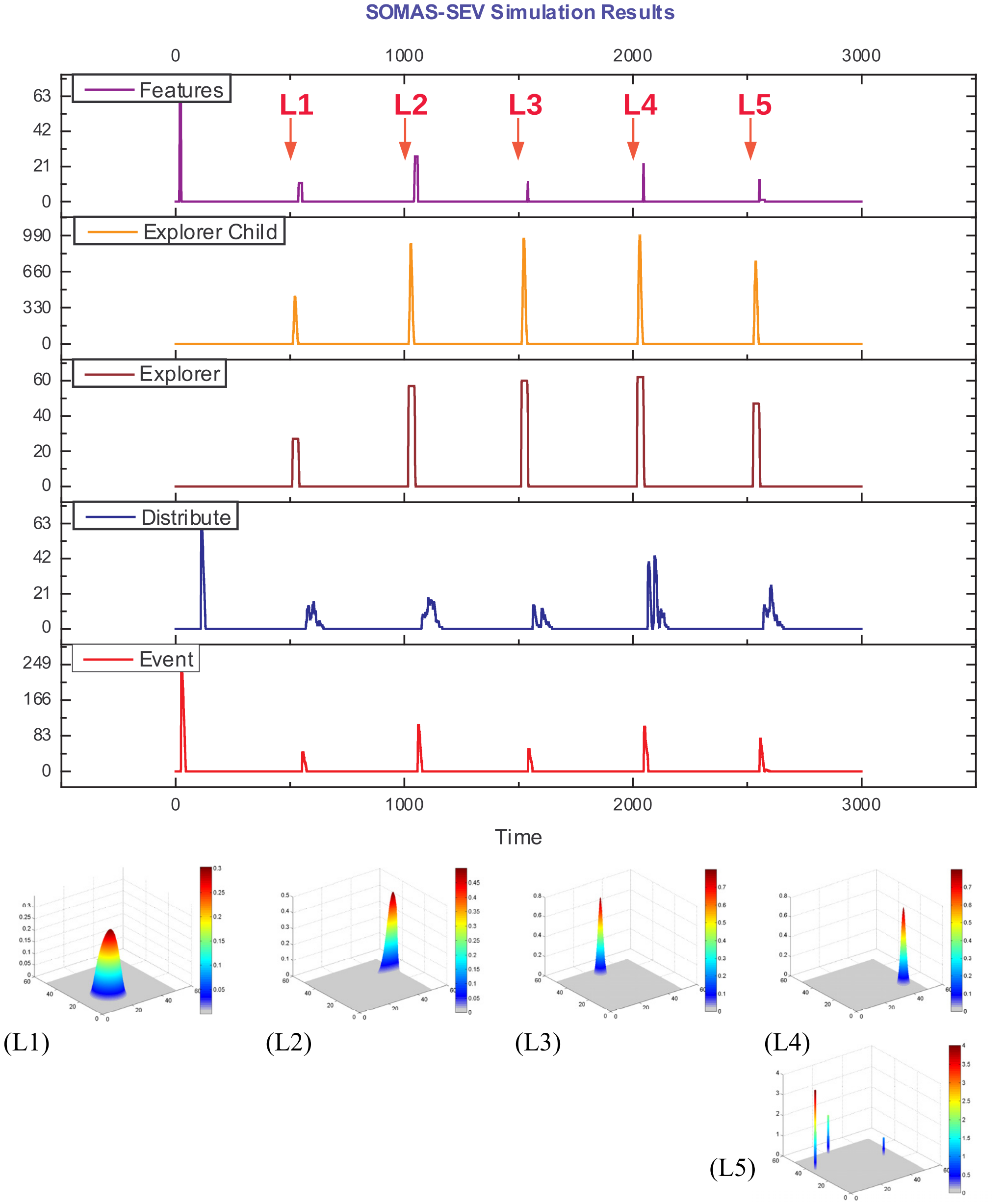


Fig. 14.12 *Agent population of the feature detection SoMAS and event-based sensor data distribution in the sensor network with real load cases (Li: i-th load case data set)*

14.3 Sensorial Material III: A Perceptive Modular Robot Arm

Intelligent behaviour of robot manipulators become important in unknown and changing environments. Emergent behaviour of a machine arises intelligence from the interactions of robots with its environment. Sensorial materials equipped with networks of embedded miniaturized smart sensors can support this behaviour.

Integrated autonomous decentralized sensor networks provide perception in a robot arm manipulator. Each sensor network is connected with a set of strain gauge sensors mounted on a flexible polymer surface, delivering spatial resolved information of external forces applied to the robot arm, required for example for obstacle avoidance or for manipulation of objects.

Each autonomous sensor node provides communication, data processing, and energy management implemented on microchip level.

Commonly a high number of strain gauge sensors are used to satisfy a high spatial resolution. The approach uses advanced Artificial Intelligence and Machine Learning methods for the mapping of only a few non-calibrated and non-long-term stable noisy strain sensor signals to spatially resolved load information and a decentralized data processing approach to improve robustness. Robustness in the sensor network is provided by

1. Autonomy of sensor nodes;
2. Smart adaptive communication to overcome link failures and to reflect changes in network topology;
3. Using intelligent adaptive algorithms. Robust cooperation and distributed data processing is achieved by using Mobile Agent systems [WAN03].

Agent behaviour and cooperation is implemented on microchip level [BOS12A]. The central aim is to derive useful information constrained by limited computational power and noisy sensor signals unable to be captured by a complete system model. Machine Learning (ML) methods are capable to map an initially unknown n-dimensional set of input signals to a m-dimensional output set of information like the position and strength of applied forces [MIT97].

Without any interaction and material model Machine Learning requires a training phase. Additional material models and FEM simulation can reduce or avoid the training phase [BOS11C].

The training set contains recorded load positions, masses and classification results for different load cases determined via sensor measurement.

The hyper-elastic behaviour of polymers reduces the long-term prediction accuracy of learned models as well as the consistency with FEM output, requiring Machine Learning models that automatically adjust their output to the structure's ageing process.

The robot manipulator consists of actors (joint drives) and intersection elements with integrated smart sensor networks. Distributed data processing is provided by mobile agents. The agent behaviour is implemented on hardware-level and SoC designs, shown in Figure 14.13. The intersection element connects two joint actors with a rigid double-pipe construction with surrounded two opposite placed load sensitive skins (bend rubber plate), equipped each with four strain-gauge sensor pairs (bi-axial aligned). Each sensor pair is connected to a sensor node providing parallel data processing, agent behaviour implementation, and communication/networking. All sensor nodes are arranged in a mesh-like network connected with serial point-to-point links. Communication is established by a smart and robust routing protocol. This application bases on the flat rubber plate experiments pointed out in Section 14.1.

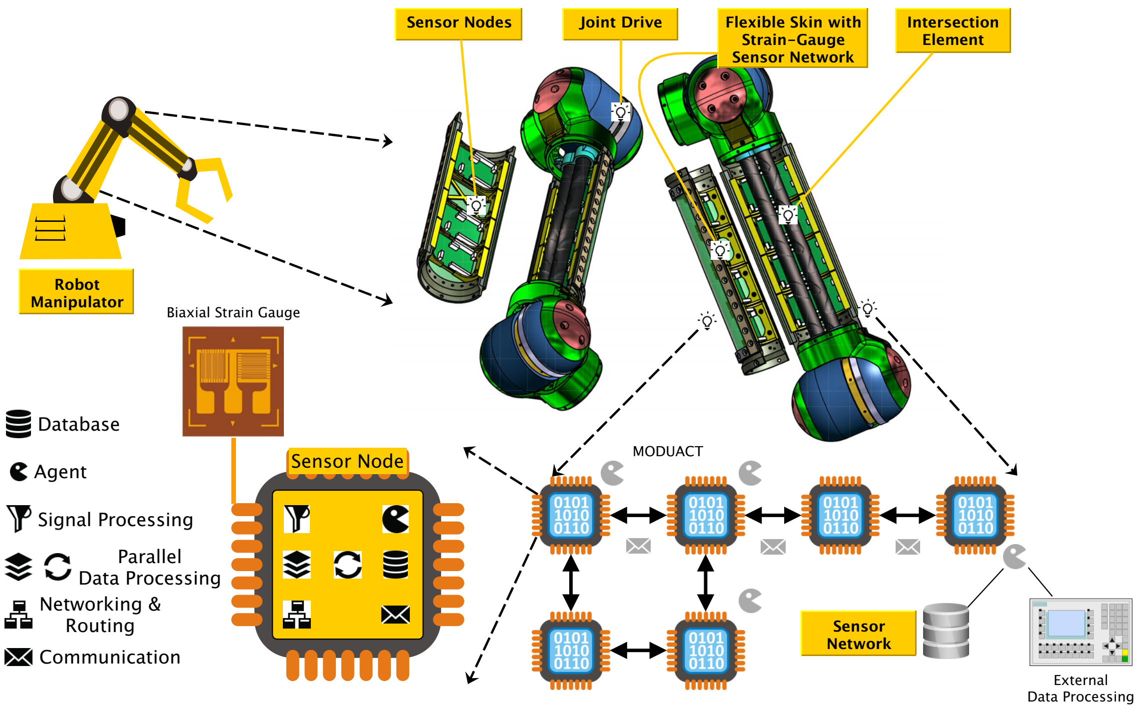


Fig. 14.13 *Robot arm manipulator intersection element equipped with smart sensor networks providing perception information of external applied load forces.*

14.4 Sensorial Material IV: A Perceptive Robotic Gripper

The dynamic process of grasping different kinds of objects that are pressure sensitive is difficult to handle with classical feedback controllers based on few force sensor values acquired and processed outside of the gripper structure. Side effects like slipping can not be detected at all or too late. Miniaturized smart sensors embedded in structures like grippers can significantly increase the perception of the environment with which a structure interacts.

A high-density network of strain-gauge sensors distributed in/on the gripper structure providing local sensor signal-to-information computation can deliver much more suitable information.

Traditionally, strain-gauge sensors are used to measure an applied force in a specific direction. The analogue signal acquisition is difficult due to low noise immunity of weak input signals. External signal acquisition with a large distance from sensor to electronics raises noise and reduces signal-to-noise ratio and resolution.

We propose and demonstrate the integration of an active smart sensor network into a mechanical gripper structure (finger). The network consists of several highly miniaturized low-power sensor nodes providing sensor signal acquisition, data processing, and communication. Each sensor node can handle up to two strain-gauge sensors detecting different forces at different positions of the gripper structure. The relation between strain and force is derived from FEM simulation of the gripper structure under certain load conditions.

Each node performs sensor signal acquisition using a zooming ADC approach, sensor data evaluation, and auto-calibration. Hence, non-calibrated and non-long term-stable sensors can be integrated and used, a prerequisite for robust sensorial materials.

It can be demonstrated that an integrated sensor network leads to increasing functionality and robustness.

A smart communication protocol is used to provide robust and fault-tolerant communication between nodes and an external interface, for example, a generic processor-based controller.

Beside the collection of single force values measured at different positions of the gripper, temporal and spatial composition information derived from the set of measured forces can be computed using data fusion, performed by the nodes of the sensor network itself using distributed computing algorithms. These are overload conditions, force gradients, object recognition and classification, and other higher-level information, which can be computed.

A multi-agent system is used for a decentralized and self-organizing approach of data processing in the distributed system, i.e., the sensor network, enabling the mapping of distributed data sets to related information required for the object manipulation.

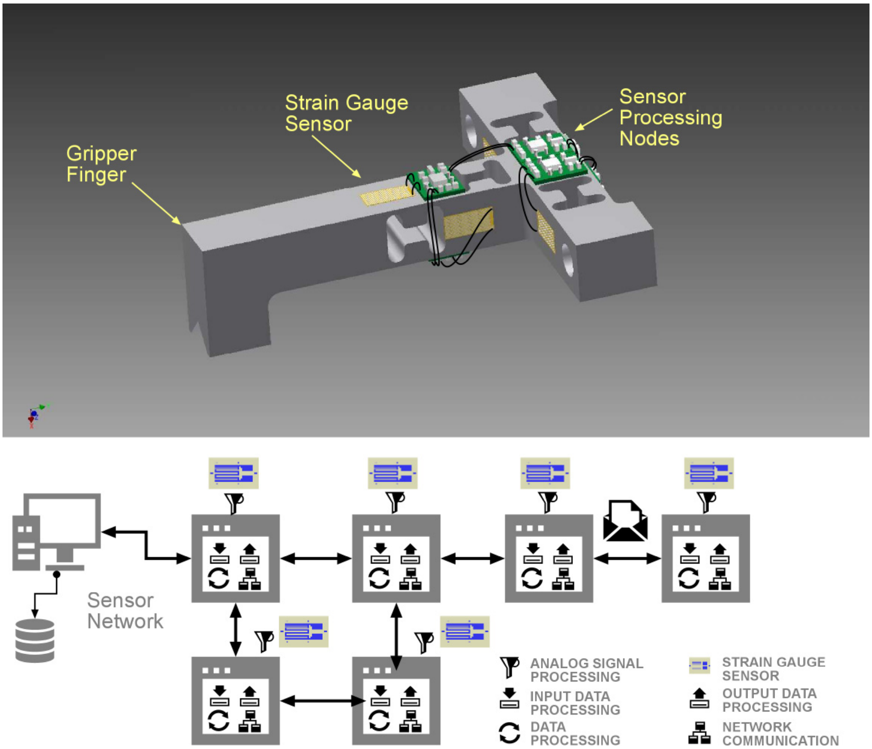


Fig. 14.14 *One gripper finger equipped with a sensor network connecting nodes each servicing a strain gauge sensor mounted on the gripper structure. The network topology is shown in the lower part (mech. structure based on* [BOS12D]*)*

Mechanical grippers are key components of handling devices in automated assembly systems. For complex handling tasks, these grippers have to be equipped with additional force-measuring modules.

The proposed gripper fingers, based on work in [BOS12D], contain six single-force sensors and can measure forces along multiple axes, shown in Figure *14.14*.

Each sensor is connected to an active sensor node consisting of signal and data processing, communication modules, and power regulation. Each sensor node can connect with neighbour nodes (up to four links).

14.5 Sensor Clouds: Adaptive Cloud-based Design and Manufacturing

This section outlines the architecture for additive and adaptive manufacturing based on a closed-loop sensor processing approach with data mining concepts combined with Internet-of-thing architectures.

Additive and adaptive cloud-based design and manufacturing are attractive in the field of robotics, not only limited to industrial production robotics, mainly targeting service robots and semi-autonomous carrier robots. In cloud-based manufacturing, the consumer of the products is integrated in the cloud-based manufacturing process [WU12], directly involved in the manufacturing process using distributed cloud computing and distributed storage solutions.

Robots can be considered as active and autonomous data processing units that are commonly already connected to computer networks and infrastructures. Robots use inherent sensing capabilities for their control and task satisfaction, commonly using integrated sensing networks with sensor preprocessing, deriving some inner state of the robot, for example, mechanical loads applied to structures of the robot or operational parameters like motor power and temperature. The availability of the inner perception information of robots enable the estimation of working and health conditions initially not fully considered at design time. The next layer in cloud-based adaptive manufacturing can be the inclusion of the products themselves delivering operational feedback to the current design and manufacturing process, leading to a closed-loop evolving design and manufacturing process with an evolutionary touch, shown in Figure *14.15*. This evolutionary process adapts the product design, for example the mechanical construction, for future product manufacturing processes based on a back propagation of the perception information (i.e., recorded load histories, working and health conditions of the product) collected by living systems at run-time. The currently deployed and running series of the product enhances future series, but not in the traditional coarse-grained discrete series iteration. This process can be considered as a continuously evolving improvement of the robot by refining and adapting design parameters and constraints that are immediately migrated to the manufacturing process. A robot consists of a broad range of parts, most of them are critical for system failures. The most prominent failures are related to mechanical and electro-mechanical components, which are caused by overload conditions at run-time under real conditions not to be considered or unknown at initial design time.

The integration of robots as product and their condition monitoring in a closed-loop design and manufacturing process is a challenge and introduces distributed computing and data distribution in strong heterogeneous processing and network environments. One major question to be answered is the sensing of meaningful condensed product condition information and the delivery to the designer and factory. It is a similar issue like arising in the Internet-of-Things domain. New unified data processing and communication methodologies are required to overcome different computer architecture and network barriers, delivered by the unified distributed data processing model of the mobile agents that are self-contained and autonomous virtual processing units. The mobile agents represent mobile computational processes that can migrate on the Internet and as well in sensor networks.

Multi-agent systems (MAS) represent self-organizing societies consisting of individuals following local and global tasks and goals including the coordination of information exchange in the design and manufacturing process.

Agents are already deployed successfully for scheduling tasks in production and manufacturing processes [CAR00B], and newer trends poses the suitability of distributed agent-based systems for the control of manufacturing processes [LEI15], facing not only manufacturing, but maintenance, evolvable assembly systems, quality control, and energy management aspects, finally introducing the paradigm of industrial agents meeting the requirements of modern industrial applications. The MAS paradigm offers a unified data processing and communication model suitable to be employed in the design, the manufacturing, logistics, and the products themselves.

The scalability of complex industrial applications using such large-scale cloud-based and wide-area distributed networks deals with systems deploying thousands up to a million agents. But the majority of current laboratory prototypes of MAS deal with less than 1000 agents [LEI15]. Currently, many traditional processing platforms cannot yet handle big numbers with the robustness and efficiency required by industry [MAR05][PEC08]. In the past decade the capabilities and the scalability of agent-based systems have increased substantially, especially addressing efficient processing of mobile agents.

The programmable agent processing platform *PAVM* introduced in Chapter 7 can be deployed in such strong heterogeneous network environments, ranging from single microchip up to WEB *JavaScript* implementations, being fully compatible on operational and interface level. Multi-agent systems can be successfully deployed in sensing applications, for example, structural load and health monitoring, with a partition in off- and online computations, as introduced in Section *14.2*. Distributed data mining and Map&Reduce algorithms are well suited for self-organizing MAS. Cloud-based computing, as a base for cloud-based manufacturing, means the virtualization of resources, i.e., storage, processing platforms, or information.

Traditional closed-loop processes request data from sources (products, robots) by using continuous request-reply message streams. This approach leads to a significant large amount of data and communication activity in large-scale networks. Event-based sensor data and information distribution from the sources of sensing events, triggered by the data sources (the robots) themselves, can improve and reduce the allocation of computational, storage, and communication resources significantly.

- A cloud in terms of data processing and computation is characterized by and composed of:
- A parallel and distributed system architecture
- A collection of interconnected virtualized computing entities that are dynamically provisioned
- A unified computing environment and unified computing resources based on a service-level architecture
- A dynamic reconfiguration capability of the virtualized resources (computing, storage, connectivity and networks)

Cloud-based design and manufacturing is composed of knowledge management, collaborative design, and distributed manufacturing. Adaptive design and manufacturing enhanced with perception delivered by the products incorporates finally the products in the cloud-based design and manufacturing process.

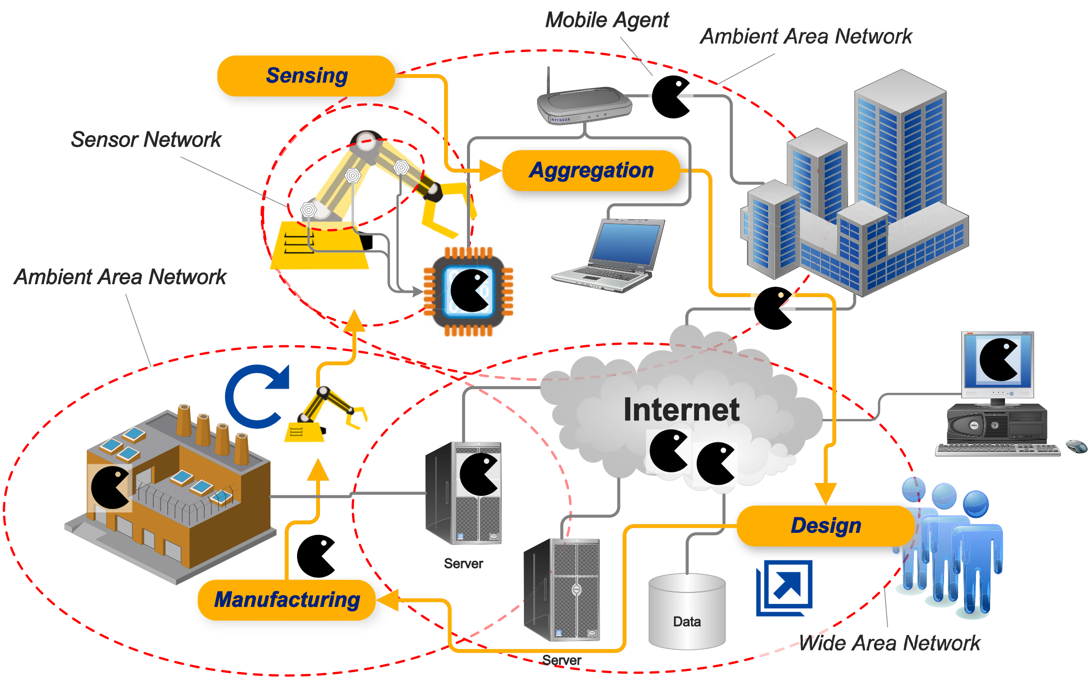


Fig. 14.15 *Additive and adaptive Manufacturing with back propagation of sensing data using mobile agents from robots to the design and manufacturing process resulting in continuous series improvements.*

Agent Classes. The entire MAS society is composed of different agent classes that satisfy different sub-goals and reflect the sensing-aggregation-application layer model: event-based sensor acquisition including sensor fusion (Sensing), aggregation and distribution of data, preprocessing of data and information mapping, search of information sources and sinks, information delivery to databases, delivery of sensing, design, and manufacturing information, propagation of new design data to and notification of manufacturing processes, notification of designer, end users, update of models and design parameters. Most of the agents can be transferred in code frames with a size lower than 4kB, and depends on the data pay-load they carry. At run-time, agents are instantiated from these different classes, and agents can change to a subclass behaviour depending on current sensing, goals, and their inner state.

Considering adaptive design and manufacturing that consumes the product information to adapt products semi-continuously, an exponential-like increase of data and information can be expected.

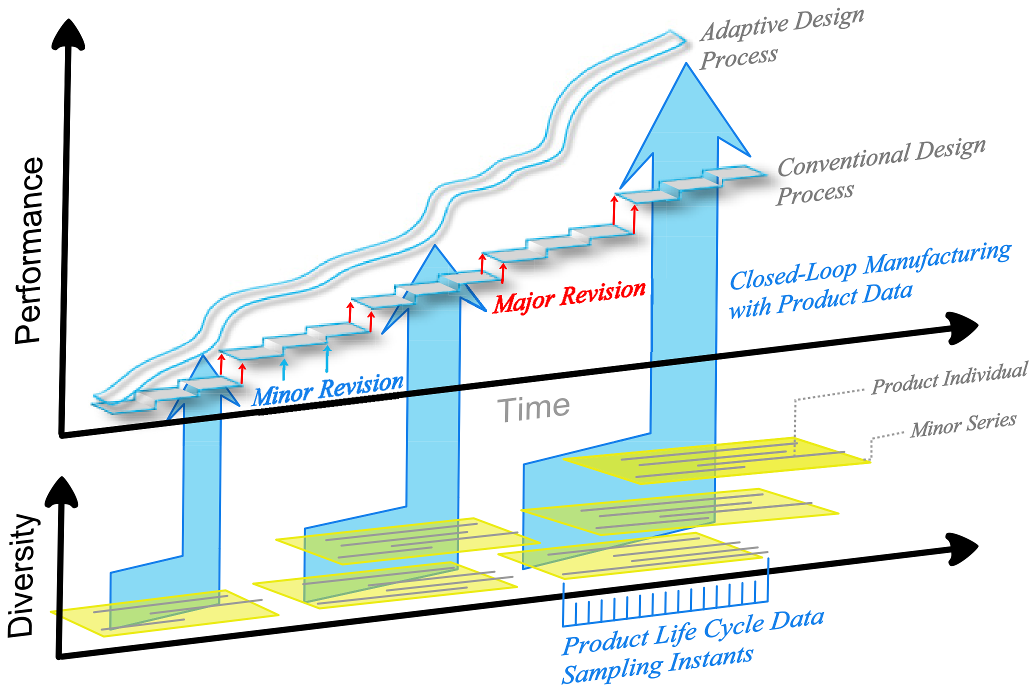


Fig. 14.16 *Effect of product life cycle data sampling on adaptive design and manufacturing processes. A temporal increase in product diversity and the number of product individuals result in an exponential-like increase of the amount and density of sensing information processed and distributed by the MAS* [BOS14G].

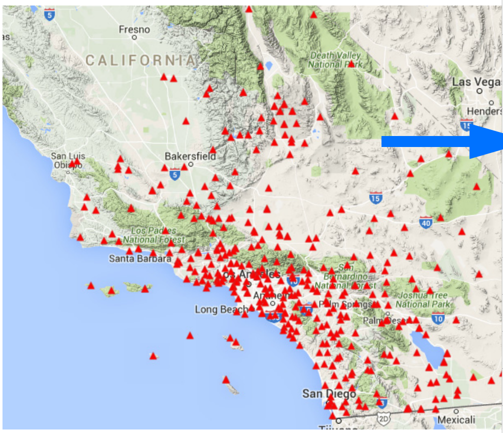
This is caused by minor revisions of products, classical happening commonly in monthly intervals, now shrinking to day or hour intervals, and that result in a significant increase of diversity and the number of individuals, shown in Figure 14.16. It is expected that this data explosion can only be handled by SoMAS.

14.6 Sensor Networks: Distributed Earthquake Monitoring

Among micro-scale material-integrated sensing systems there are large-scale wide-area sensor networks. In [BOS16C] and [BOS17C] the *JAM* agent processing platform (see Section 8.1) was proposed for a seismic earthquake monitoring system. Self-organizing mobile agents are suitable for on-line distributed seismic data evaluation and information retrieval.

The seismic stations of the South California CI network are mapped on a two- dimensional grid with spatial proximity, shown in Figure 14.17. Each (virtual) node in the network starts a resident node agent responsible for data sampling, reduction, and for the creation and notification of a learner agent. If the node agents detect vibration activity beyond a threshold, they will notify the learner agents via tuple-space interaction. The learner will send out exploration agents that collect neighbourhood data, finally back delivered to the learner agent.

One major challenge in large-scale seismic networks is data reduction. The raw sensor data contains temporal resolved seismic recordings of at least three sensors (horizontal East, horizontal North, and vertical acceleration sensors) with a time resolution about 10ms, resulting in a very high-dimensional data vector. Additionally, a seismic sensor samples only noise below a threshold level, mainly resulting from urban vibrations and sensor noise itself.



RCT	VES	MLAC	SPG	ISA	TIN	CWC	JRC2	CGO	CLC	MPM	GRA	SLA	FUR	SHO
PHL	SMM	TFT	BAK	WER	MAG	ARV	TEH	WBS	LRL	CCC	GSC	DSC	TUQ	LDF
SMB	LCP	FIG	MPP	MPI	LJR	LDR	EDW2	EDW	SBB2	LMR2	RRX	NBS	HEC	NEE
SDP	NJQ	SYP	SBC	WGR	OSI	PDE	BTP	ALP	LEV	VCS	LKL	ADO	VTV	DAN
USB	STC	SIO	SES	MOP	SMV	QUG	LFP	RIN	CHF	TA2	LUG	BBR	JVA	PDM
TOV	AGO	WSS	NOT	HLL	DEC	PAS	CBC	MIK	KIK	MWC	BFS	SBPX	MCT	IRM
LGU	SPF	DJJ	GSA	CRP	CAC	RUS	RIO	PDU	FON	CLT	CFS	HLN	SVD	BLA
SCZ2	SMS	PDR	LCG	USC	WTT	LGB	WLT	OLI	CHN	MLS	RVR	RSB	RSS	BEL
LAF	LTP	DLA	LBW1	BRE	FUL	SRN	CRN	PER	BBS	MSJ	SLR	DEV	PLC	MGE
RPV	MIS	FMP	STS	LLS	SAN	OGC	STG	PLS	DGR	AGA	THX	CTC	BC3	BLY
SNCC	SBI	CIA	KML	SDD	BCC	GOR	CAP	PLM	DNR	BOR	SAL	NSS2	RXH	SSW
SCI2	SDG	SDR	DPP	OLP	EML	BAR	JCS	DVT	ERR	SWS	WES	DRE	BTC	GLA

Fig. 14.17 *Mapping of internet-connected seismic station network on 2d grid*

The vibration (acceleration) is measured in two perpendicular horizontal and one vertical directions. This gives a significant information for an earthquake recognition and localization. The data reduction is performed by a node agent present on each seismic measuring station platform. Only the compact string patterns are used as an input for the distributed learning approach. Based on this data, the learning system should give a prediction of an earth-quake event and a correlation with past events. To deploy regional learning for a spatial ROI, seismic stations should be arranged in a virtual network topology with connectivity reflecting spatial neighbourhood, e.g., by arranging all station nodes in a two-dimensional network. The virtual links between nodes are used by mobile agents for exploration and distribution paths. They do not necessarily reflect the physical connectivity of station nodes.

The deployment of agents in the network, the agent behaviour, and their interaction performing distributed learning were already discussed in Section *10.6.4*. Using JAM agents, scalable and event-based sensor data processing is provided with Machine Learning as a platform service. Agents can carry learned models (mobile learner) without carrying the learner code. Incremental learning avoids an accumulated database carried by agents.

In [KON16], smart phones were successfully used to enhance the earthquake prediction by extending the seismic database with sensor data from mobile devices. In [POU15], an open participatory platform for privacy-preserving social mining (Planetary Nervous System) was introduced, i.e., basically a virtualization of sensors that can profit from the proposed agent framework. The distributed learning system deployed in the seismic station network using the local station data can be extended by devices from such ubiquitous networks, which can execute the learner agents collecting sensor data (vibration, air pressure, temperature) from such devices. In contrast to seismic stations located at fixed and well-known positions, mobile devices change their position dynamically.

The mobile learner carrying an already learned spatially local model in a specific region, can migrate to mobile devices in this region and performs further learning or prediction. The extension of earthquake analysis with a large number of ubiquitous mobile devices can aid to improve disaster management significantly by providing spatially fine resolved sensor and event data covered by a high node density. Furthermore, facility sensor networks can be included providing additional information about the buildings (health) state (illustrated in Figure *14.18*).

The *JAM* platform fits well in such large-scale strong heterogeneous and changing environment consisting of a broad diversity of devices: Seismic stations in buildings connected via the Internet, seismic stations on sea connected via satellite links or radio, servers, mobile devices connected via mobile networks or WLAN.

Fig. 14.18 *Combining seismic monitoring networks with ubiquitous sensing by smartphones*

14.7 Crowd Sensing

Most crowd sensing platforms are using cloud- or centralized data base approaches for the aggregation and processing of user and sensor data, e.g., McSense [CAR13] or *Nervousnet* [POU15], based on a client-to-server architecture, though supporting distributed pre-processing.

The Planetary Nervous System (*Nervousnet*) is an environment and platform consisting of sensors that process a set of input streams of data generated from physical or virtual sensors. The environment defines the context within that the virtual sensor operates to generate its output stream [POU15].

The platform provides ubiquitous social mining as a public service. Sensing systems consist of three different functional layers: Sensing, Aggregation, and Application. All three layers can be represented by virtual sensors and agents. Data sharing, data collection, and data fusion are main building blocks of such a system. In contrast to traditional embedded sensor networks, social mining in public networks requires privacy rules, as discussed in [MUS16]. Sensing devices, e.g., smart phones, commonly interact with a Cloud-like service architecture (device-to-cloud communication). Crowd sensing has already been successfully applied to different purposes. Discussed in the previous section, in [KON16], smart phones were used to compose a seismic network in urban environments for spatially fine-grained earthquake monitoring. But as in the current *Nervousnet* approach, the distributed data is evaluated either locally or at a central instance. With the emerging IoT, Crowd Sensing is extended by Things Sensing.

Cloud-based computing with MAS, e.g., as a base for crowd sensing and participatory social mining use cases, means the virtualization of resources, i.e., storage, processing platforms, sensing data or generic information. Mobile Agents reflect a mobile service architecture. Commonly, distributed perceptive systems are composed of sensing, aggregation, and application functional layers.

The scalability of complex ubiquitous applications using such large-scale cloud-based and wide area distributed networks deals with systems deploying thousands up to a million agents.

The JAM platform and mobile JAM agents can be used to combing the concept of virtual sensors, crowd sensing, and distributed event-based data processing in string heterogeneous environments, shown in Figure 14.19.

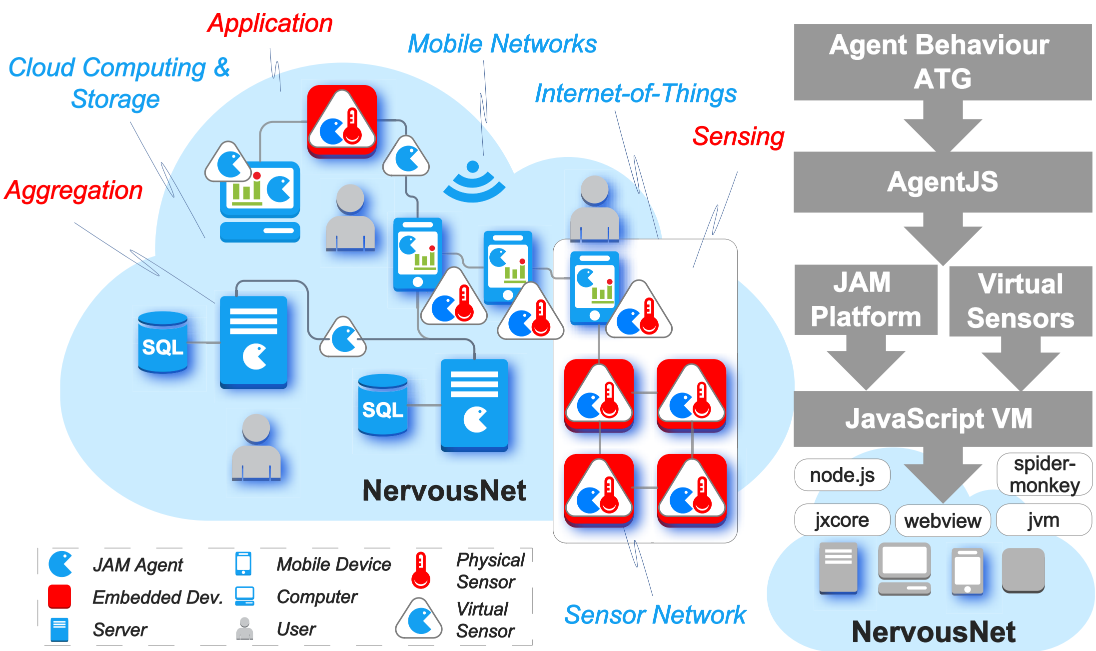


Fig. 14.19 *Unified Mobile Network/IoT/Cloud Distributed Perception and Information Processing with mobile agents using the JavaScript (JS) Agent Machine Platform (JAM) and the Nervousnet Service as the organizational layer composed of virtual sensors, represented with JAM agents.*

Agents operating on a particular node (e.g., a mobile device) can interact and synchronize by using tuple-spaces as a suitable MAS interaction and coordination paradigm in loosely coupled, changing, and self-organizing MAS.

Virtual Sensors and Agents

Large-scale sensing application can be composed of virtual sensors. A virtual sensor is a software component being the core cell of the *Nervousnet* platform. Each software component is treated as a sensor, processing an input stream and computing an output stream. Each physical sensor is a "data stream" transformer, too, but based on physical principles. A virtual sensor is a processing system as well as a data storage (data base). In this work, virtual sensors are represented by mobile agents, performing the sensing, aggregation, and application (or delivery) of accumulated and processed sensor data. As discussed in the next section, these agents are highly portable and can be executed by a wide range of devices including smart phones. The mobility enables self-organizing and adaptive mining systems controlled by environmental constraints rather than by individual users. In [MUS16], users using a smart phone App. are considered as agents. This role is replaced in this work by the deployment of agents that perform tasks autonomously.

The agents interact with each other by accessing tuple spaces or by exchanging signals. The advantage of tuple spaces and mobile agents is the generative nature. A sensor data tuple can be stored in an environment without physical sensors by mobile agents, enabling the access of remote sensor data by other agents. In the original *Nervousnet* platform, mobile Apps. deliver sensor data to the *Nervousnet* data bases, and access control is performed by the *Nervousnet* platform. The autonomy of agents and the anonymous nature of tuples introduce privacy issues, which require dedicated privacy control mechanisms. Although data encryption can be used to protect sensor data, a privacy protection layer applied to sensor data without encryption stripping private device and user data can be considered as a more powerful and useful technique. One private information still exists: The location of agents and the sensor data they collect from devices, which can be easily recognized by mobile agents applying path tracing and other relative localization methods. Therefore, agents require encrypted keys to access personal and sensitive sensor data on mobile devices, granted by the user or trusted platform.

The principle *Nervousnet* architecture composed of virtual sensors and the JAM agent relationship are shown in Figure 14.20. The architecture is hierarchical and can be extended easily with additional virtual sensors. Virtual sensors are deployed in mobile, traffic, and ubiquitous environments. Upper-level virtual sensors provide storage, sharing, and analytic services. Virtual sensors and virtual sensor agents represent the sensing, aggregation, and applications layers of crowd sensing systems.

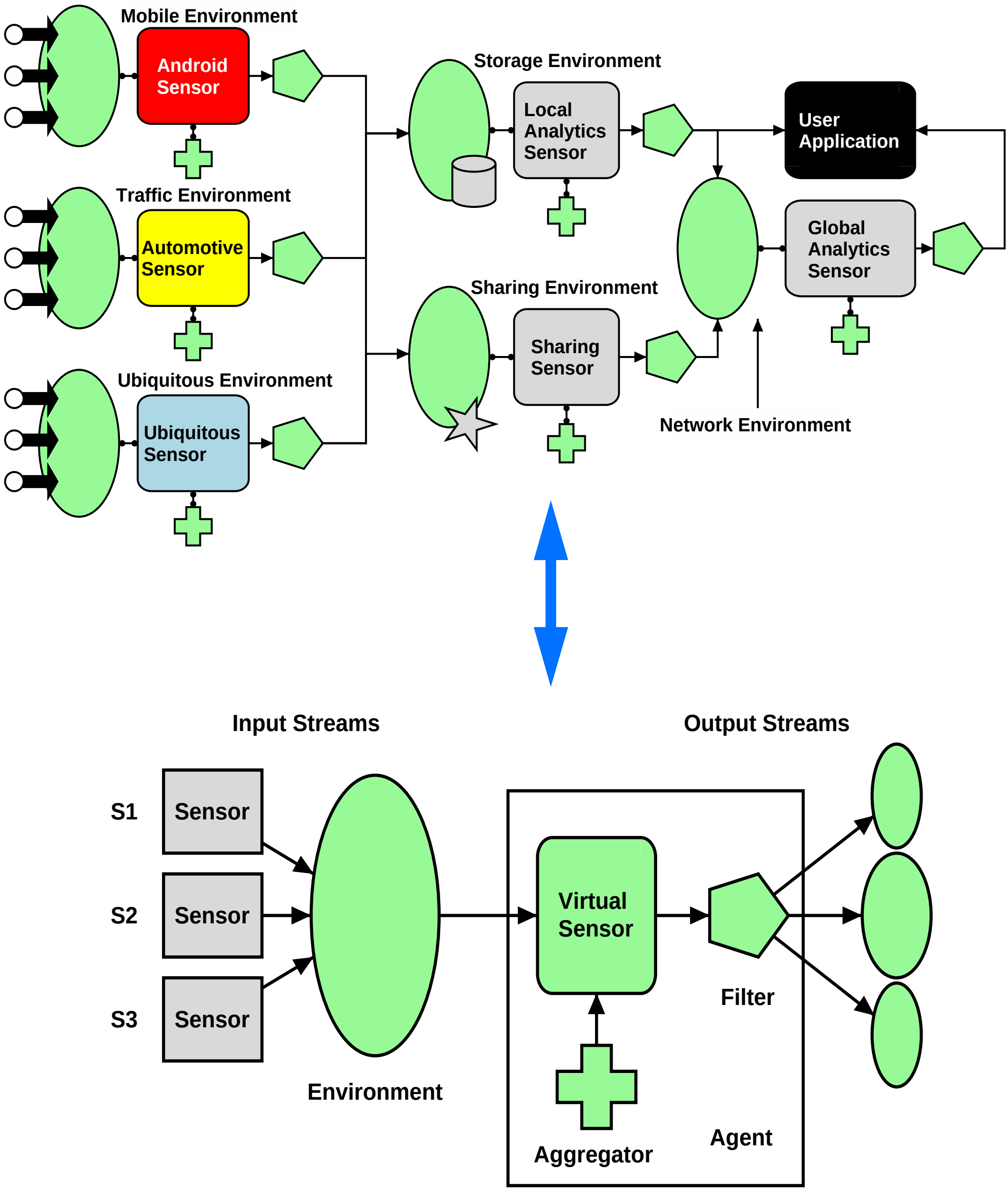


Fig. 14.20 *(Top) Hierarchical sensor aggregation by virtual sensors (Nervousnet architecture* [POU15]*) that can be implemented by JAM agents* [BOS17A] *(Bottom)*

Crowd sensing applications are mainly operating in public environments. There is not always Internet connectivity via mobile networks. For this reason, a beacon network, e.g., established via *Bluetooth*, can be used to connect mobile devices. Since beacons can have Internet connectivity or not, mobile *JAM* agents can be used to transport data between mobile devices and the Internet, shown in Figure 14.21 (piggyback approach).

A mobile device entering the communication range of a beacon can send out exploration agents that collect and deliver sensor data, finally carried to another beacon area. Data can be exchanged between different agents via tuple spaces.

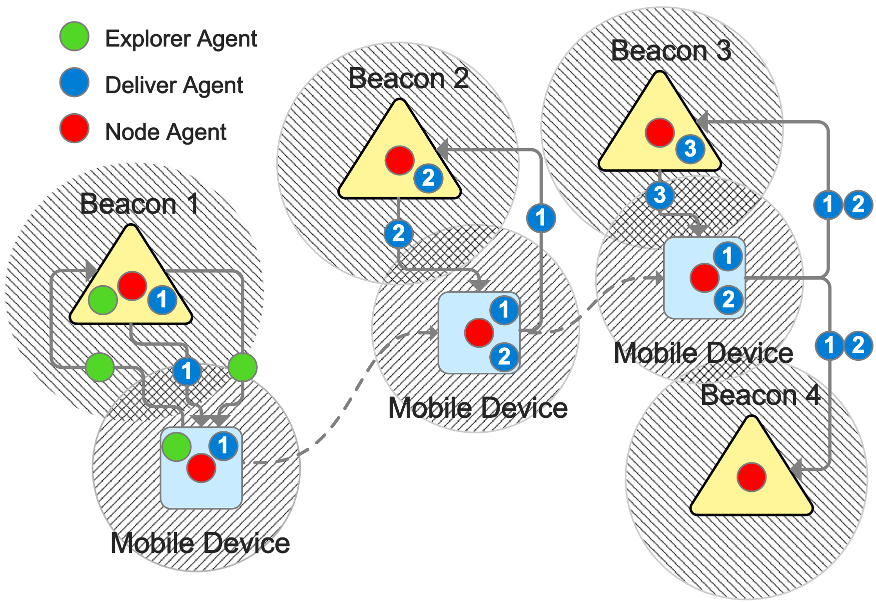


Fig. 14.21 *Principle network topology with spatially distributed beacons (non-mobile) and mobile devices, the MAS and the agent-node interactions.*

14.8 Further Reading

1. S. Poslad, *Ubiquitous Computing: Smart Devices, Environments and Interactions*. 2009, Wiley, ISBN 9780470035603
2. S. Bosse, D. Lehmhus, W. Lang, and M. Busse, Eds., *Material-Integrated Intelligent Systems - Technology and Applications*, 1. ed. Wiley VCH, 2018, ISBN 9783527336067
3. M. J. McGrath and C. N. Scanaill, *Sensor Technologies Healthcare, Wellness, and Environmental Applications*, Apress Open, 2014, ISBN 9781430260134
4. D. Balageas, C. Fritzen, and A. Güemes, *Structural health monitoring*, ISTE, 2006, ISBN 9781905209019
5. Y. Xu, W. J. Li, and K. K. C. Lee, *Intelligent Wearable Interface*, Wiley, 2008, ISBN 9780470179277
6. C. ian Borcea, M. Talasila, and R. Curtmola, *Mobile Crowdsensing*, CRC Press, 2017, ISBN 9781498738446

epubli, ISBN 9783746752228 (2018)