Chapter 3
Neural Networks in Intelligent Sensors and Measurement Systems for Industrial Applications

Stefano FERRARI, Vincenzo PIURI
Department of Information Technologies, University of Milan
via Bramante 65, 26013 Crema, Italy

Abstract. This chapter discusses the basic concepts of intelligent instrumentation and measurement systems based on the use of neural networks. The concept of intelligent measurement is introduced as a preliminary step in industrial applications to extract information concerning the monitored or controlled system or plant as well as the surrounding environment. Implementation of intelligent measurement systems encompassing neural components is tackled, by providing a comprehensive approach to optimum system design. Issues and examples concerning the use of neural networks in intelligent sensing and measurement systems are discussed. The main objective is to show the feasibility and the usability of these techniques to implement a wide variety of adaptive sensors as well as to create high-level sensing systems able to extract abstract measures from physical data, with special emphasis on industrial applications.

3.1. Introduction to intelligent measurement systems for industrial applications

The conventional sensors, instrumentation, and measurement systems are based on dedicated components with some tunable parameters which allows for appropriate calibration and, possibly, for some adaptation to the operating conditions. Some flexibility of the physical architecture is provided in virtual instrumentation [1] by adopting a microprocessor-based structure in which the measurement procedure is defined in the algorithms executed by the microprocessors. However, these solutions have a rather limited "intelligence", i.e., a limited ability of extracting knowledge from the real world to define and modify their own behavior. In particular, they are not able to understand and learn the desired behavior from the observation and analysis of sufficient examples of such a behavior; besides, they are not able to dynamically adapt their own behavior to changing operating conditions and requirements.

The use of neural networks as design and implementation technique allows – in several cases of practical interest – for achieving this flexibility and adaptability. Neural networks have in fact been shown effective to tackle several cases in which an algorithmic description of the computation to produce the desired outputs is either difficult to identify or is too complex, while it is rather easy to collect examples of the desired system behavior [2-7]. This is valid also to implement advanced sensors that process basic physical quantities to extract high-level information, possibly mimicking biological systems, to create adaptable and evolvable instrumentation having high accuracy and low uncertainty,

and to realize measurement systems that are able to create comprehensive views of the monitored system by intelligent sensor fusion and adaptation [8]. For an introduction to the neural computation, refer to [2-7]: in the sequel of the book, the reader is assumed to be rather familiar with the basic concepts of neural networks.

In Section 3.2 the design issues, technologies and problems are discussed to provide a comprehensive view of the interacting goals and characteristics that need to be carefully balanced for an optimum implementation of an intelligent measurement system. Hardware and software solutions are presented. A comprehensive design methodology is then introduced. In Section 3.3 the practical use of the neural paradigms is discussed in several application cases for intelligent sensors and measurement systems, as a fundamental basis for any industrial applications. Approaches available in the literature are analyzed to show the effectiveness and the efficiency of the neural-based approaches for the given application constraints.

3.2. Design and implementation of neural-based systems for industrial applications

To introduce adaptivity in measurement systems and industrial applications, neural networks were widely experimented, especially when sufficient examples of the expected behavior were available or created at a reasonable cost. A huge number of successful results and cases were reported in the literature, as well as in many other cases neural networks were proved to be not so effective and efficient.

The key for the success with these technologies is the use of a comprehensive and structured design methodology. This methodology should encompass not only the analysis of the desired system behavior, but also the understanding of all application constraints and their incorporation within the overall design process in order to identify the most suited solution in the whole space of the possible ones [9,10]. In particular, ability of strict real-time operation is essential in many industrial applications to deal with the fast evolving application system and environment. Accuracy and uncertainty of the outputs are important in many practical applications, e.g., in monitoring and control systems whenever critical decision must be taken and a smooth behavior of the system is desirable for wearing, economical, or safety reasons; this is the case of many industrial and environmental applications. Economical cost may be critical in mass production applications and when profit margin is rather small. Volume and power consumption may become relevant whenever portability of the application system is vital, e.g., in embedded systems for telecommunication. In several cases, these constraints set conflicting goals for the design process: the final solution needs therefore to balance them in a satisfactory way, possibly according to priorities defined by the designer.

3.2.1 Design of the neural paradigm

Any design methodology has to identify the neural solution that best tackles the specific application problem and satisfies the application constraints. In the literature many neural networks were shown effective in various applications [2-7], ranging from feed-forward multi-layered perceptrons to feed-back networks, from self-organizing maps to radial basis functions, and much many.

The identification of the most suited network is therefore the first complex task for the designer. From an abstract point of view this problem could be tackled by describing the neural computation as a network of processing elements (neurons). Each neuron generates its output by applying a non-linear function to the summation of its inputs. A neuron is

connected to all other neurons by weighted links through which its outputs are presented as inputs to the receiving neurons; inputs from the external environment are delivered to all neurons. Memory elements are introduced at the neuron's inputs to allow for memorizing the dynamic behavior of the system. The neural computation is therefore parametric in the number of neurons, the memory elements, the non-linear functions, and the interconnection weights. The neural computation is expected to approximate as best as possible the desired (static or dynamic) behavior described by a set of examples. This view allows for defining a mathematical approach to the identification of the optimum neural computation that solves the envisioned application: the problem could be in fact stated as a functional. The solution of the functional is the best neural computation for the given application problem. Constraints on the system characteristics can be defined so that solution of the functional will be constrained. Unfortunately, this approach is not practically feasible since the optimization space is too huge: the exploration will take an unacceptably long time.

The neural computation needs therefore to be defined in a more efficient way through a sequence of steps that explore the alternatives by exploiting the available knowledge cumulated by researchers and practitioners around the world along the past twenty years. To achieve this goal we start from the desired behavior, as defined by the available examples, and the application constraints (e.g., concerning accuracy, uncertainty, power consumption, economical cost, etc.).

First of all, the most appropriate neural paradigm must be identified among the wide spectrum of neural families proposed in the literature. In particular, the overall topology of the network and the internal structure of the neurons must be selected. In the case different alternatives have been shown effective in cases similar to the envisioned application, all of them should be explored in the subsequent steps to finally achieve the most suited solution. Selection is in fact usually not immediately feasible at this initial design stage since detailed characteristics and constraints need to be taken into account; besides, an accurate evaluation of the performance can be done only when the actual implementation has been selected. For example, feed-forward neural structures can be adopted in all applications in which a mathematical function needs to be approximated or for classification when input-output examples are available. Feedback networks are appropriate for modeling dynamic behaviors, e.g., in control applications, by using a feed-forward structure with a feedback loop which supplies the past history to the network inputs through memory elements. Self-organizing maps are effective for classification when classes are not a-priori defined. The sigmoid function to generate the neuron's output is one of the widely used in theoretical research; in the practice approximated versions outperform the theoretical sigmoid as computation power is concerned.

Second, the most appropriate network model must be chosen within the selected family by defining the structural characteristics of the model. Namely, we need to identify the number of neurons in the network and, in the case of dynamic systems, the length of memory history. Experience can be useful to make these selections. A theoretical framework should consider the complexity of the application problem as defined by the set of examples that characterize the desired behavior. In the literature, some methodological guidelines have been presented to dimension the network [11,12], also by taking into account the quantity and the distribution of examples over the field of the desired behavior. In general, the typical approach is based on tentative cases having different network size and on the analysis of the accuracy achieved in their outputs: from the literature a promising range is foresee, then experiments will lead to subsequent refinements by focusing the attention on the most attracting sub-ranges till the probable optimum structure. Similarly, we should operate to identify the number of memory elements required to hold the system history. It is important to point out that the trial-and-error approach that is used to configure

completely the network requires to evaluate the accuracy of the outputs and the other characteristics of the model (e.g., the generalization ability). Consequently, the optimum dimension of the neural network depends on the optimum configuration of the network weights that is achieved at the end of the configuration procedure for the envisioned network structure. To break this loop we need therefore to adopt an iterative approach: we have to complete the configuration by assuming that the network under consideration has the optimum size and, then, go back to evaluate if such network was actually optimum.

The third step consists of configuring the neural network interconnection weights by learning the desired behavior either by a supervised or an unsupervised training procedure. Many techniques were developed in the literature for the different neural models [2-7]. For example, several variations of the back-propagation algorithm were experimented for the feed-forward networks. Extensions for feedback network were also studied. Self-adaptation was proposed for self-organizing maps. Selection of the most suited learning approach can be performed by searching in the best results presented in the literature for the envisioned model family and application. Learning must be configured to take into account the actual characteristics of the implementation that will be adopted. For example, possible approximations of the theoretical non-linear functions, that are adopted to achieve a better implementation (e.g., from the point of view of the circuit complexity and power consumption in the case dedicated hardware solutions, or the computation complexity in the case of software realizations), must be considered also in training to create a consistent solution. Large network errors and even convergence problems in dynamic systems may be in fact induced in the application system during in the operating life by having trained the neural model with ideal conditions and, then, by having applied the approximations. This is the typical case that occurs when training is performed by using a theoretical sigmoid, while a multi-step function is adopted in the real system.

In the fourth step, the training procedure is applied to configure the operational parameters of the network model. Two basic issues must be carefully considered since they greatly affect the quality of the network and, consequently, the accuracy of the outputs: which data should be used for training and how long learning should be continued. In many real applications the examples of the desired behavior are available only in a limited quantity. Often it may be not easy or cheap to collect these examples for different reasons: for example, in some cases running the physical experiments to collect the data may be economically expensive, sometimes there is no personnel available enough to do the tests, in other cases the production cannot be suspended to perform experimental runs, and some operating conditions may be difficult to apply. When a limited set of data is available, it must be split in two parts: one to actually perform training, the second to validate the training result (i.e., the characteristics of the network such as the generalization ability, the robustness, and the accuracy). The validation data should never be used for training in order to have an impartial evaluation; using training data for validation will result in an optimistic – sometimes too much – evaluation of the network abilities. However, the less training data are collected, the lower is the quality of training and the higher is the network error in generating the desired outputs. Some additional guidelines can be found in the literature to deal with these issues and to evaluate the related network accuracy, e.g., see [13]. Duration of training is critical as well. In fact, if learning is too prolonged the network tends to learn the examples too much and to loose the generalization ability. Training should be applied till the network error decreases when test examples are presented: when the error becomes steady, training should be terminated. In the case of periodic or continuous learning, the procedure and the network configuration update must be controlled so as to allow for a high generalization ability and accuracy. By analyzing the neural model and the validation data, we can derive also the confidence that we can have on the computation outputs [14].

More detailed guidelines to create the neural paradigms can be found in the following chapters with specific reference to the envisioned specifications and application areas.

After the previous steps, we obtain a configured neural paradigm that is able to solve the envisioned application problem, possibly with the desired accuracy and uncertainty. It is worth nothing that the configured neural network *is* an algorithm, since it defines exactly sequence of all operations and all operand values required to generate the network outputs from the current input data. When configured, the computation of each neuron is in fact a weighted summation followed by a non-linear function, while the topology of the neural network defines the activation order of the neurons' computation and the data flow. The difference of neural paradigms with conventional algorithmic approaches consists of the fact that the algorithm designer has to define the sequence of operation to solve the application problem, while the neural designer has only to select the computational model and learning identifies the exact sequence of operations from the behavior examples.

In several application cases, neural solutions have been shown superior to algorithmic approaches, when the design and environmental conditions discussed at the beginning of this section apply. In many other cases efficiency and accuracy of algorithms remain outstanding. However, there are several cases in which a suited combination of the characteristics and properties of both of these computational approaches may lead to more advanced solutions. The efficiency of algorithms to tackle specific tasks for which they are known and effective can be in fact merged with the adaptivity and the generalization ability from examples of the neural paradigms. This results in the composite systems [9]. In composite systems the computation is partitioned in algorithmic and neural components to exploit the best features of each of these approaches. From the high-level functional description of the application and the related constraints it is therefore necessary to perform appropriate analysis of the desired behavior to partition the application system and to derive the high-level description of each algorithmic and neural component. Then, learning allows for configuring each neural component so as to create its final algorithmic description. The resulting high-level description of the whole system consists thus of the collection of the algorithmic description of all components, independently from the way in which the designer initially described each of them.

3.2.2 Design of the neural implementation

The second complex task for the designer is now the identification of the most suited solution for implementing the neural computation (or the composite system) that has been reduced to an algorithmic description for the envisioned application and with the given constraints. Several approaches have been presented in the literature, with different performance, cost, power consumption, and accuracy.

Several proposals were made in the literature by using analog hardware (e.g., [15-19]). Analog integrated circuits for neural computation are based on the fundamental laws of electric circuits: the Kirchhoff's and Ohm's laws. According to the Ohm's law, the voltage across an electric dipole is proportional to the current flowing through it. A linear dipole can represent a neural synapses: the voltage across the dipole represents a neuron input and the proportionality constant the related interconnection weight; the current flowing through the dipole is the weighted input. According to the Kirchhoff's current law, the total current entering a circuit node is null (currents exiting the node are accounted as negative terms). If the negative poles of the dipoles associated to a neuron are grounded together, the weighted summation of the neuron's inputs is the total current flowing to the ground. Similar results can be achieved by using other circuit topologies and devices (e.g., operational amplifiers and transistors). The use of analog circuit for neural computation is very effective since

computation is performed at a very high speed (i.e., the speed allowed by the propagation and stabilization of the electric signals), the dimension of the circuit is very small, and all neural signals are represented by continuous values (thus allowing for theoretically representing very accurate values). However, there are two main drawbacks that greatly limit the practical usability of this approach. First, the configuration of the neural system is fixed at production time; consequently, the interconnection weights cannot be changed at power up and a specific circuit needs to be fabricated for each application case. Second, fabrication inaccuracies that are typical of any production process make impossible to guarantee a good accuracy of the characteristic parameters of the devices and, consequently, the accuracy of the neural interconnection weights. This approach should be adopted only if the overall network behavior is highly robust with respect to the variation of the network parameters.

Analog hardware with digital weights can be adopted to achieve some configurability of the interconnection weight (e.g., [20,21]). In this case a mixed-mode multiplier computes the input weighting. The multiplier (i.e., the weight) is given in the binary representation. Multiplication is performed in parallel on each multiplier digit by using dedicated circuitries; the analog multiplicand is presented in parallel to each of these single-digit multipliers. Each binary digit of the multiplier controls the flow of the current through the corresponding single-digit multiplier: no current will be generated if the control digit is zero; otherwise a current proportional to the binary weight of the digit is generated. The multiplication result is obtained by adding all currents generated by the single-digit multipliers according to the Kirchhoff's current law. Performance of this approach is still very high and control of the accuracy of characteristic device parameters is limited. Interconnection weights are discretized since they are given in the binary representation; this influences the accuracy of the final outputs. The network dimensions and topology as well as the neuron's operation are fixed at production time, thus limiting the circuit flexibility. The circuit size is larger than the pure analog approach since the mixed-mode multipliers are more complex.

Complete control of the accuracy can be achieved by adopting digital dedicated hardware architectures (e.g., [22-26]): all data are discretized and given in binary representation and all operations are performed digitally. Interconnection weights are configurable, but the network topology and size as well as the neuron's behavior are still fixed at production time. Performance is much lower than in the corresponding analog implementations due to the nature and the realization of the digital operations, but still it is rather high. The circuit complexity becomes relevant and, consequently, the integrated circuit becomes rather large. To limit the size and allow for fabrication, several neural operators often share in time some components, by introducing suited registers and clocking schemes; for example, one digital multiplier can be multiplexed among all interconnection weights of a neuron or the same circuit can compute the operations of several neurons sequentially. These architectures may have a limited circuit complexity for some classes of neural networks, e.g., when the neuron output is a single-digit binary value. The data discretization limits the accuracy, although it is exactly predictable.

The use of configurable digital hardware allows for high configurability (e.g., [27-29]). The typical approach consists of implementing the neural networks on an FPGA: all operations are mapped onto the logic blocks and interconnection paths of the FPGA. The high-level description of the neural operation (e.g., written in C, SystemC, or VHDL languages) is translated into the corresponding FPGA configuration that will be loaded on memory-based architectures or will be used to set the operations and interconnections in fuse-based architectures. Any neural topology and size and any neuron operation can be accommodated in the FPGA, provided that sufficient logic blocks and interconnections are

available and that an appropriate operation schedule is adopted. Performance is lower than the dedicated digital architecture since basic neural operations involve more and slower physical components. Accuracy is influenced by the discretized operands.

Programmable digital architectures provide the highest configurability since the neural operations are described in suited programs. Since the computation is known, the accuracy can be evaluated; also in this case accuracy is influenced by the discretized operands.

Neurocomputers were developed to perform the neural computation in an efficient way by preserving the system flexibility (e.g., [30-32]). The behavior of these architectures is similar to the one of a conventional computer: the architecture consists of a memory in which the sequences of specialized operations that describe the neural computation are stored, and processing units that are able to fetch, decode, and execute these sequences stored in the memory. To achieve high performance these architectures make use of dedicated functional units to execute the operations that are the most frequent in the neural computations, and efficient interconnection structures to distribute the neurons' outputs to the receiving neurons. The specialized functional units may be implemented in FPGA to ensure additional flexibility. Any neural network can therefore be implemented by this kind of architectures, provided that the instructions executable by the processing units are able to describe the desired neural behavior.

All of the above solutions suffer from the same problem: the more the architecture is dedicated, the more expensive it becomes since it cannot be mass-produced and reused in a large number of instances and different applications. To overcome this drawback, non-specialized processors should be adopted so that they can be directly purchased on the market as components off the shelf.

In this perspective, digital signal processors (DSP) are an attractive solution that combines reasonably high performance with programmability (e.g., [33-35]). These processors have an architecture that usually includes supports and functional units specialized for the most frequent signal processing operations, e.g., convolution and correlation. Since the weighted summation coincides with these operations, it can be efficiently executed on DSP processors available on the market. The neural computation is obtained by executing dedicated software written for the selected DSP processor. This approach needs anyway to use processors, boards, software development environments, and programming skills that are less available – and thus more expensive – than for the widely-used general-purpose processing architectures.

General-purpose processors are the most flexible computing structures for which many programmers have sufficient knowledge and expertise to produce good programs. Processors for personal computers are among these structures. For these architectures dedicated software can be written in high-level programming languages to perform any neural computation. Performance is lower than in DSP architectures with similar characteristics since the efficient dedicated supports for DSP operations are not available in general-purpose systems. To speed up the performance general-purpose supercomputers can be used, e.g., [36-38].

To reduce the development costs due to the need of experienced programmers and to widen the use of neural computation also among practitioners with limited programming experience, general-purpose architectures with configurable software simulators can be adopted (e.g., [39]). In these software simulators, through a graphical interface, the designer can build the neural paradigm to tackle his application; typically he can select – in a predefined but usually very large set – the desired family of neural networks, the specific network dimension, and the appropriate weight configuration. In some simulators the designer is even allowed for creating his own network model. Performance is usually limited since configurability is obtained by interpreting the neural computation, thus

leading to a slow execution. Some of these simulators are however able to produce a compiled version of the neural computation so as to greatly speed it up with respect to the interpreted version.

Dedicated software or neural network simulators are also needed to support learning. In any of these cases the network model adopted for learning must be identical to the one that will be used in the operating life. In particular, great care is necessary in verifying that all network characteristics, the precision of the data representation, the accuracy both of each operation and of the sequences of operations, all data uncertainties are identical in order to guarantee that the learnt behavior coincides with the one shown during the operational life of the neural network.

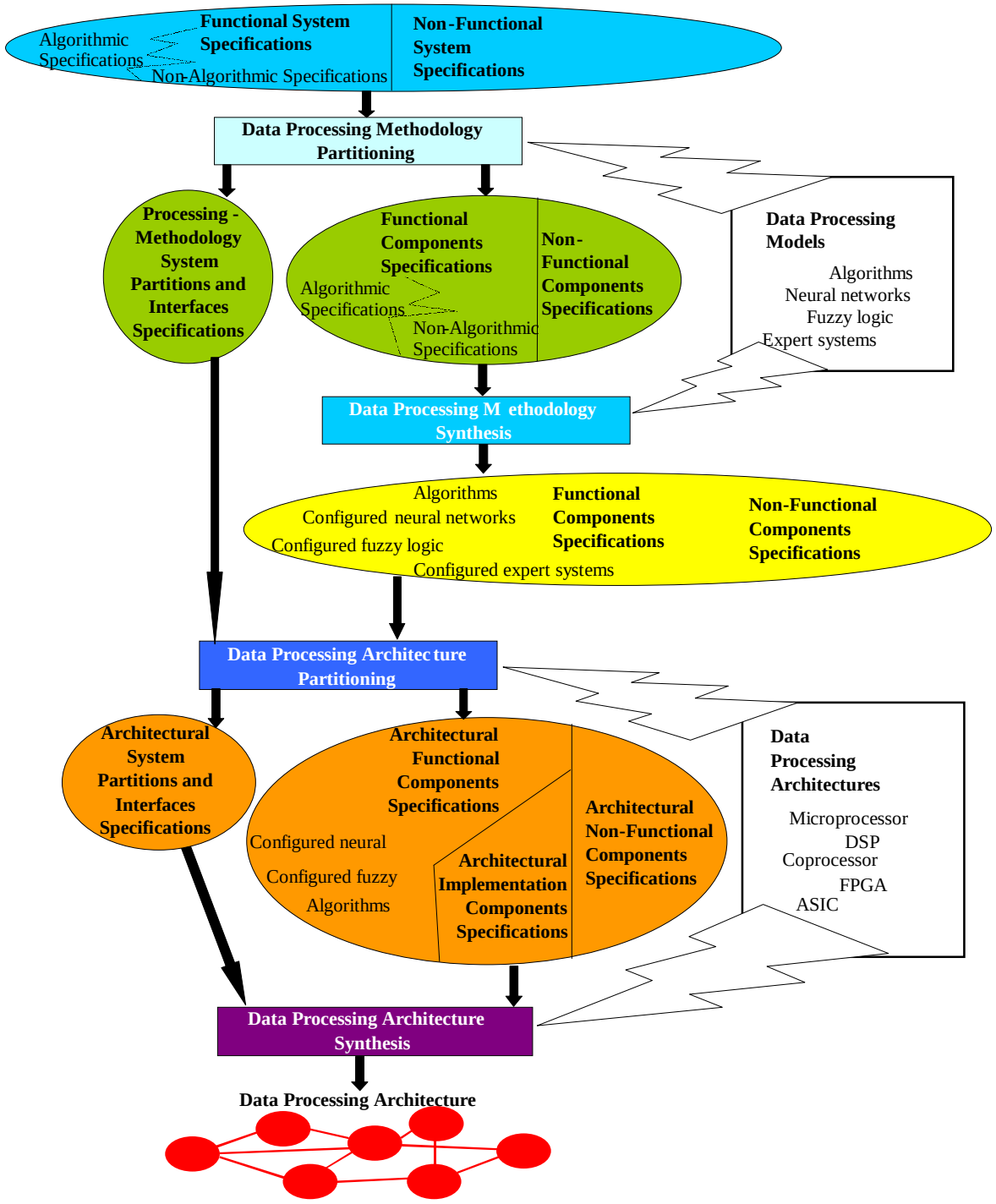

Figure 1: A comprehensive design methodology for composite systems.

3.2.3 A comprehensive design methodology for composite systems

To implement adaptive approaches in measurement systems and industrial applications a compehensive methodology is necessary to specify all issues discussed above at a high abstract level and to synthesize an optimal composite structure, according to a multi-objective optimization function. System-level design techniques (originally proposed for DSP and telecommunication applications based on algorithmic approaches [40]) have been extended (e.g., [9]) to deal also with soft-computing paradigms. This implies to consider two orthogonal perspectives into a homogeneous view with all non-functional and implementation constraints: the algorithmic/soft-computing synthesis and the conventional hardware/software synthesis. The resulting methodology is summarized in Fig. 1.

The first phase of the high-level design methodology consists of the system specification. The functional characteristics define the system behavior. High-level formal specifications are widely used, e.g., by means of the sequencing graphs [41]. For static digital systems, the combinatorial function that generates the expected output for each input is given; in dynamic digital systems, the state diagram relates each pair of input and system state to the output and the next state. Analog models typically describe plants and industrial processes by means of differential equations, often continuous-valued and possibly at the partial derivatives. Data are traditionally represented and processed as crisp values; fuzzy values generalize the data representation when the envisioned characteristic is a deterministic collection of crisp values. Fuzzy rules algorithmically define how the desired outputs must be generated. Expert systems use rules to explore the space of possible solutions. Neural networks are defined by examples by means of the training set: in static networks the input-output pairs for supervised learning or the input set for unsupervised training describe the desired behavior; the evolution of the system state is captured by means of the ordered sequences of the input-output pairs in dynamic networks. To identify the optimum solution for the envisioned application, the design methodology should consider – as early as possible – also all non-functional specifications, e.g., accuracy, uncertainty, performance, real-time operation, throughput, operation complexity, circuit complexity, and power consumption.

The second design phase consists of partitioning the system in components described by different computational paradigms (i.e., into algorithmic and soft computing components), by taking into account also the non-functional constraints. Some algorithmic and soft computing components can be functionally equivalent, even if their expressiveness, completeness, conciseness, and non-functional specifications may be different. The model to be selected is the one that best balances – not necessarily optimizes – the application requirements: the model chosen for a component greatly impacts on the implementation characteristics, e.g., complexity, performance, and power consumption. Computational paradigm partitioning identifies boundaries among components and the related interfaces so that each of these components is efficiently implemented. Natural and evident boundaries are first taken into account as defined by the designer's specifications. Partitioning is then guided by suited quality measurement to split components into simpler subsystems that can be efficiently represented by one model. Aggregation and separation techniques are used to resize components and to group the homogeneous ones in the perspective of the implementation.

The third design phase is the computational paradigm synthesis, which consists of configuring each component and the related interfaces. For algorithmic components the procedure describing the desired computation is derived. For soft computing components the corresponding synthesis is performed. For neural models the learning procedure is applied: this produces the algorithmic description of the network operation. For statistical

model, the parameters are identified on the available data by statistic techniques. At the end of the paradigm synthesis, all components are described by algorithms.

The fourth design phase is the hardware/software partitioning that splits the algorithmic specification of the system into components to be implemented in dedicated analog, digital, or mixed hardware devices, in configurable hardware components, or in software programs running on DSP or general-purpose processors. This can be obtained by using one of the many hardware-software co-design techniques proposed in the literature and widely available in commercial CAD tools. Partitioning is guided by the non-functional specifications. It is worth noting that hardware/software partitioning is independent from computational paradigm partitioning. At the end of this phase the processing system architecture and the detailed structure of each component are obtained.

The fifth design phase is the synthesis of the processing architecture. This can be achieved by means of the traditional techniques for system synthesis: programming of the software components and digital/analog synthesis of the hardware devices (e.g., [42]).

3.3. Application of neural techniques for intelligent sensors and measurement systems

Neural techniques were shown effective and efficient in enhancing the characteristics of sensors and measurement systems as well as in industrial applications. In the literature many perspectives were presented to introduce “intelligence” in these systems by means of neural networks:

- sensor enhancement allows for creating devices which are able to physically sense quantities for advanced applications,
- sensor linearization simplifies the use of sensors in measurement systems and applications by providing an idealized view of the sensor,
- sensor fusion merges information from several sensors, possibly of different type, to create new combined measurements,
- sensor diagnosis verifies the correct operation of the sensor and detect the possible presence of errors due to faults,
- virtual sensors indirectly observe quantities for which no specific sensor is available by using information about quantities related to the desired one,
- remote sensing allows for indirectly measuring physical quantities without using a sensor that physically enters in contact with the measurand quantity,
- high-level sensors measure abstract quantities (i.e., not directly related to physical quantities) which are of interest for the applications,
- distributed intelligent sensing systems create a cooperative collection of sensors that provides a comprehensive view of the system under measure,
- calibration allows for correctly relating the measured values performed by sensors and measurement systems to the physical values of the quantities under measurement.

3.3.1 Sensor enhancement

The physical sensing materials have usually complex non-linear behaviors that need to be related to the corresponding values of the measured quantities. In particular, some physical characteristics of the sensing material when operating in physical contact with the measurand quantity vary according to the physical laws that regulate the interaction between the system under measurement and the measurement system. The varying physical quantity of the sensing material that best represent the quantity under measurement is assumed as the output of the sensor: this value is associated to the measurand quantity.

Neural networks can be used to create advanced sensors by suited processing the physical outputs of the sensing materials to extract the measurement of the desired physical quantity, especially when conventional processing techniques have been shown inaccurate or with not sufficient adaptivity. In some cases, neural approaches are also useful to enhance the accuracy of the measurement procedure so as to enhance the quality of the delivered measurements. Among sensors that benefit from neural technologies the literature reports sensors that reproduce the five human senses, as well as sensors for many other environmental and industrial quantities like mechanical quantities (e.g., distance, force, pressure), thermical quantities (e.g., temperature), and chemical quantities (e.g., concentration, presence of substances).

Image sensors are the basic step to reproduce the natural sight. Conventional digital cameras have image sensors, composed by a grid of sensible materials, that are able to capture the light (intensity and color); in each pixel information is transformed into a digital representation. Advanced image sensors mimic the behavior of the natural photoreceptors (the elementary components of the retina) in the human eyes, to allow for capturing images in a more "intelligent" and flexible way [43-45]. Human photoreceptors in fact have self-adaptive abilities to deal with light intensity and color saturation in order to create high-quality images even in the presence of adverse environmental conditions. Besides the image characteristics are represented in an impulsive way for subsequent processing by the brain. The artificial photodetector is obtained by using groups of photodiodes, which are sensible to various wavelengths, and interconnection circuits, that provide lateral connections and information processing among neighboring cells. When the photodetector is hit by light within its sensitivity range, it generate impulses proportional to the light intensity; impulses are then filtered by taking into account the events occurred in time and in the areas nearby. Neural networks are used to implement the non-linear lateral cooperation. This approach may have several benefits, including less saturation, reduced calibration, higher quality, higher accuracy, higher time sensitivity, and less power consumption.

By using either the intelligent photodetectors or conventional cameras with suited post-processing, an artificial retina can be implemented, whose behavior is similar to the human one [46-48], to provide a prosthesis to overcome blindness and severe visual imparities when the optical nerves and the optical brain functions are still in good conditions and operational. At the moment, the complexity of data processing required to generate appropriate signals for the brain is too high to be compacted into a small integrated circuit; besides, power consumption and power supply are still a relevant problem that needs external batteries and frequent recharges. These constraints prevent – nowadays – to realize prosthesis for permanent implantation in the human body instead of the natural retina. However, the feasibility of the approach was demonstrated by using stimulating devices implanted on the optical nerves and a processing system out of the human body: a prototype system was even recently implanted on a patient with interesting – although still low quality – results. The image taken from the image sensor array is transformed into an impulse-based representation suited for stimulating the optical nerves, also by using a neural-based approach. The image representation is then coded and transmitted wireless to the receiver implanted in the human body. Received data are decoded and delivered to the optical nerve stimulators. The image is thus transferred from the artificial eye to the brain for the usual processing and understanding.

At a higher abstraction level, visual sensors analyze an image or a sequence of images to detect and understand the objects contained in the images themselves and, eventually, to observe objects' motion [49-53]. This function mimics the image understanding activity of the natural brain. Objects are identified by extracting characteristic features from the image and by comparing the combinations these features with those of the classes of objects to be

recognized: an object is identified when its features are similar to those of one of such classes. Motion is detected and analyzed by observing the variations of the features in the images of the sequence. Neural networks were shown effective for these adaptive tasks, which have many practical applications not only in the medical field, but mainly several industrial and robotics areas whenever image analysis and understanding is important.

Hearing sensor and the artificial cochlea can be realized similarly to the sight aids in order to assist people with severe hearing imparities with adaptive personalized prosthesis [54,55]. Conventional hearing aids increase the volume of any acoustic signal (voice, sounds, noise), possibly by filtering out some frequency bands; in general this approach has limited adaptivity to the patient and deliver too much noise that makes the patient uncomfortable. A neural-based approach can outperform the conventional one for the voice: it can understand the speech and synthesize the voice from the basic phonemes. First a microphone captures the voice; then signal (also neural) processing detects the boundaries between words, extracts the phonemes of each word, and identifies the words possibly by using also a vocabulary. The coded words are then wireless transmitted to the implant in the human body, where they are decoded and used to drive the voice reconstruction by cascading the appropriate phonemes. This signal is used to stimulate the auditory nerve as the natural cochlea does.

Odor sensors and the artificial nose were also successfully experimented by using neural solutions [56-59]. The natural nose identifies odors by detecting the presence and the quantity of chemicals in the air. It has receptors that are sensible to some specific classes of chemicals; the brain merges all olfactory information and classifies the odor on the basis of its experience and its knowledge of objects' smell. In the artificial nose, sensing materials react to the presence of some chemical families, possibly different with those of the natural receptors: these reactions are transduced into electric signals. On the basis of the type of active sensing materials (i.e., the detected family of chemicals) and the amount of their activity, the artificial nose classifies the smelled odor. This system was conceived for automatic odor analysis in industrial applications, e.g., in alimentary factories to identify rotting food or to grade the maturation level. Effectiveness of the neural approach is due to the relevant problem non-linearities and the difficulties to give an algorithmic approach.

Similarly, the natural tongue identifies tastes by analyzing the presence and the quantity of chemicals on the object touched by the tongue (the saliva transports the chemicals from the surface of the tasted object to the papillae). In the artificial tongue [57], sensing materials are used to detect some chemical families that are on the surface of the touched objects, as in the artificial nose. Classification leads to identify the taste of the object according to the kind of tastes to which the sensing materials are reacting and to the knowledge used to configure the classifier. Also the artificial tongue is used to mimic the human counterpart, e.g., in alimentary factory to automatically discriminate different types and mix of foods and beverages. It is worth nothing that artificial nose and tongue are based on the same operating principles: the only difference is how the chemicals are brought to the sensing devices (thorough the air in the nose, by contact or through water in the tongue).

Tactile sensors are important for advanced robotics when robotic hands need to take objects carefully (e.g., delicate, deformable, or slippery objects) or when objects must be tactilely recognized. The natural skin contains an array of tactile sensors that are able to observe the tridimensional field of mechanical force due to gripping an object; from the analysis of the field of mechanical force, the brain is able to recognize the shape of the touched surface by comparing the current one to its knowledge. The artificial tactile sensors reproduce the ability of neurally reconstructing the field of forces from the individual information coming from the sensing units: from the analysis of the field of mechanical

force they are able to classify the surface shape, to identify the surface state, and to predict the slippery of the grip [60-63].

By using neural techniques several other advanced sensors were developed to measure mechanical quantities, very well suited for industrial applications. In pressure sensors [64] the neural networks were used to correlate the strongly non-linear output of a barometric cell to the corresponding pressure value, by incorporating the specific characteristics of the cell that vary from one cell to another due to the inaccuracies of the production process. Adaptive distance sensors can be implemented by adopting a sonar- or laser- based system [65,66]. Surface roughness can be deduced by analyzing and intelligent merging several of these measurements taken at a short distance [67]. Velocity and angular velocity can be measured by adaptive analysis of the position of the envisioned object [68,69]. Other quantities reported in the litertaure concern, among the many examples, force [70,71], torque [72,73], and strain [74].

The use of neural networks was proved effective also to implement many other sensors and measurement systems for electromagnetic quantities (e.g., [75]), environmental quantities (e.g., temperature and humidity [76-79]) and chemical and biological quantities (e.g., [80-85]). All these cases have many practical implications, especially in a wide variety of industrial production areas, in the biomedical fields, and in environmental monitoring. The basic goals of the use of neural technologies are the same of the cases presented above: to achieve a better evaluation of the system output, to achieve adaptability of the measurement system, and to describe the desired behavior in an easier way by examples.

3.3.2 Sensor linearization

Linearization of the physical output generated by a sensor is useful for many practical applications in all areas. On the market many cheap sensors become nowadays available: low cost makes them highly desirable to reduce the cost of products and systems. However, these sensors often have non-linear output functions corresponding to linearly changing values of the physical measurand quantity. In these cases the subsequent data processing has to deal with such non-linearity to produce the measured value. For example, thermocouples typically measure the temperature by producing an electric voltage at their outputs; this voltage is non-linearly related to the actual temperature. The temperature value needs to be deduced from a conversion table or function; since the correspondence between the sensor output and the temperature is often quite difficult to be given as a simple function, a look-up table can be adopted. Performing this conversion with the required accuracy a lot of efforts is required either for the possible computational complexity of the complex conversion function or for the large size of the look-up table.

If the sensor output was linear, the conversion will be much easier since it will be a simple multiplication for a constant gain, typical of that sensor. The applications could thus be written in a much simpler way, especially when system control is envisioned.

Unfortunately, the reality cannot be changed to make it ideal. However, it is possible to save the simplified view of a non-linear sensor for the application designers by linearizing the output of the sensor itself, i.e., linearization can be embedded in the sensing system so that non-linearities will remain hidden in it. This will not remove the computational or memory efforts mentioned above since they will remain hidden in the measurement system, but it will allow for a much simpler use of the sensor in the various applications since the non-linear sensor and the linearization procedure will constitute a single system.

Linearization can be pursued by several techniques. As already said, the look-up table is easy to create although may become expensive in terms of memory usage. To save memory

at the expenses of computational complexity, a conversion function can be adopted. When only two samples are available, the linear interpolation identifies the straight line that passes through the samples. When some more samples are available, higher accuracy can be achieved by splitting the data set into adjacent intervals and by looking for the broken line that touches all samples. In both cases higher accuracy and smoothness of the interpolation can be obtained by adopting higher-order interpolating functions like quadric functions and polynomials; the drawback is the higher computational complexity when the linearized outputs must be computed. When several examples are available, regression techniques (namely, linear, quadratic, and polynomial regression), possibly in intervals of the sampled quantity, can be used to linearize the output function: the linear, quadratic, or polynomial function that gives the output correspondence is the one that minimizes the average approximation error for the sampled data.

Neural networks are an alternative approach that allows for constructing the output correspondence for the linearized sensor by learning it from examples [86-88]. After learning the neural network computes the function that minimizes the global error on the whole sensor operating range without the need of using different functions for each interval. In sensor linearization the neural network computes therefore the non-linear mapping from the outputs of the physical sensor to the linearized outputs of the ideal sensor. This is a task that is well suited for neural paradigm; in fact multi-layered perceptrons were proved to be universal approximators of non-linear functions, although efficiency is not guaranteed.

The neural approach presents an additional feature: it is able to deal with discretization of the outputs for the digital representation at the same time of linearization. This minimizes the overall error due both to linearization and discretization since the neural network can take both of them into account during the learning phase.

3.3.3 Sensor fusion

Sensors used to observe the status and the behavior of a physical system collect a mass of data that – as a whole – characterizes the system. However, each sensor produces a partial view of the observed system, according to the specific physical quantity it measures. In real applications decisions are usually taken on the basis of the overall system status and behavior, for example to generate the most suited control signals; an individual perspective is in fact often not sufficient to achieve the desired system behavior. The global view needs to be reconstructed by analyzing information from each sensor into a comprehensive perspective. This can be achieved by sensor fusion, i.e., by merging the data produced by each sensor into higher abstraction information to create a single data stream coming from the set of sensors. A wide literature is available about sensor fusion, the related technologies, and applications by using neural-based approaches, e.g., [89-92].

To measure the same physical quantity in a given region of space, more than one sensor (possibly of different type) can be used. This allows for enhancing the confidence in the measurement. In fact the availability of more samples from different sources can be used to better tackle uncertainty and accuracy. The use of different kind of sensors to measure the same physical quantity is useful to enhance the reliability and the confidence in the measured values by exploiting the sensor diversity, at limited costs.

Sensors for different physical quantities of the same physical region are useful to create a comprehensive view of that region by integrating the information provided by each sensor. This kind of integration is specific for each application and group of sensors considered. In general, the availability of multisensorial information allows for depicting a view of the system that averages and amalgamates the individual contributions, thus

limiting the dominance of each sensor onto the comprehensive view and, consequently, avoiding polarizations and enhancing the overall quality.

Sensor fusion for data integration can be implemented by means of a single merging procedure that computes all refined and combined outputs depicting the comprehensive view. This is efficient when the interdependencies among the measured quantities are numerous and each of them involves most of the measured quantities. Alternatively, individual merging procedures can be adopted to refine each measurement by taking into account the information provided by the other sensors. This is suited when interdependencies involve a limited number of different physical quantities for each measurement to be refined.

3.3.4 Sensor diagnosis

Various causes (e.g., aging) may lead to measure drifts in sensors. Sensor fusion, e.g., by neural networks and the continuous comparison among the samples taken by various identical sensors about at the same time can be used to early detect this phenomenon and, eventually, to mask its effects by correcting – as much as possible – the wrong measures before recalibrating the drifted sensor itself [93-98]. The correct measure is the one on which most of the sensors agree, within a given tolerance range of values (this interval is due to the uncertainty in the measurement: comparison needs to be considered positive not only if the measured values are identical, but also if their uncertainty intervals overlap). When a sensor is identified as not sufficiently "reliable" due to drifting, its measurements can be ignored and decisions about the subsequent measured values taken only on the basis of the responses of the remaining reliable sensors. This is especially useful when maintenance and recalibration are difficult, or expensive, or even impossible, or cannot be performed too frequently.

Similarly, normal wear and accidents in the operating environment may induce faults into a sensor, i.e., may change the physical structure either of the whole sensor or one or more of its parts. Some of these faults do not affect the normal operation of the sensor, which continues to deliver correct outputs, i.e., to produce measures that coincide with the actual value of the quantity under measurement. Other faults may appear as erroneous values delivered by the sensor, i.e., different from the outputs that such a sensor would have delivered in the absence of the fault. Sensor fusion can be adopted to support various strategies for fault tolerance. First of all, it can be used for sensor error detection. Comparison of the sensors outputs points out the presence of erroneous measurements and, thus, of faulty sensors: an error is detected whenever the compared sensors outputs are different and exceed the intrinsic tolerance due to the measurement uncertainty. The faulty sensor is the one that disagrees from the value delivered by the other sensor. Error correction can be realized by majority voting the sensor outputs (an odd number of sensors is required): the output value – including the tolerance of the measurement uncertainty– on which there is the higher consensus is assumed as the correct value by masking the actual presence of the error. To preserve the detection and correction abilities as much as possible, fault insulation must be applied, by removing the faulty sensor from the active operation (i.e., by ignoring its outputs). It is worth nothing that these abilities are somewhat reduced when a sensor becomes faulty and is insulated since its contribution to the comparisons is now missing. Repair allows for recovering the sensor in the normal operation and for restoring the full fault tolerance abilities.

Different kind of sensors to measure the same physical quantity may enhance the fault tolerance. Different types of sensors will have different wear, or will be subject to different

aging mechanisms, or will have different faults and errors. Diversity minimizes the probability that sensors are progressively changing in a similar way about at the same time.

Sensors for different physical quantities of the same physical region can be adopted also to overcome possible drifting or temporary errors in measurements due to local transient events in sensors that have no specific relevance for the observation of the whole system [93,95,97,99,100]. Information produced by a sensor, which are not consistent with the whole picture of the system as created by the other sensors, can be identified as erroneous and, thus, ignored.

Instead of relying on comparison among real data, diagnosis can be also performed by adopting a model-based approach [97,101,102]. A model of the sensor is created by means of system identification techniques (e.g., by using neural models). The model is exploited to predict the expected sensor output from the sensor past outputs without any physical redundancy of the sensor: if the expected output differs too much from the actual one, an error is detected.

3.3.5 Virtual sensors and remote sensing

A real sensor can be used when there are sensing materials and techniques that allow for observing the desired quantity and when this sensor can be placed in the desired location of the system to perform the measurement. In some cases, this is not feasible since direct sensing of the desired physical quantity is not technically viable, or is not convenient for the application, or can be dangerous for the system and operator safety, or is economically expensive.

In some cases, although the desired quantity is difficult to be directly measured, other quantities strictly related to it can be observed more easily. An indirect measurement procedure can thus be created. Sensors are placed in the system (where feasible, appropriate, or convenient) to observe the quantities that can be directly measured. The laws that describe the relationships among the quantities measured by these sensors and with the desired quantity are identified: they can involve mechanical physics, chemistry, optics, electromagnetism, etc. From these laws it is possible to extract a function that gives the indirect measurement of the desired quantity from the values of the directly measured quantities. The sensors for direct measurements and this function constitute a *virtual sensor*. It is virtual since it is not a physically existing sensor to directly observe the desired quantity.

Since neural networks can be widely used as function approximators, they are also effective as data processing tools to merge the values coming from the physical sensors according to the merging function that computes the indirect measurement by applying the relationships among the measured quantities (e.g., [103,104]).

A special case of virtual sensor is when the quantity to be measured is in a location too far from the measuring system. In other cases the quantity to be measured involves a wide region of space and would require a too high number of sensors or an iterative sensing in the whole region, while the desired quantity is only a concise information (e.g., average or total value). In both of these types of cases only an indirect measurement technique is appropriate: in the literature this approach is known as *remote sensing*.

Examples of these measurements taken from satellites encompass the Earth surface parameters (e.g., the canopy temperature, the soil temperature, the canopy water content, and the soil moisture content), the rainfall, the snowfall, the air pollution, the CO emission, the ozone hole, etc. Also in these applications the neural networks proved their effectiveness and – sometimes – their superiority in merging information and extracting concise views [105-109].

3.3.6 High-level sensors

Several applications needs to extract a compact representation of the observed phenomenon or system, by analyzing and combining an often large quantity of data coming from many sensors. This is the case – only to mention very few examples – of production monitoring, product quality assessment, on-line diagnosis of complex systems, human health monitoring, object detection and recognition in vision, motion analysis, pattern and image recognition, decision making, risk prediction, and finance applications.

Only an abstract quantity that concisely characterizes the status and the behavior of the envisioned system is of interest, not the whole mass of sensors data. This quantity does not need to be physical, although it is related to physical quantities. For example, in the product quality monitoring, this concise information is the quality of the product, i.e., the complete integrity of the product or the presence of possible production defects; different classes of quality can be defined and produced object must be analyzed and properly classified.

For these abstract quantities there is obviously no physical sensor to perform direct measurements. However, like in virtual sensors, data collected from real sensors are processed to obtain the measure of the desired abstract quantity (e.g., the quality class in the example above).

Typical processing operations to create high-level sensors are classification and clustering of sensor data. Neural networks have been widely experimented and shown effective as tools to perform these tasks. In the literature many examples and techniques are reported [2-7], according to the presence of a supervisor guidance to create the classes or to the autonomous clustering by similarity. Among the many, some examples available in the literature concerning high-level sensors are [110-112].

3.3.7 Distributed intelligent sensing systems

Measurement systems and related industrial applications are continuously increasing in size and complexity. In particular, they are including many sensors to gather as much information as possible about the status and the evolution of the physical system. In many cases sensors and measurement systems need to be distributed in spaces that can range from a house to a production plant, to a metropolitan to a large geographical area, and even to the whole Earth, in order to collect the desired information and provide a comprehensive view of the whole monitored system. In these cases measurement operations need to be performed in processing architectures, connected in a computer network in order to exchange information and cooperate.

In the literature *networked sensing systems* and *distributed measurement systems* are two models that were proposed for creating these complex measurement systems (e.g., [113-116]). In networked sensing systems, the sensors are connected to a network (either private or public) so that a centralized measurement procedure can request for measurements from sensors and create a comprehensive view of the system by working in a centralized way: only physical sensors are decentralized in the monitored space. In distributed measurement systems the measurement procedure is distributed as well. Data acquisition is performed by a group of cooperating processing systems; each processing unit creates a partial view of the whole system picture by interacting with other units.

A model recently proposed in the field of artificial intelligence is the *perceptive agency* [117]. From a general logical point of view, agents are executors of specific activities, while an agency is a place where agents can meet, interact, and exchange information. Agents can be program running on computers, or hardware architectures dedicated to specific tasks, or robots able to physically interact with the external world. Agents can be fixed or mobile

(e.g., mobile software agents able to travel in the computer network, or mobile robots). An agency is a program running on a computer or a computer network to support agent cooperation. A perceptive agency is an agency in which the agents cooperate to perform measurements and monitor the desired system. Differently from distributed measurement systems, in a perceptive agency the components (i.e., the agents) do not know each other in advance: each declares which are its features and cooperation is dynamically built through an interactive agreement process. Such an approach allows – in particular – for high modularity, scalability, fault tolerance, and adaptability.

In all of the distributed architectures mentioned above for the sensing and measurement system, the neural networks can play different relevant roles: they can be used to enhance the individual sensors, to merge multisensor information as virtual sensors, to support adaptive remote sensing, and to create high-level sensors based on distributed information. In summary they can be used to introduce flexibility and adaptability in the distributed sensing and measurement systems, making more "intelligent" these procedure.

3.3.8 Calibration

Calibration [118] is the operation that establish, under given conditions, the relationship between values produced by a sensor or an instrument and the known values of the measurand. In the practice, similarly to sensor linearization, this operation consists in identifying a relationship to convert the physical sensor output into an ideal sensor output. The ideal (although not necessarily linear) description of the sensor behavior is appreciated in the applications to specify the desired behavior on the basis of ideal reference sensors so as to avoid to know the actual details of the specific sensor that has been installed in the system.

Implementation of this conversion relationship may consist either in a look-up table or in a function. As in sensor linearization, the use of a function allows for saving a large amount of memory space. To identify the function global interpolation techniques (e.g., Newton's or Lagrange's interpolation) can be adopted: they compute the polynomial – of a given order – that passes through all calibration samples; coefficients are computed by looking to the whole interval in which the function has to be defined. Local interpolation techniques (e.g., splines) look for the polynomial (of a given order) that passes through the samples contained in small windows (only few samples long) of the whole function codomain. Regression techniques (e.g., least mean squares) looks for functions that approximates the samples, without necessarily passing through them, by minimizing the global approximation error.

Feed-forward neural networks (as universal function approximators) are another regression-type technique that can be effectively used to approximate the desired function described by the sampled calibration data. In several cases neural networks have shown a higher approximation ability, accuracy, robustness, and generalization ability than conventional regression techniques at a similar or mildly higher computational complexity both for static and dynamic calibration (e.g., [119-122]). High generalization ability is highly appreciated in calibration since it allows to achieve the same calibration quality with a smaller number of samples. Conventional regression techniques need to know the maximum order of the polynomial to be used for approximation: neural networks are autonomously able to find the best approximation for the given network dimension.

The sensor fusion ability of neural networks can also be exploited to easier calibrate sensors in which the operating conditions depend from other parameters (e.g., the temperature for a high-accuracy pressure sensor [122]) as well as to calibrate multisensor systems (e.g., [123]).

References

[1] L. Cristaldi, A. Ferrero, and V. Piuri, "Programmable instruments, virtual instruments, and distributed measurement systems: what is really useful, innovative and technically sound?," *IEEE Instrumentation & Measurement Mag.*, vol. 2, pp. 20–27, Sept. 1999.

[2] J. Hertz, A. Krogh, and R. G. Palmer, *An Introduction to the Theory of Neural Computation*. Lecture Notes Volume I, Addison Wesley, 1991.

[3] E. Sinencio-Sanchez and C. Lau, *Artificial Neural Networks: Paradigms, Applications, and Hardware Implementations*. IEEE Press, Dec. 1992.

[4] J. Zurada, *Introduction to Artificial Neural Systems*. St. Paul: West Publishing Company, 1992.

[5] L. Fausett, *Fundamentals of Neural Networks*. Prentice Hall, Englewood Cliffs, 1994.

[6] S. Haykin, *Neural networks: a comprehensive foundation*. New Jersey, USA: Prentice Hall, 1999.

[7] T. Kohonen, *Self-Organizing Maps*, vol. 30 of *Springer Series in Information Sciences*. Berlin, Heidelberg, New York: Springer, 3 ed., 2001.

[8] C. Alippi, A. Ferrero, and V. Piuri, "Artificial intelligence for instruments and measurement applications," *IEEE Instrumentation & Measurement Mag.*, vol. 1, pp. 9–17, June 1998.

[9] C. Alippi, S. Ferrari, V. Piuri, M. Sami, and F. Scotti, "New trends in intelligent systems design for embedded and measurement application," *IEEE Instrumentation & Measurement Mag.*, vol. 2, pp. 36–44, June 1999.

[10] C. Alippi, V. Piuri, and F. Scotti, "Accuracy versus complexity in RBF neural networks," *IEEE Instrumentation & Measurement Mag.*, vol. 4, pp. 32–36, Mar. 2001.

[11] A. Weigend and D. Rumelhart, "The effective dimension of the space of hidden units," in *Proc. IEEE Int. Joint Conf. on Neural Networks, 1991*, vol. 3, pp. 2069–2074, 1991.

[12] C. Alippi, R. Petracca, and V. Piuri, "Off-line performance maximization in feedforward neural networks by applying virtual neurons and covariance transformations," in *Proc. IEEE Int. Symp. on Circuits and Systems, 1995*, pp. III.2197–III.2200, Apr. 1995.

[13] N. Murata, S. Yoshizawa, and S. Amari, "Network information criteriondetermining the number of hidden units for an artificial neural network model," *IEEE Trans. on Neural Networks*, vol. 5, pp. 865–872, Nov. 1994.

[14] K. Fukunaga, *Introduction to statistical pattern recognition*. New York: Academic Press, 1972.

[15] Y. Ota and B. Wilamowski, "Analog implementation of pulse-coupled neural networks," *IEEE Trans. on Neural Networks*, vol. 10, pp. 539–544, May 1999.

[16] H. Abdelbaki, E. Gelenbe, and S. El-Khamy, "Analog hardware implementation of the random neural network model," in *Proc. IEEE-INNS-ENNS Int. Joint Conf. on Neural Networks, 2000*, pp. 197–201, 2000.

[17] G. Indiveri, "A neuromorphic VLSI device for implementing 2D selective attention systems," *IEEE Trans. on Neural Networks*, vol. 12, pp. 1455–1463, Nov. 2001.

[18] C. Lu, B. Shi, and L. Chen, "Hardware implementation of an on-chip BP learning neural network with programmable neuron characteristics and learning rate adaptation," in *Proc. Int. Joint Conf. on Neural Networks, 2001*, pp. 212–215, 2001.

[19] A. Ö•renci, G. Dundar, and S. Balkir, "Fault-tolerant training of neural networks in the presence of MOS transistor mismatches," *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, pp. 272–281, Mar. 2001.

[20] A. Heittmann and U. Ruckert, "Mixed mode VLSI implementation of a neural associative memory," in *Proc. Seventh Int. Conf. on Microelectronics for Neural, Fuzzy and Bio-Inspired Systems, 1999*, pp. 299–306, 1999.

[21] K. Waheed and F. Salam, "A mixed mode self-programming neural system-on-chip for real-time applications," in *Proc. Int. Joint Conf. on Neural Networks, 2001*, vol. 1, pp. 195–200, 2001.

[22] S. Kung, "Tutorial: digital neurocomputing for signal/image processing," in *Proc. 1991 IEEE Workshop Neural Networks for Signal Processing*, pp. 616–644, 1991.

[23] M. Yasunaga, N. Masuda, M. Yagyu, M. Asai, K. Shibata, M. Ooyama, M. Yamada, T. Sakaguchi, and M. Hashimoto, "A self-learning digital neural network using wafer-scale LSI," *IEEE J. of Solid-State Circuits*, vol. 28, pp. 106–114, Feb. 1993.

[24] C. Chin Wang, C. Jung Huang, and Y. Pei Chen, "Design of an innerproduct processor for hardware realization of multi-valued exponential bidirectional associative memory," *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, pp. 1271–1278, Nov. 2000.

[25] R. Perfetti and G. Costantini, "Multiplierless digital learning algorithm for cellular neural networks," *IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications*, vol. 48, pp. 630–635, May 2001.

[26] T. Schoenauer, S. Atasoy, N. Mehrtash, and H. Klar, "NeuroPipe-chip: A digital neuro-processor for spiking neural networks," *IEEE Trans. on Neural Networks*, vol. 13, pp. 205–213, Jan. 2002.
[27] M. Arroyo León, A. Ruiz Castro, and R. Leal Ascencio, "An artificial neural network on a field programmable gate array as a virtual sensor," in *Proc. Third Int. Workshop on Design of Mixed-Mode Integrated Circuits and Applications, 1999*, pp. 114–117, 1999.
[28] G. Frank, G. Hartmann, A. Jahnke, and M. Schäfer, "An accelerator for neural networks with pulse-coded model neurons," *IEEE Trans. on Neural Networks*, vol. 10, pp. 527–538, May 1999.
[29] C.-M. Kim and S. Y. Lee, "A digital chip for robust speech recognition in noisy environment," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, 2001*, vol. 2, pp. 1089–1092, 2001.
[30] Intel Corp., Santa Clara, CA, *80170NX ETANN, Data Sheet*, 1991.
[31] Y. Sato, K. Shibata, M. Asai, M. Ohki, M. Sugie, T. Sakaguchi, M. Hashimoto, and Y. Kuwabara, "Development of a high-performance general purpose neuro-computer composed of 512 digital neurons," in *Proc. Int. Joint Conf. on Neural Networks, 1993*, vol. 2, pp. 1967–1970, 1993.
[32] U. Ramacher, W. Raab, J. Hachmann, J. Beichter, N. Bruls, M.Wesseling, E. Sicheneder, J. Glass, A. Wurz, and R. Manner, "SYNAPSE-1: a highspeed general purpose parallel neurocomputer system," in *Proc. 9th Int. Parallel Processing Symp., 1995*, pp. 774–781, 1995.
[33] U. Müller, A. Gunzinger, and W. Guggenbühl, "Fast neural net simulation with a DSP processor array," *IEEE Trans. on Neural Networks*, vol. 6, pp. 203–213, Jan. 1995.
[34] M. Murakawa, S. Yoshizawa, I. Kajitani, X. Yao, N. Kajihara, M. Iwata, and T. Higuchi, "The GRD chip: genetic reconfiguration of DSPs for neural network processing," *IEEE Trans. on Computers*, vol. 48, pp. 628–639, June 1999.
[35] V. Cantoni and A. Petrosino, "Neural recognition in a pyramidal structure," *IEEE Trans. on Neural Networks*, vol. 13, pp. 472–480, Mar. 2002.
[36] E. Kerckhoffs, F. Wedman, and E. Frietman, "Speeding up backpropagation training on a hypercube computer," *J. of Neurocomputing*, vol. 4, pp. 43–63, 1992.
[37] M. Kuga, Y. Namiuchi, B. Apduhan, and T. Sueyoshi, "Implementation and performance evaluation of a neural network simulator on highly parallel computer AP-1000," in *Proc. 1993 Int. Conf. on Parallel And Distributed Systems*, pp. 722–726, July 1993.
[38] X. Liu and G. Wilcox, "Benchmarking of the CM-5 and the Cray machines with a very large backpropagation neural network," in *IEEE Int. Conf. on Neural Networks, 1994*, vol. 1, pp. 22–27, 1994.
[39] H. Demuth and M. Beale, *Neural Network Toolbox Users Guide*. The MathWorks, Inc., 7 ed., Mar. 2001.
[40] G. D. Micheli and M. Sami, eds., *Hardware/Software CoDesign*, vol. 310 of *NATO ASI Series*. Kluwer Academic Publishers, 1995.
[41] D. D. Gajski, F. Vahid, S. Narayan, and J. Gong, *Specification and Design of Embedded Systems*. Englewood Cliffs, New Jersey 07632: Prentice Hall, 1994.
[42] Ptolemy – http://ptolemy.eecs.berkeley.edu/
[43] C. Nilson, R. Darling, and R. Pinter, "Shunting neural network photodetector arrays in analog CMOS," *IEEE J. of Solid-State Circuits*, vol. 29, pp. 1291–1296, Oct. 1994.
[44] T. Yagi, Y. Hayashida, and S. Kameda, "An analog VLSI which emulates biological vision," in *Proc. Second Int. Conf. on Knowledge-Based Intelligent Electronic Systems, 1998*, vol. 3, pp. 454–460, 1998.
[45] M. Wilcox and D. Thelen Jr., "A retina with parallel input and pulsed output, extracting high-resolution information," *IEEE Trans. on Neural Networks*, vol. 10, pp. 574–583, May 1999.
[46] M. Becker, R. Eckmiller, and R. Hunermann, "Psychophysical test of a tunable retina encoder for retina implants," in *Proc. Int. Joint Conf. on Neural Networks, 1999*, vol. 1, pp. 192–195, 1999.
[47] W. Liu, K. Vichienchom, M. Clements, S. DeMarco, C. Hughes, E. McGucken, M. Humayun, E. D. Juan, J. Weiland, and R. Greenberg, "A neuro-stimulus chip with telemetry unit for retinal prosthetic device," *IEEE J. of Solid-State Circuits*, vol. 35, pp. 1487–1497, Oct. 2000.
[48] C.-Y. Wu, L.-J. Lin, and K.-H. Huang, "A new light-activated CMOS retinal-pulse generation circuit without external power supply for artificial retinal prostheses," in *The 8th IEEE Int. Conf. on Electronics, Circuits and Systems, 2001*, vol. 2, pp. 619–622, 2001.
[49] S. Watanabe and M. Yoneyama, "An ultrasonic visual sensor for threedimensional object recognition using neural networks," *IEEE Trans. on Robotics and Automation*, vol. 8, pp. 240–249, Apr. 1992.
[50] C.-F. Chiu and C.-Y.Wu, "The design of rotation-invariant pattern recognition using the silicon retina," *IEEE J. of Solid-State Circuits*, vol. 32, pp. 526–534, Apr. 1997.

[51] Z. Lu and B. Shi, "Subpixel resolution binocular visual tracking using analog VLSI vision sensors," *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, pp. 1468–1475, Dec. 2000.

[52] N. Goerke, R. Schatten, and R. Eckmiller, "Enhancing active vision by a neural movement predictor," in *Proc. Int. Joint Conf. on Neural Networks, 2001*, vol. 2, pp. 1312–1317, 2001.

[53] G. Foresti and S. Gentili, "A hierarchical classification system for object recognition in underwater environments," *IEEE J. of Oceanic Engineering*, vol. 27, pp. 66–78, Jan. 2002.

[54] M. Leisenberg, "Hearing aids for the profoundly deaf based on neural net speech processing," in *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, 1995*, vol. 5, pp. 3535–3538, 1995.

[55] C.-H. Chang, G. Anderson, and P. Loizou, "A neural network model for optimizing vowel recognition by cochlear implant listeners," *IEEE Trans. on Neural Systems and Rehabilitation Engineering*, vol. 9, pp. 42–48, Mar. 2001.

[56] C. Di Natale, A. Macagnano, R. Paolesse, E. Tarizzo, A. D'Amico, F. Davide, T. Boschi, M. Faccio, G. Ferri, F. Sinesio, F. Bucarelli, E. Moneta, and G. Quaglia, "A comparison between an electronic nose and human olfaction in a selected case study," in *Proc. Int. Conf. on Solid State Sensors and Actuators, 1997*, vol. 2, pp. 1335–1338, 1997.

[57] P. Wide, F. Winquist, P. Bergsten, and E. Petriu, "The human-based multisensor fusion method for artificial nose and tongue sensor data," *IEEE Trans. on Instrumentation and Measurement*, vol. 47, pp. 1072–1077, Oct. 1998.

[58] R. Dowdeswell and P. Payne, "Odour measurement using conducting polymer gas sensors and an artificial neural network decision system," *Engineering Science and Education Journal*, vol. 8, pp. 129–134, June 1999.

[59] E. Hines, E. Llobet, and J. Gardner, "Electronic noses: a review of signal processing techniques," *IEE Proc. — Circuits, Devices and Systems*, vol. 146, pp. 297–310, Dec. 1999.

[60] G. Canepa, M. Morabito, D. De Rossi, A. Caiti, and T. Parisini, "Shape from touch by a neural net," in *Proc. IEEE Int. Conf. on Robotics and Automation, 1992*, vol. 3, pp. 2075–2080, 1992.

[61] W. McMath, M. Colven, S. Yeung, and E. Petriu, "Tactile pattern recognition using neural networks," in *Proc. Int. Conf. on Industrial Electronics, Control, and Instrumentation, 1993*, vol. 3, pp. 1391–1394, 1993.

[62] A. Caiti, G. Canepa, D. De Rossi, F. Germagnoli, G. Magenes, and T. Parisini, "Towards the realization of an artificial tactile system: fineform discrimination by a tensorial tactile sensor array and neural inversion algorithms," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 25, pp. 933–946, June 1995.

[63] G. Canepa, R. Petrigliano, M. Campanella, and D. D. Rossi, "Detection of incipient object slippage by skin-like sensing and neural network processing," *IEEE Trans. on Systems, Man and Cybernetics, Part B*, vol. 28, pp. 348–356, June 1998.

[64] J. Patra, A. Kot, and G. Panda, "An intelligent pressure sensor using neural networks," *IEEE Trans. on Instrumentation and Measurement*, vol. 49, pp. 829–834, Aug. 2000.

[65] S. Aisawa, K. Noguchi, and T. Matsumoto, "Neural processing-type displacement sensor employing multimode waveguide," *IEEE Photonics Technology Letters*, vol. 3, pp. 394–396, Apr. 1991.

[66] A. Carullo, F. Ferraris, S. Graziani, U. Grimaldi, and M. Parvis, "Ultrasonic distance sensor improvement using a two-level neural-network," *IEEE Trans. on Instrumentation and Measurement*, vol. 45, pp. 677–682, Apr. 1996.

[67] K. Zhang, C. Butler, Q. Yang, and Y. Lu, "A fiber optic sensor for the measurement of surface roughness and displacement using artificial neural networks," *IEEE Trans. on Instrumentation and Measurement*, vol. 46, no. 4, pp. 899–902, 1997.

[68] J. Kramer, R. Sarpeshkar, and C. Koch, "Pulse-based analog VLSI velocity sensors," *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 44, pp. 86–101, Feb. 1997.

[69] G. Brasseur, "Modeling of the front end of a new capacitive finger-type angular-position sensor," *IEEE Trans. on Instrumentation and Measurement*, vol. 50, pp. 111–116, Feb. 2001.

[70] K.-J. Xu and C. Li, "Dynamic decoupling and compensating methods of multi-axis force sensors," *IEEE Trans. on Instrumentation and Measurement*, vol. 49, pp. 935–941, Oct. 2000.

[71] M. H. Choi and W. W. Lee, "A force/moment sensor for intuitive robot teaching application," in *Proc. IEEE Int. Conf. on Robotics and Automation, 2001*, vol. 4, pp. 4011–4016, 2001.

[72] B. Fahimi, G. Suresh, and M. Ehsani, "Torque estimation in switched reluctance motor drive using artificial neural networks," in *Proc. 23rd Int. Conf. on Industrial Electronics, Control and Instrumentation, 1997*, vol. 1, pp. 21–26, 1997.

[73] F. Discenzo, F. Merat, D. Chung, and P. Unsworth, "Low-cost optical neural-net torque transducer," in *IEE Colloquium on Intelligent and Self-Validating Sensors (Ref. No. 1999/160)*, pp. 15/1–15/4, 1999.

[74] W. Bock, E. Porada, and M. Zaremba, "Neural processing-type fiberoptic strain sensor," *IEEE Trans. on Instrumentation and Measurement*, vol. 41, pp. 1062–1066, Dec. 1992.
[75] J. Dias Pereira, O. Postolache, and P. Silva Girão, "A temperature compensated system for magnetic field measurements based on artificial neural networks," *IEEE Trans. on Instrumentation and Measurement*, vol. 47, pp. 494–498, Apr. 1998.
[76] C. Chan, W. Jin, A. Rad, and M. Demokan, "Simultaneous measurement of temperature and strain: an artificial neural network approach," *IEEE Photonics Technology Letters*, vol. 10, pp. 854–856, June 1998.
[77] S.-L. Tsao, J. Wu, and B.-C. Yeh, "High-resolution neural temperature sensor using fiber Bragg gratings," *IEEE J. of Quantum Electronics*, vol. 35, pp. 1590–1596, Nov. 1999.
[78] A. Chatterjee, S. Munshi, M. Dutta, and A. Rakshit, "An artificial neural linearizer for capacitive humidity sensor," in *Proc. 17th IEEE Instrumentation and Measurement Technology Conf., 2000*, vol. 1, pp. 313–317, 2000.
[79] M. Dawson, A. Fung, and M. Manry, "A robust statistical-based estimator for soil moisture retrieval from radar measurements," *IEEE Trans. on Geoscience and Remote Sensing*, vol. 35, pp. 57–67, Jan. 1997.
[80] H.-K. Hong, H. W. Shin, H. S. Park, D. H. Yun, C. H. Kwon, K. Lee, S.-T. Kim, and T. Moriizumi, "Gas identification using oxide semiconductor gas sensor array and neural-network pattern recognition," in *The 8th Int. Conf. on Solid-State Sensors and Actuators, 1995 and Eurosensors IX*, vol. 1, pp. 687–690, 1995.
[81] T. Lu and J. Lerner, "Spectroscopy and hybrid neural network analysis," *Proc. IEEE*, vol. 84, pp. 895 –905, June 1996.
[82] M. Giacomini, C. Ruggiero, S. Bertone, and L. Calegari, "Artificial neural network identification of heterotrophic marine bacteria based on their fatty-acid composition," *IEEE Trans. on Biomedical Engineering*, vol. 44, pp. 1185–1191, Dec. 1997.
[83] A. Pardo, S. Marco, and J. Samitier, "Nonlinear inverse dynamic models of gas sensing systems based on chemical sensor arrays for quantitative measurements," *IEEE Trans. on Instrumentation and Measurement*, vol. 47, pp. 644–651, June 1998.
[84] S. Osowski and K. Brudzewski, "Hybrid neural network for gas analysis measuring system," in *Proc. 16th IEEE Instrumentation and Measurement Technology Conf., 1999*, vol. 1, pp. 440–444, 1999.
[85] T. Sobanski, A. Szczurek, and B. Licznerski, "Application of sensor array and artificial neural network for discrimination and qualification of benzene and ethylbenzene," in *24th Int. Spring Seminar on Electronics Technology: Concurrent Engineering in Electronic Packaging, 2001*, pp. 150–153, 2001.
[86] M. Attari, F. Boudjema, and M. Heniche, "An artificial neural network to linearize a G (tungsten vs. tungsten 26% rhenium) thermocouple characteristic in the range of zero to 2000°C," in *Proc. IEEE Int. Symp. on Industrial Electronics, 1995*, vol. 1, pp. 176–180, 1995.
[87] G. Dempsey, N. Alt, B. Olson, and J. Alig, "Control sensor linearization using a microcontroller-based neural network," in *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics, 1997*, vol. 4, pp. 3078–3083, 1997.
[88] N. Medrano-Marqués and B. Martín-del-Brío, "Sensor linearization with neural networks," *IEEE Trans. on Industrial Electronics*, vol. 48, pp. 1288–1290, Dec. 2001.
[89] S. Ben-Yacoub, Y. Abdeljaoued, and E. Mayoraz, "Fusion of face and speech data for person identity verification," *IEEE Trans. on Neural Networks*, vol. 10, pp. 1065–1074, Sept. 1999.
[90] A. Filippidis, L. Jain, and N. Martin, "Multisensor data fusion for surface land-mine detection," *IEEE Trans. on Systems, Man and Cybernetics, Part C*, vol. 30, pp. 145–150, Feb. 2000.
[91] Z. Zhang, S. Sun, and F. Zheng, "Image fusion based on median filters and SOFM neural networks: a three-step scheme," *Signal Processing*, vol. 81, pp. 1325–1330, June 2001.
[92] Y. Xia, H. Leung, and E. Bossé, "Neural data fusion algorithms based on a linearly constrained least square method," *IEEE Trans. on Neural Networks*, vol. 13, pp. 320–329, Mar. 2002.
[93] M. Napolitano, G. Silvestri, D. Windon II, J. Casanova, and M. Innocenti, "Sensor validation using hardware-based on-line learning neural networks," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 34, pp. 456–468, Apr. 1998.
[94] O. Postolache, P. Girão, H. Ramos, and J. Dias Pereira, "A temperature sensor fault detector as an artificial neural network application," in *Proc. MELECON 98, 9th Mediterranean Electrotechnical Conf., 1998*, vol. 1, pp. 678–682, 1998.
[95] Y. Liu, Y. Shen, and H. Hu, "A new method for sensor fault detection, isolation and accommodation," in *Proc. 16th IEEE Instrumentation and Measurement Technology Conf., 1999*, vol. 1, pp. 488–492, 1999.

[96] T. Long, E. Hanzevack, and W. Bynum, “Sensor fusion and failure detection using virtual sensors,” in *Proc. 1999 American Control Conf.*, vol. 4, pp. 2417 –2421, 1999.
[97] G. Betta and A. Pietrosanto, “Instrument fault detection and isolation: state of the art and new research trends,” *IEEE Trans. on Instrumentation and Measurement*, vol. 49, pp. 100–107, Feb. 2000.
[98] A. Sachenko, V. Kochan, V. Turchenko, V. Golovko, J. Savitsky, A. Dunets, and T. Laopoulos, “Sensor errors prediction using neural networks,” in *Proc. IEEE-INNS-ENNS Int. Joint Conf. on Neural Networks, 2000*, vol. 4, pp. 441–446, 2000.
[99] H. Jin, C. Chan, H. Zhang, and W. Yeung, “Fault detection of redundant systems based on B-spline neural network,” in *Proc. 2000 American Control Conf.*, vol. 2, pp. 1215–1219, 2000.
[100] G. Yen and W. Feng, “Winner take all experts network for sensor validation,” in *Proc. 2000 IEEE Int. Conf. on Control Applications, 2000*, pp. 92–97, 2000.
[101] E. Eryurek and B. Upadhyaya, “Sensor validation for power plants using adaptive backpropagation neural network,” *IEEE Trans. on Nuclear Science*, vol. 37, pp. 1040–1047, Apr. 1990.
[102] S. Naidu, E. Zafiriou, and T. McAvoy, “Use of neural networks for sensor failure detection in a control system,” *IEEE Control Systems Mag.*, vol. 10, pp. 49–55, Apr. 1990.
[103] K. Cohen, Y. Hu, W. Tompkins, and J.Webster, “Breath detection using a fuzzy neural network and sensor fusion,” in *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing, 1995*, vol. 5, pp. 3491–3494, 1995.
[104] A. Chong, S. Wilcox, and J. Ward, “Prediction of gaseous emissions from a chain grate stoker boiler using neural networks of ARX structure,” *IEE Proc. — Science, Measurement and Technology*, vol. 148, pp. 95–102, May 2001.
[105] Y. Ninomiya, “Quantitative estimation of SiO_2 content in igneous rocks using thermal infrared spectra with a neural network approach,” *IEEE Trans. on Geoscience and Remote Sensing*, vol. 33, pp. 684–691, May 1995.
[106] D. Tsintikidis, J. Haferman, E. Anagnostou, W. Krajewski, and T. Smith, “A neural network approach to estimating rainfall from spaceborne microwave data,” *IEEE Trans. on Geoscience and Remote Sensing*, vol. 35, pp. 1079–1093, Sept. 1997.
[107] P. Chang and L. Li, “Ocean surface wind speed and direction retrievals from the SSM/I,” *IEEE Trans. on Geoscience and Remote Sensing*, vol. 36, pp. 1866–1871, Nov. 1998.
[108] Y.-A. Liou, Y. Tzeng, and K. Chen, “A neural-network approach to radiometric sensing of land-surface parameters,” *IEEE Trans. on Geoscience and Remote Sensing*, vol. 37, pp. 2718–2724, Nov. 1999.
[109] C. Clerbaux, J. Hadji-Lazaro, S. Payan, C. Camy-Peyret, and G. Mégie, “Retrieval of CO columns from IMG/ADEOS spectra,” *IEEE Trans. on Geoscience and Remote Sensing*, vol. 37, pp. 1657–1661, May 1999.
[110] B. Arrue, A. Ollero, and J. M. de Dios, “An intelligent system for false alarm reduction in infrared forest-fire detection,” *IEEE Intelligent Systems*, vol. 15, pp. 64–73, May 2000.
[111] T. Chady, M. Enokizono, R. Sikora, T. Todaka, and Y. Tsuchida, “Natural crack recognition using inverse neural model and multi-frequency eddy current method,” *IEEE Trans. on Magnetics*, vol. 37, pp. 2797–2799, July 2001.
[112] T. Chady, M. Enokizono, and R. Sikora, “Signal restoration using dynamic neural network model for eddy current nondestructive testing,” *IEEE Trans. on Magnetics*, vol. 37, pp. 3737–3740, Sept. 2001.
[113] G. Pottie and W. Kaiser, “Wireless integrated network sensors,” *Communications of the ACM*, vol. 43, pp. 51–58, May 2000.
[114] R. Van Dyck and L. Miller, “Distributed sensor processing over an ad hoc wireless network: simulation framework and performance criteria,” in *IEEE Military Communications Conf., 2001*, vol. 2, pp. 894–898, 2001.
[115] L. Guibas, “Sensing, tracking and reasoning with relations,” *IEEE Signal Processing Mag.*, vol. 19, pp. 73–85, Mar. 2002.
[116] F. Zhao, J. Shin, and J. Reich, “Information-driven dynamic sensor collaboration,” *IEEE Signal Processing Mag.*, vol. 19, pp. 61–72, Mar. 2002.
[117] F. Amigoni, A. Brandolini, G. DAntona, R. Ottoboni, and M. Somalvico, “Artificial intelligence in science of measurements and the evolution of the measurements instruments: A perspective conception,” in *Proc. 2002 IEEE Int. Symp. on Virtual and Intelligent Measurement Systems*, pp. 26–31, May 2002.
[118] J. Dias Pereira, P. Silva Girão, and O. Postolache, “Fitting transducer characteristics to measured data,” *IEEE Instrumentation & Measurement Mag.*, vol. 4, pp. 26–39, Dec. 2001.

[119] W. Bock, E. Porada, M. Beaulieu, and T. Eftimov, “Automatic calibration of a fiber-optic strain sensor using a self-learning system,” *IEEE Trans. on Instrumentation and Measurement*, vol. 43, pp. 341–346, Apr. 1994.
[120] P. Kluk and R. Morawski, “Static calibration of transducers using parametrization and neural-network-based approximation,” in *Proc. IEEE Instrumentation and Measurement Technology Conf., 1996*, vol. 1, pp. 581–585, June 1996.
[121] D. Massicotte, S. Legendre, and A. Barwicz, “Neural-network-based method of calibration and measurand reconstruction for a high-pressure measuring system,” *IEEE Trans. on Instrumentation and Measurement*, vol. 47, pp. 362–370, Apr. 1998.
[122] R. Schultz, “Applications of neural networks for transducer calibration and signal processing of transducer data containing periodic interference,” in *Proc. American Control Conf., 1999*, vol. 3, pp. 1661–1662, June 1999.
[123] T.-F. Lu, G. C. Lin, and J. R. He, “Neural-network-based 3D force/torque sensor calibration for robot applications,” *Engineering Applications of Artificial Intelligence*, vol. 10, pp. 87–97, Feb. 1997.