Emergent Coordination in Distributed Sensor Networks

Heikki Hyötyniemi

Cybernetics Group
Helsinki University of Technology, Control Engineering Laboratory
P.O. Box 5500, FIN-02015 HUT, Finland
heikki.hyotyniemi@hut.fi
http://www.control.hut.fi/hyotyniemi

Abstract. In today's systems, one would like to distribute the control of process components to reach better fault tolerance and independence of centralized hierarchies. However, there exist no general theoretical frameworks for mastering such distributed systems. In this paper, an approach is presented for implementing distributed sensor networks. Based on the modern cybernetic studies, it turns out that local interactions among sensors result in global behaviors: The grid of sensors implement principal componment filtering of the sensor signals.

1 Introduction

Remember "Solaris"? In this classic science fiction novel by Stanislaw Lem, people are faced with the Ocean, an example of *distributed intelligence.* This is a mindboggling idea — truly something too strange to be understood. However, today there is a need to master such distributed environments.

This need of conceptual tools for mastering distributed systems is not only practical; indeed, the same questions need to be answered to understand the very essence of intelligence. It seems that *emergence* is one of the key features common to real-life intelligent systems: From very local operations, some global functionality and adaptability emerges with no need of centralized coordination. For example, it is intuitively understood that in human and in ant societies the *whole* is more than its parts.

Similarly, in an industrial environment, a "society" of *sensors,* for example, could behave in an intelligent way — if only the principles of such distributed orchestration, or "ubiquitous computation", were found. The problem is that the traditional tools for AI are based on the idea of coordination and orchestrated co-operation.

This paper discusses a very narrow application area of network technologies. Even though this approach differs very much from mainstream Semantic Web applications, it can be instructive to see what can be reached when the problems are attacked applying domain-specific rather than completely general-purpose tools. In special domains one can reach the *deep structures* within the data

— and, indeed, it is claimed that in such domains one can extract not only information but also semantics-bound *knowledge* out from the observations. This makes it possible to implement new kinds of autonomous distributed systems.

2 Ontologies in a sensor system

There exist many industrial processes where the systems have to be described in terms of partial differential equation models with distributed parameters. In principle, in such systems complete information about the process state cannot be measured, the state being infinite-dimensional. However, the new sensor technology still promises enhanced information about the process state: It is possible to place high numbers of sensors in the process, forming a *distributed sensor network.* Using *bluetooth* techniques, etc., information exchange between neighboring sensors can be arranged.

Today's challenge is to orchestrate the sensors so that the best possible information is gathered from the process. In this context, one can speak of *sensor fusion;* the resulting "clever" virtual components can be called *soft sensors.* It seems that research on distributed networks today very much concentrates on explicitly finding the global structure for the sensor system. If the structure is known, it is, of course, simple to implement applications in a traditional, globally controlled, centralized way. For example, having a global model available, one can implement a *Kalman filter* for measurement enhancement (see [1]). The problem here is that finding the global model is by no means a simple task. What is more, it is intuitively clear that such centralized approaches are clumsy; they do not reflect the underlying distributed structure in a natural way.

Alternative approaches to modeling distributed systems have been studied actively. The mainstream framework for distributed systems is the *agent* perspective [13]. Unfortunately, there is no solid theory available for such agent systems. Another approach to networked systems has been studied, for example, in [2]; nevertheless, it seems the approaches typically are rather qualitative, not offering concrete design methodologies. Indeed, it seems that after years of fancy paradigms, it might be fruitful to look back at the very principles. The intuition here is that without working knowledge of the natural language of complex systems — that is, *mathematical system theory* — the ideas remain too vague, and the relevant concepts cannot be defined and appropriate issues cannot be discussed in a concrete enough way. It turns out that in a mathematically compact environment the connection between *data* and *structure* can be explicitly determined; it is interesting to see what kind of system properties are dictated by these intuitions.

Not all networks or agent systems can be studied in the mathematically compact framework, of course — some application domains truly are based on complex software architectures and are too disperse to be captured in a clear setting. Mathematical formalizations are not important *per se.* It is necessary that the essential phenomena can be quantified in a homogeneous framework; the goals of individual agents have to be compatible and mutually comparable,

that is, numerically characterizable. Using the jargon from the Semantic Web area, the *ontologies have to be machine-comprehensible.* In such a case, "agent fusion" can be carried out using multivariate statistical methods.

To make machine "understand" the data, *semantics* of that data needs to be somehow formalized and implemented in the data processing mechanisms. It turns out that the ideas of *naturalistic* and *contextual* semantics offer the key towards such "functionalized semantics": It is assumed that the meaning of data is determined by the connections to the outside environment, and by the connections among the processing elements, so that *ontologies are directly determined by the correlation structures among data.* When different processing elements can communicate with each other exchanging information, and when they can autonomously assess that information, the human can be detached from the data analysis loop; this opens up new perspectives, as highly iterative and boring data refinement procedures can be automated.

A sensor network where there are various more or less compatible measurements that should be appropriately combined, is an excellent example of such environments where semantics can be quantified. In what follows, the basic components, or "agents", in the sensor system are called *nodes.* The nodes are identical, there is some limited computing capacity available in them, and they can measure their environment; further, it is assumed that the nodes can transfer simple information to their (immediate) neighbors. It turns out that applying appropriate local algorithms, the system *state* (implicitly) emerges from the node network. In this paper, this internal state is applied for refining, or filtering, the sensor measurements in the nodes to attenuate noise. Also *actuators,* for example, could be implemented based on such models, so that a complete distributed control environment could be constructed; such issues are, however, not concentrated on here.

3 Structure from data

Computers implementing mathematical algorithms are good and tireless at refining and polishing parameters within some fixed structural framework. However, in a truly intelligent system, this is not enough — new structures need to emerge from computations.

Finding structure from observations is an age-old philosophical dilemma that is plaguing AI approaches today. This problem is now only made more challenging as this structure determination task should be carried out in a distributed manner. The claim here is that modern mathematical tools can solve this problem — at least in some cases. What are then such cases like? Intuition about this issue is just as important as the methodologies themselves.

It turns out that when a high-dimensional space is cleverly compressed, the resulting data structures typically have some relevance — in some cases it can even be claimed that they can be interpreted in terms of mental representations [8]. Also the mindless machine can search for the underlying statistical dependencies in the data, and, hopefully, some illusion of intelligence emerges.

In simple ("locally unimodal") cases this clever compression can be based on multivariate statistical analysis, and specially on *principal component analysis* PCA (for example, see [3]).

Assume that the observation data (mean-zeroed, real-valued signals) at time t are collected in the m dimensional vector $\chi(t)$. Further, assume that one has a large set of such data samples. The statistical properties of the data (up to the second order) are captured by the covariance matrix:

$$R_{\chi\chi} = \mathrm{E}\{\chi\chi^T\}. \tag{1}$$

The structure of such a matrix can best be studied in terms of *eigenvalues* and *eigenvectors*. It turns out that, because of the construction of the covariance matrix, one can write the eigenvalue decomposition simply as

$$R_{\chi\chi} = \Theta\Lambda\Theta^{-1} = \Theta\Lambda\Theta^T. \tag{2}$$

Here, Θ contains the m orthonormal eigenvectors as its columns, and the diagonal matrix Λ contains the corresponding m non-negative eigenvalues on its diagonal.

Now, it turns out that the eigenvalues λ_i directly determine the "significance" (in terms of data variance) of the corresponding eigenvector direction when explaining the data distribution in the m dimensional space. This means that data compression can be carried out in the mathematically optimal way (assuming Gaussianity of data) by ignoring the data variability in the less visible directions, that is, by eliminating those subspace directions altogether. After such data reduction has been carried out, and after the data has been projected onto the $n < m$ dimensional subspace, the resulting n "latent variables" (projections along the subspace axes) characterize the data. This reduced representation can efficiently be utilized, for example, for implementing *regression* onto the output space. Such *principal component regression* (PCR) turns out to be robust against noisy and collinear (redundant) data (for example, see [9]).

However, when applying PCA for compression of sensor information, for example, it turns out that there are shortcomings. If the sensors are measuring some dynamic process, static data vectors cannot capture the *essence* of the time-varying process: Typically there is inertia, memory, in the system, and the future behaviors are dependent of the past, not only of the current time.

Fortunately, system theory reveals that for a linear d'th order linear system, the history can be captured in a d dimensional *state vector*. This means that the dynamics, all relevant information about the past system behavior, can be captured in a static form. Still, the theory only promises the existence of such state; the challenge is to determine it.

The state is not unique, and it need *not* be minimal; and, indeed, it is a straightforward task to determine a non-optimal state vector that qualifies. For example, if one defines the (preliminary) state vector as

$$\chi(t) = \begin{pmatrix} y(t) \\ y(t-\kappa) \\ \vdots \\ y(t-(d-1)\kappa) \end{pmatrix} \tag{3}$$

where y is the μ dimensional system output vector, and κ is some sampling interval, it is certain that the system state is captured by the vector χ. Of course, this preliminary state is redundant, because its dimension is $d\kappa$ rather than d — but now the dynamics is captured in a static form, and it suffices to apply some compression technique. For example, if PCA is applied, one can find the minimal d dimensional state vector as

$$x(t) = \theta^T \cdot \chi(t). \tag{4}$$

Here, θ represents the set of d most significant of the eigenvectors in $R_{\chi\chi}$. When this state sequence has been found, it is easy to find the complete state-space model for the observed system. This means that the multivariate state-space model with noise models can be reconstructed from data in a more or less autonomous way. The final number of states d does not need to be determined beforehand, and the causal structure or connections between signals need not be determined beforehand: All these issues are settled only after the correlation structures between the signals are analyzed.

This approach to system identification is rather new, and it is known as *subspace identification* [14]. There exist some industrial applications of subspace identification (for example, see [5]), and in [10], the possibilities of using this technique as the "brains" of an smart process analysis device, are studied.

Yet, all of the above analyses still take place in a centralized way: The data has to be collected to some central unit that is used for constructing a global model for the system. And it is a *decentralized* scheme one is now looking for.

4 Decentralized PCA

The underlying theory here is based on neural networks research (for example, see [6]). One paradigm there concentrates on *principal component networks* [4]. The following is based on studies carried out for Hebbian neuron structures [7]; it turns out that the results can be generalized to other kinds of networks, too.

Assume that a continuous time state-space system is described (somewhat abnormally) as

$$\frac{d}{dt}x(t) = -Ax(t) + Bu(t). \tag{5}$$

Here, x is the n dimensional state vector, and u is the m dimensional input. Assuming that the dynamics of the input is much slower that what is the system dynamics, one can approximately write

$$x(t) = A^{-1}Bu(t) = \phi^T u(t), \tag{6}$$

that is, the system state can be calculated from the input using a linear static mapping, assuming invertibility of A. Now, if the matrices A and B are selected as

$$A = \mathrm{E}\left\{xx^T\right\} \tag{7}$$

and

$$B = \mathrm{E}\left\{xu^T\right\}, \tag{8}$$

so that the elements A_{ij} reflect the long-term *co-activity* between the state elements i and j, it turns out that in stationary state the system carries out *principal subspace analysis* for the input data. This is shown (using discrete-time formulation) in [11].

Principal subspace means that the vectors x occupy the same subspace as the n most significant principal components of the input data u. However, the principal components are not completely localized, meaning that there holds $\phi = \theta D$, where D is some invertible $n \times n$ matrix. As shown in [11], it is often not necessary to explicitly solve for the principal components; for example, when implementing principal component regression, vector x can directly be used. Assume that one wants to estimate y. To implement this, some extra operations are needed (see [11]): First, the normalized state can be found as

$$\frac{d}{dt}v(t) = -Av(t) + x(t), \tag{9}$$

and, after that, the estimate becomes

$$\hat{y}(t) = Cv, \tag{10}$$

where

$$C = \mathrm{E}\left\{yx^T\right\}. \tag{11}$$

Principal component regression from input *back* to input, or *filtering* of the measurements, can be carried out if one selects $y \equiv u$, and $C = B^T$. This "principal component filtering" can be written as a single state-space system (excplicit time indices being dropped for brevity)

$$\begin{cases} \dfrac{d}{dt}\begin{pmatrix} x \\ v \end{pmatrix} = \left(\begin{array}{c|c} -A & \mathbf{0} \\ \hline I & -A \end{array}\right)\begin{pmatrix} x \\ v \end{pmatrix} + \begin{pmatrix} B \\ \mathbf{0} \end{pmatrix} u \\ \hat{u} = \begin{pmatrix} \mathbf{0} \\ B \end{pmatrix}^T \begin{pmatrix} x \\ v \end{pmatrix}. \end{cases} \tag{12}$$

Above, the limitations caused by the model linearity are compensated by the dynamic nature of the model. Indeed, when the signals traverse back and forth

in the system *ad infinitum,* some functionality emerges that cannot be predicted in the component level. The model is *cybernetic:* Global high-level functionalities emerge from local interactions and feedbacks among components [12].

In the model (5), the interactions among components are packed into the matrices A and B. These matrices can be determined in a local way, that is, only state elements i and j are involved when the matrix element A_{ij} is being determined. This intuition of "localized PCA" can readily be implemented in a sensor network for implementing filtering of measurements, for example, as shown below.

5 Implementing a sensor network

Assume that the sensor network consists of n nodes that act as independent decentralized computing elements. Each node i, $1 \leq i \leq n$, is characterized by two state variables, x_i and v_i, for representing node activity and *normalized* node activity, respectively. Each of the nodes contains various sensors; it is assumed here that the delayed signals are regarded as separate sensor units. The set of sensors in node i will be denoted N_i. The global data structures can be reconstructed from the local ones as

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \quad v = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}, \quad \text{and} \quad u = \begin{pmatrix} u_{\iota \in N_1} \\ \hline \vdots \\ \hline u_{\iota \in N_n} \end{pmatrix}. \tag{13}$$

Each node i communicates with its immediate neighbors j, receiving x_j, v_j, and the *state corrections* δx_{ij}. Correspondingly, each node sends to its neighbors its own state, its normalized state (these are the same for all neighbors), and its contribution to changing the neighbors states, δx_{ji} (these are different for all neighbors). The "correction matrix" can be defined as

$$\begin{aligned} \left(\Delta x_1^{\text{out}} \cdots \Delta x_n^{\text{out}} \right)^T &= \left(\Delta x_1^{\text{in}} \cdots \Delta x_n^{\text{in}} \right) \\ &= \begin{pmatrix} \delta x_{11} & \cdots & \delta x_{n1} \\ \vdots & \ddots & \vdots \\ \delta x_{1n} & \cdots & \delta x_{nn} \end{pmatrix}. \end{aligned} \tag{14}$$

The covariance matrices are decomposed as

$$A = \begin{pmatrix} \left(A^T\right)_1^T \\ \hline \vdots \\ \hline \left(A^T\right)_n^T \end{pmatrix} \tag{15}$$

and

$$B = \left((B)_{\iota \in N_1} \middle| \cdots \middle| (B)_{\iota \in N_n} \right). \tag{16}$$

Note that the above notations are employed just to stick to the convention: Matrix indices refer to columns (or sets of columns), whereas vector indices refer

to vector elements directly. Using these notations, the algorithm for keeping the node states up to date can be written separately for each node:

$$\begin{cases} \dfrac{d}{dt}x_i = -\left(A^T\right)_i^T x + \displaystyle\sum_j \delta x_{ij} \\ \dfrac{d}{dt}v_i = -\left(A^T\right)_i^T v + x_i. \end{cases} \tag{17}$$

Here, the corrections are calculated as

$$\Delta x_i^{\text{out}} = (B)_{\iota\in N_i}\, u_{\iota\in N_i}, \tag{18}$$

and, finally, the filtered measurement estimate becomes

$$\hat{u}_{\iota\in N_i} = (B)_{\iota\in N_i}^T\, v. \tag{19}$$

The covariances are updated locally in each node as

$$\begin{cases} \dfrac{d}{dt}\left(A^T\right)_i = \lambda\left(x_i x - \left(A^T\right)_i\right) \\ \dfrac{d}{dt}\left(B\right)_{\iota\in N_i} = \lambda\left(x u_{\iota\in N_i}^T - \left(B\right)_{\iota\in N_i}\right), \end{cases} \tag{20}$$

where λ determines the adaptation rate. Above, the procedure is shown in continuous time; in practice, because of the limited bandwidths, the process has to be appropriately discretized. In any case, it is assumed that the inputs u vary considerably slower than x_i and v_i, and the covariances are adapted even slower. If this is not the case, if u changes fast, the system can be boosted by adding an additional factor γ:

$$\begin{cases} \dfrac{d}{dt}x_i = -\gamma\left(A^T\right)_i^T x + \gamma\displaystyle\sum_j \delta x_{ij} \\ \dfrac{d}{dt}v_i = -\gamma\left(A^T\right)_i^T v + \gamma x_i. \end{cases} \tag{21}$$

Of course, the bandwidth requirements are simultaneously increased.

6 Additional issues

The interpretation of the network behavior is straightforward as long as the network is dense, and the nodes are fully connected: In that case, the global analysis directly applies, meaning that principal component filtering is being carried out. Similarly, in a totally disconnected network, each node just calculates some kind of weighted average of its inputs.

However, the situation is completely different if it is only some nodes that cannot communicate with each other, so that some elements in the matrices A and B are fixed to zero. As presented in [11], *unsymmetric* connections, so that A is forced to be triangular, results in *self-organization:* Rather than finding only the principal subspace, the principal components themselves are found.

Assuming that the loss of contact is two-directional, so that the A matrix still remains symmetric — does some kind of self-organization take place in that case? The hypothesis here is that *yes, that will happen.* For example, assume that the nodes are located in a chain, so that only the nearest neighbors can communicate with each other. This means that in A there is a *band* along the diagonal. Those nodes that can better see their neighbors start searching for the direction of common variation, whereas the more blinded units concentrate more on their local data, explaining the variation that remains. In this sense, there will assumedly emerge some kind of *spatial organization* and *localization.* This will take place based on observed correlations in the measurements, without actual location information. This behavior is illustrated in the experiments.

If a sensor network scheme is implemented as proposed above, the operation starts from local measurements, $A = I$, because no information about the environment exists. As information from the neighbors cumulates, the filtered measurements can get more and more accurate, and the system gradually becomes *smarter.* Assuming that some of the sensors are located "up-stream", the changes in the "down-stream" direction can be anticipated. If the noise level is high, the down-stream sensors may begin to trust the global sensor state more. The up-stream sensors always have to trust only their measurements. It needs to be noted that it need not be the same quantities that are measured in all nodes; if there is correlation, this information can still be utilized regardless of the units.

Similarly — just trusting correlations, not trying to construct any causal models, turns out to be a rather wise strategy. One should not try to implement *too smart* devices. Conclusions and applications of the new functionalities are to be carried out by some outside expert having wider understanding: The Platonian problem, detecting the real underlying phenomena beneath the observations, cannot be solved by the new devices themselves, and expert knowledge is needed. Whereas observing signals is safe, trying to control them is hazardous.

Above it was assumed that the data has zero mean. If this is not the case, some preprocessing of the measurement data u_{meas}, and postprocessing of the results is needed. Perhaps the easiest way to condition the input data is the filter it using a high-pass filter:

$$G(s) = \frac{s}{1 + \tau s}, \tag{22}$$

so that $u(s) = G(s)u_{\text{meas}}(s)$. Here, denominator time constant τ is added to avoid excessive amplification of noise. In demanding environments, another possibility is to apply a *lead compensator*

$$G(s) = \frac{1 + \tau_1 s}{1 + \tau_2 s}, \tag{23}$$

where one has to select $\tau_1 \gg \tau_2$ to emphasize changes in signals. The model is also to be constructed for the filtered signals: To eliminate this differentiation from the final estimates, one has to integrate the results. However, to avoid "drifting" of the estimates, some kind of time constant is necessary here, too,

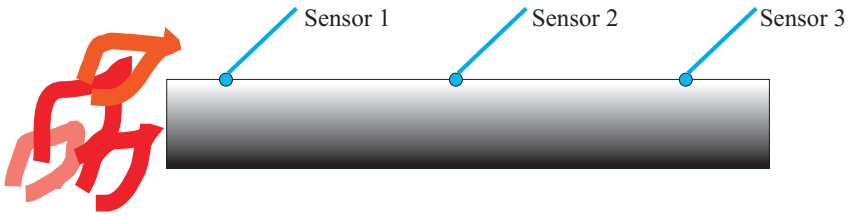

Fig. 1. Distributed parameter system: Heated rod

so that the biases finally decay, and asymptotically the estimate $\hat{u}_{\text{meas}}$ tries to reach the actual measurement u_{meas}:

$$\frac{d}{dt}\hat{u}_{\text{meas}} = \hat{u} + \epsilon \cdot (u_{\text{meas}} - \hat{u}_{\text{meas}}). \tag{24}$$

In practice, the variation levels of the measurement signals determine how "visible" they are when constructing PCA models. Typical approach is to normalize the measurements, so that they all have unit variance. Inversely, the estimates have to be rescaled back to original variation scale.

If one wants to estimate the validity of the model, for example, one can easily determine the *Hotelling's* T^2 statistic

$$T^2 = x^T \mathrm{E}\{xx^T\}^{-1} x = x^T v, \tag{25}$$

and the Q statistic

$$Q = (u - \hat{u})^T (u - \hat{u}). \tag{26}$$

These can be calculated node-wise, so that, for example, $T^2 = \sum_i T_i^2$, where $T_i^2 = x_i v_i$. These node-wise terms can be used for estimating the local match of the model against the incoming data: Note that T_i^2 has χ^2 distribution with a single degree of freedom.

7 Application example

As an application example, a heat diffusion process is studied, being a typical example of partial differential equation models. Assume that a rod is being heated from the other end (Fig. 1); the temperature is measured in three locations along the rod (considerable amount of measurement noise being added). In Fig. 2, the correlations between the measurements are shown. The correlations reveal that sensors 1, 2 and 3 are located in a chain, 1 being nearest and 3 being farthest from the heat source.

In each node, only the temperature is measured; the input vector consists of the current and two previous time step measurements, that is,

$$u_i(k) = \begin{pmatrix} T_i(k) \\ T_i(k-1) \\ T_i(k-2) \end{pmatrix}, \tag{27}$$

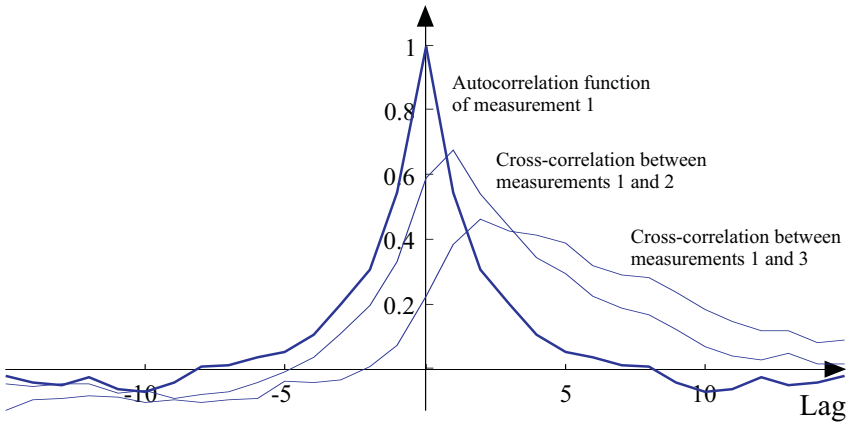

Fig. 2. Correlations between measurements in nodes

meaning that there are altogether 9 inputs in the global model. The algorithms were implemented in discrete rather than in continuous time (see [11]), that is, the signal bandwidth demand could be kept lower when data was transferred only at certain time instants (the local calculations in the nodes, or finding the steady state of x and v, was assumed to be, on the other hand, very fast, practically instantaneous).

In Fig. 3, the original (discrete-time) measurements in node 1 are shown, together with filtered estimates, for separate validation data having the same statistical properties as the training data. First, it is clear that the local model that only has information about the local environment in node 1, becomes a finite-impulse response filter utilizing the local measurements only: The filtered result is a weighted average of the past three measurements. This means that the result resembles traditional "dummy" filtering, so that the smoothened responses are reached with the cost of delayed and sluggish estimates if some change is taking place in the measurements. The global model, where all nodes are assumed to be connected to each other, and the chain model, where only the neighboring ones are connected, seem to be faster; indeed, it seems that no filtering takes place whatsoever. This is due to the structure of the measurements: Node 1 is the first one in the row, and no additional information can be received from the other nodes. However, note that this structural fact has not been given *a priori;* it is the observed correlations among the signals only that dictate the emerging filter structures.

In this example, no prescaling of signals was carried out; this means that the signals with high variation (for example, the temperature reading in node 1) are represented rather accurately in the estimates (but also the "noise fidelity" is high).

In Fig. 4, the corresponding signals in the last sensor are shown. First, it can be noted that because of the low-pass nature of the process, the noise is automatically dampened, but it seems that the filtering schemes that are based

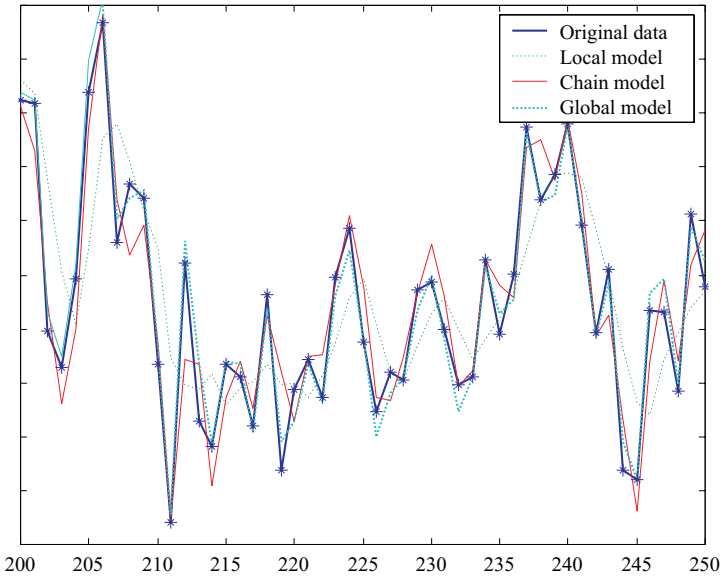

Fig. 3. Original and filtered measurements in the first node

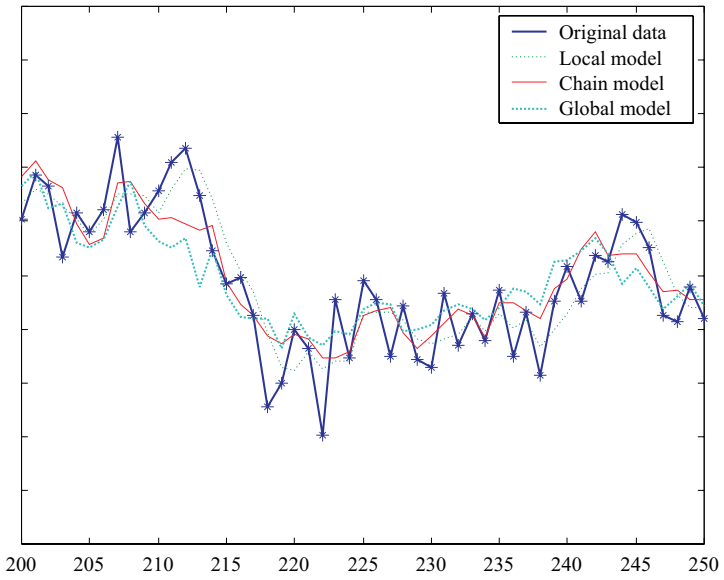

Fig. 4. Original and filtered measurements in the last node

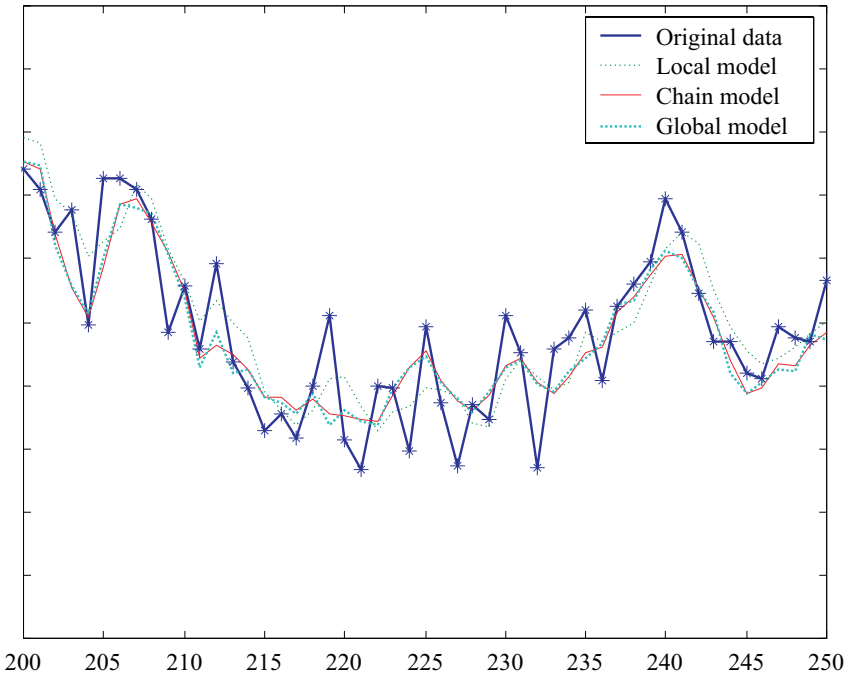

Fig. 5. Original and filtered measurements in the middle node

on interactions among nodes can further attenuate the disturbances. Whereas the local filter always remains somewhat slow, the more "intelligent" schemes can anticipate the future behaviors. Also the incomplete filtering scheme, where the nodes 1 and 3 are not connected, gives good filtering results.

Even though node 3 is the "last" node in the row, receiving the smoothest temperature readings, it seems that the filtering results in node 2 are even better (see Fig. 5) — indeed, this node receives information from both directions. What is more, it seems that the chain structure provides at least as accurate filtering as the global model does; it can even be claimed that the results are *more accurate* when there is less (irrelevant) information available. When only appropriate information is involved (data from the nearest neighbors only) the model contains less variables, becoming faster and more robust. The variation oriented latent variables (in the PCA sense) are *localized* in the sensor grid.

The sensors above were linear devices — one could assume that no new functionalities would pop up in such sensor systems. However, the truly distributed implementation offers real added value here: Remember that there were structural constraints imposed on the sensor connections (in the "chain" case). Whereas complete connections would result in the (trivial) PCA and PCR, in a network with incomplete connections something else emerges. In such a case there is no simple global cost criterion available, and the overall system properties cannot be analyzed easily that way. Still, it is intuitively clear that such

localized optimization is the smart way to carry out the measurement fusion task.

8 Discussion

The researach on distributed sensor networks is still mostly academic. Where is the added value, where could such *ad hoc* networks be truly used? Is the added value so great that the increased sensor complexity is motivated? It is difficult to answer this kind of questions; so many technical branches have evolved very fast after some threshold has been passed. As the prices come down, and as the cost of adding microprocessors to field devices becomes negligible, new functionalities can be included in the devices with practically no cost. If the sensors are connected to each other using radio frequency connection (using "bluetooth", for example), there would be the necessary infrastructure already available. If the theory of distributed systems becomes mature enough, the user does not need to know about the underlying functionalities. This is the key point: The new properties must not be a new burden to the user. Then, even minor enhancements in the device operation may pay back.

Whatever is the practical value of the proposed approach, there are theoretically interesting consequences: It can be claimed that this is the first consistent *quantitative* approach to capturing the coordination of distributed agents in complex networks. The traditional way to study such systems is based on software and purely qualitative techniques. If the toolboxes of mathematical analysis could efficiently be applied in such distributed environments, much more powerful design methodologies would be at hand. New mathematical theory of *sparse correlation matrices* and *distributed principal components* is needed.

References

1. K.J. Åström and B. Wittenmark: *Computer-Controlled Systems.* Prentice Hall, NJ (1997).
2. A.-L. Barabasi: *Linked: The New Science of Networks.* Perseus Books, Cambridge, MA (2002).
3. A. Basilevsky: *Statistical Factor Analysis and Related Methods.* John Wiley & Sons, New York, NY (1994).
4. K.I. Diamantaras and S.Y. Kung: *Principal Component Neural Networks: Theory and Applications.* Wiley, New York, NY (1996).
5. W. Favoreel, B. de Moor, and P. van Overschee: Subspace state space identification for industrial processes. *Journal of Process Control,* **10**, pp. 149–155 (2000).
6. S. Haykin: *Neural Networks — A Comprehensive Foundation.* Prentice–Hall, Upper Saddle River, NJ (1999).
7. D.O. Hebb: *The Organization of Behavior: A Neuropsychological Theory.* John Wiley & Sons, New York, NY (1949).
8. H. Hyötyniemi, P. Saariluoma: *Chess — Beyond the Rules.* In Timo Honkela, (ed.): *Games, Computers and People,* Finnish Artificial Intelligence Society, Helsinki, Finland, pp. 100–112 (1999).

9. H. Hyötyniemi: *Multivariate regression — techniques and tools.* Helsinki University of Technology, Control Engineering Laboratory, Report 125 (2001).
10. H. Hyötyniemi and J. Miettunen: *Towards Smart Devices: Enhancing Measurements by Automated Data Reconciliation.* Proceedings of IASTED Conference on Intelligent Systems and Control, Salzburg, Austria (2003).
11. H. Hyötyniemi: Hebbian and Anti-Hebbian Learning: System Theoretic Approach. Submitted to *Neural Networks* (2004).
12. H. Hyötyniemi: *Cybernetics — Towards a Unified Theory?* Submitted to *STeP 2004,* Vantaa, Finland (2004).
13. V. Lesser, C.L. Ortiz Jr., and M. Tambe (eds.): *Distributed Sensor Networks — A Multiagent Perspective.* Kluwer Academic Publishers, Boston, MA (2003).
14. P. van Overschee and B. de Moor: *Subspace Identification for Linear Systems: Theory — Implementation — Applications.* Kluwer Academic Publischers, Boston, MA (1996).

15. *Additional material will be available in public domain in near future at* `http://www.control.hut.fi/cybernetics`.