Computational Sensor Networks

Tom Henderson, Chris Sikorski
School of Computing
University of Utah
Salt Lake City, UT, USA
Email: tch@cs.utah.edu

Edward Grant, Kyle Luthy
Electrical and Computer Engineering
North Carolina State University
Raleigh, NC, USA
Email: egrant@ncsu.edu

***Abstract*— We propose *Computational Sensor Networks* as a methodology to exploit models of physical phenomena in order to better understand the structure of the sensor network. To do so, it is necessary to relate changes in the sensed variables (e.g., temperature) to the aspect of interest in the sensor network (e.g., sensor node position, sensor bias, etc.), and to develop a computational method for its solution. As an example, we describe the use of the heat equation to solve the sensor localization problem. Simulation and physical experiments are described.**

I. Introduction

A model-based approach to the design and implementation of *Computational Sensor Networks* (CSNs) is proposed. This high-level paradigm for the development and application of sensor device networks provides a strong scientific computing foundation, as well as the basis for robust software engineering practice. The three major components of this approach include (1) models of phenomena to be monitored, (2) models of sensors and actuators, and (3) models of the sensor network computation. We propose guiding principles to identify the state or structure of the phenomenon being sensed, or of the sensor network itself. This is called *computational modeling*. These methods are then incorporated into the operational system of the sensor network and adapted to system performance requirements to produce a mapping of the computation onto the system architecture. This is called *real-time computational mapping* and allows modification of system parameters according to real-time performance measures. This paper deals mainly with computational modeling.

CSNs represent a scientific computing approach, and this includes the Verification and Validation (V & V) methodology of that discipline[19]; that is, model implementations must be verified (e.g., for correctness or numerical properties like error and convergence), and appropriate tests embedded in the system to monitor system correctness during execution. However, an important new aspect of this approach is that a CSN has the ability to sense and interact with the environment, and thus can run its own validation experiments to confirm or refute model structure or parameter values. Another intrinsic capability offered by CSNs is that models can be used to determine unknown aspects of the structure of the measurement system itself given a known state of the physical phenomenon. For example, given the heat flow PDE and known temperatures at fixed (but unknown) sensor node locations, the equations can be reworked so as to determine the sensor locations (i.e., to solve the sensor localization problem). This can be done for a wide variety of initial conditions and depends only on the equations defining the physical process and the specific realization of the process in the world. Thus, real-time V & V are performed and this permits a scientifically repeatable basis for sensor network experiments. Real-time computational steering is achieved by (1) embedding verification and validation modules into the executable code, and (2) modeling module performance in terms of statistically meaningful characterization of output features conceptually defined by the user.

On the sensor network side, many advances have been made in sensor network technology and algorithms in the last few years. See [25] for an overview of the state of the art. Work has been done on: architecture [15], systems and security [24], and applications. Our own work has focused on the creation of an information field useful to mobile agents, human or machine, that accomplish tasks based on the information provided by the sensor network [2], [3], [4], [9], [10], [11], [12], [13], [14]. In order to address sensor networks in a comprehensive manner, the sensor network community has initiated a research program[16] that includes work in the areas of sensor network architectures, programming systems, reference implementations, hardware

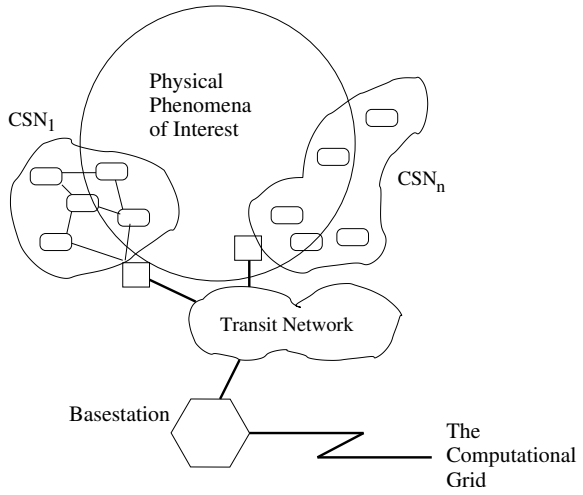

Fig. 1. *Computational Sensor Network* Large-Scale Utilization Paradigm.

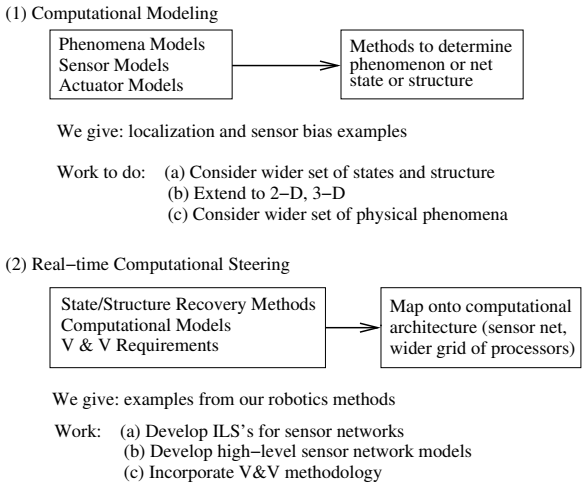

Fig. 2. *Computational Sensor Network* System Development Framework.

and software platforms, testbeds and applications. Here we **explore the impact of a computational science approach on all these aspects of sensor networks**, and show that much benefit can be derived [7], [8]; in particular, the tools developed here can be highly leveraged across many scientific communities. CSNs will provide software engineering support for scientists and engineers to exploit sensor networks where it is notoriously difficult to develop and validate systems, for example, in our proposed snow monitoring application.

II. Introduction

The *Computational Sensor Network* (CSN) paradigm is displayed in Figure 1. Physical phenomena of interest are monitored by a set of CSNs, each with its own models. CSN_i produces its results (as specified by the requirements) which are passed along to other CSNs as well as to the general computational grid. These results may provide information for observers, decision makers, or may provide dynamic data for large-scale, multi-physics simulations. Figure 2 gives our vision of the two major issues addressed by the CSN system development framework:

1) **Computational Modeling**: It is necessary to develop a framework within which it is possible to define models of physical phenomena of interest, as well as sensors and actuators, and to produce computational methods to determine state or structure of either the monitored system or the sensor network itself.
2) **Real-time Computational Steering**: Given a method developed in (1), it is necessary to combine it with a conceptual model of the sensor network, and a set of verification and validation requirements to produce a set of executable tasks which can be mapped onto the sensor network architecture as well as a wider computational grid and provide a high-level interface for human understanding.

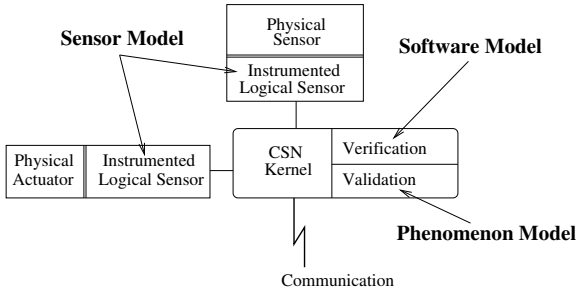

Fig. 3. Basic *Computational Sensor Network* Layout.

The layout of an individual CSN is shown in Figure 3. CSNs provide a sensor network programming paradigm built from a combination of (1) scientific computing practice, and (2) the Instrumented Logical Sensor methodology[6]. This combination permits the construction of qualitatively different applications by incorporation of the specific models for the phenomena being monitored, the sensors and actuators deployed, and the software requirements imposed.

III. Computational Modeling

One of the major innovations of this approach is the incorporation of a strong model of the phenomenon to be observed. This allows the system developer great insight into the V & V requirements. We demonstrate

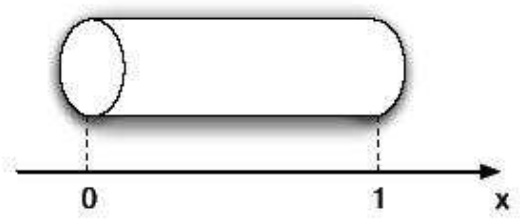

Fig. 4. Heat Flow in a Uniform Rod.

the usefulness of the CSN approach by way of an example:

- **Sensor Node Localization**: Given a strong model of the physical phenomenon, and a set of sensor nodes in unknown, but fixed, locations, use the computational model to determine the sensor node locations.

A. Sensor Node Localization

To demonstrate how this methodology can be applied, we show how the sensor node localization problem can be solved. Oftentimes sensor devices are dropped at random into an environment or maybe moved (e.g., in a snow monitoring application, the devices may move with the snow both in depth as well as tangential to the surface). Many approaches to sensor node localization have been proposed [5], [17], [18]; see [22] for a survey. As one example, Whitehouse and Culler propose a macro-calibration method for localization [23]. Their ad hoc localization system estimates distance between nodes using received signal strength information and acoustic time of flight. Although these phenomena can be modeled in the CSN context, their approach requires additional sensors (microphones) and processes. Moreover, CSNs solve an inverse problem based on the physical phenomenon - the example given in this paper uses the heat equation (note that temperature sensors are ubiquitous and the method is robust).

Consider a rod of uniform cross-section and length 1 that is completely isolated except at the ends (see Figure 4). The heat flow is therefore limited to the x direction and the development of the temperature y over time can be described by the following partial differential equation (known as the *diffusion equation*):

$$\frac{\partial y}{\partial t} = D \cdot \frac{\partial^2 y}{\partial x^2} \; with \; D = \frac{\kappa}{c \cdot \rho}$$

where κ denotes the thermal conductivity, c the specific heat capacity and ρ the density of the rod. Figure 5

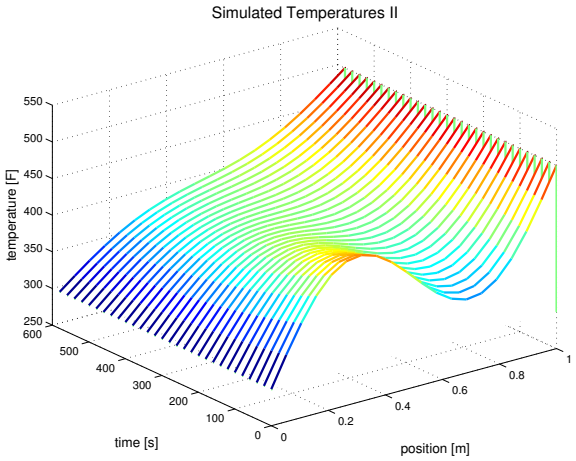

Fig. 5. Simulation of Heat Flow Equation.

shows how the temperature changes over time for an arbitrary initial state. [Note that usually the temperatures at the ends are fixed and the temperatures settle to a linear ramp (one could easily assign locations to the nodes given a temperature then); however, the basic requirement is that the temperature values in the rod change according to the heat equation in order for the method to work. It is also possible to allow the ends to vary. Also, there exist temperature distributions which are ambiguous, and thus where the method will not work – e.g., a constant temperature across the whole rod.]

Such PDE's are usually solved by discretization and approximation of the derivatives. Then the temporal variation of the rod at any location can be determined using the standard finite difference approach: a grid of discrete, general points over the domain is considered and the derivatives are replaced by their finite-difference expressions at those points. We denote the points along the x-axis by x_i, the time points by t_j (with Δt the time step) and finally the temperature at point x_i and time t_j by $y_{i,j}$. Then:

$$\left(\frac{\partial y}{\partial t}\right)_{i,j} = \frac{y_{i,j+1} - y_{i,j}}{\Delta t}$$

$$\left(\frac{\partial^2 y}{\partial x^2}\right)_{i,j} = \frac{\frac{y_{i+1,j}-y_{i,j}}{x_{i+1}-x_i} - \frac{y_{i,j}-y_{i-1,j}}{x_i-x_{i-1}}}{\frac{1}{2}(x_{i+1} - x_{i-1})}$$

which yields:

$$y_{i,j+1} = y_{i,j} + \frac{2\Delta t D}{(x_{i+1} - x_{i-1})}\left(\frac{y_{i+1,j}}{(x_{i+1} - x_i)} - \frac{y_{i,j}}{(x_{i+1} - x_i)} - \frac{y_{i,j}}{(x_i - x_{i-1})} + \frac{y_{i-1,j}}{(x_i - x_{i-1})}\right)$$

To solve the localization problem in this case, the set of equations (one for each y_i) must be solved for the x_i values. This requires solving a set of degree 3 polynomial equations - which can be a difficult problem. For example, given n sensor nodes, there are up to 3^n distinct solutions (most are complex solutions, and thus not feasible). See [21] for analytical solution methods, e.g., homotopy continuations. We do not pursue such methods here since we have discovered that in the case of sensor networks, a search over uniform samples can be performed which produces the sensor node locations quite efficiently. Consider *Algorithm Heat_1D*.

Algorithm Heat_1D

On input:
n: the number of sensor nodes
$T_n^{(j)}$: the temperature at node n at time j
x_0, T_0: min x value and temperature there
x_{n+1}, T_{n+1}: max x value and temperature there
On output:
S_i, $i = 1 \ldots n$; sensor node locations
begin
$num\_samples \leftarrow 100$ – number of location guesses
$S_{ij} \leftarrow U(x_0, x_{n+1})$ – uniform samples for locations
$T'^{(k)}_{s_{ij}} \leftarrow Heat\_1D\_Sim$ – predicted temps for x_i
$D_i \leftarrow \|T'_i - T\|$ – distance from actual temps
$D\_min = min(D_i)$ – best temperature match
while $D_min >$ *thresh*
$S_{1\ldots10} \leftarrow$ choose best guesses
$S_{11\ldots100} \leftarrow$ add more random samples
$T'^{(k)}_{s_{ij}} \leftarrow Heat\_1D\_Sim$ – predicted temps for x_i
$D_i \leftarrow \|T'_i - T\|$
$D\_min = min(D_i)$
end
end

The algorithm generates random locations for the sensors, then simulates the heat equation to obtain predicted temperatures given the assumed node locations, then uses a distance norm to obtain an estimate of how much the actual and predicted temperature values differ. Finally, it determines the minimum error guess. If the best estimate is within a certain threshold, then the algorithm returns the locations that best fit the actual data. Otherwise, new guesses are generated from perturbations of the best guesses, and random samples added for the rest, and the process continues.

The results of *Algorithm Heat_1D* are shown in Figures 6- 7. The performance of the algorithm is given

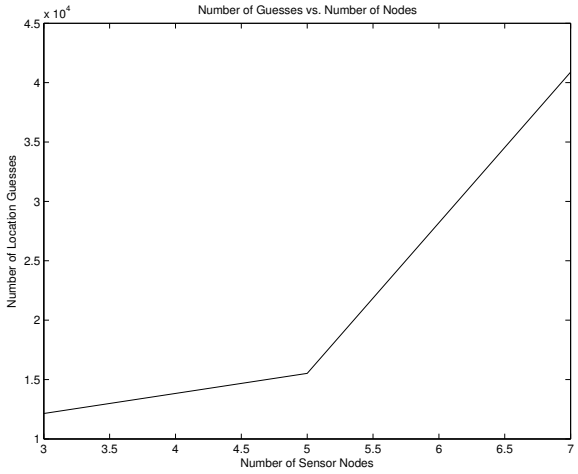

Fig. 6. Number of Locations Expanded by *Algorithm Heat_1D*.

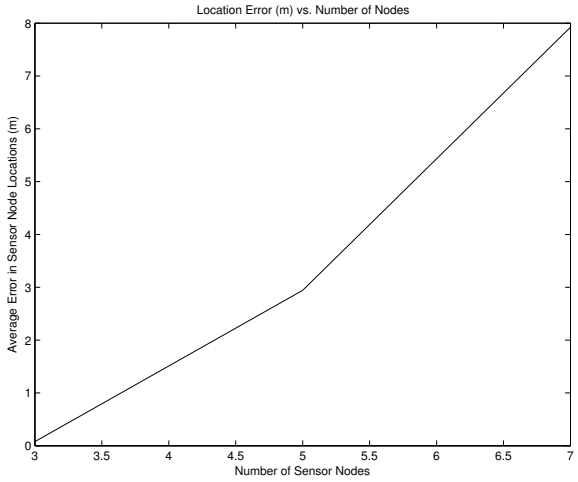

Fig. 7. Error (m) in Solution Location by *Algorithm Heat_1D*.

in terms of number of guesses produced versus number of nodes and error in the sensor locations found. Several sets of node locations were used in this simulation experiment, and there is no assumption on the order of the nodes along the rod. The algorithm has also been implemented and run on the Tmote Sky, and runs quite efficiently; however, it has not been used in physical experiments yet. [Note that in order to get $O(\Delta t + (\Delta x)^2)$ error in approximation, we must select Δt sufficiently small to guarantee stability; i.e., $\Delta t \leq \frac{(\Delta x)^2}{2}$. Sparse sampling on the interval may result in a relatively large Δx.]

We have also applied the method to data taken from an experimental apparatus (Figure 8 shows the layout). A one meter long stainless steel rod (304CG) of diameter one inch is connected to a steam chamber at one end and is instrumented with type T thermocouples located

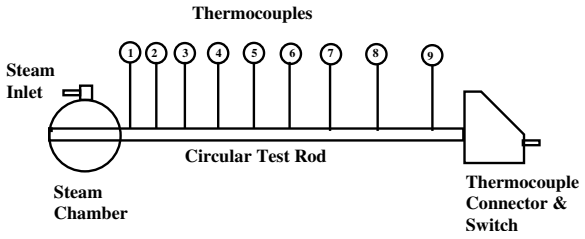

Fig. 8. Layout of the Heated Rod Experiment.

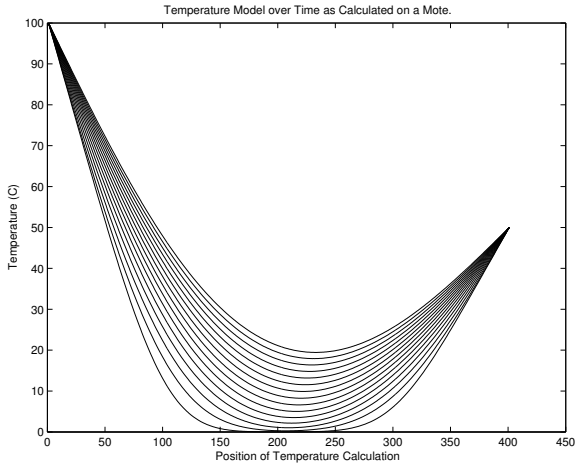

Fig. 9. Forward Temperature Simulation from Tmote Sky Execution.

at 0.005m, 0.035m, and 0.095m, respectively, from the steam chamber. The thermocouples are connected to 10 channel selector switches which in turn are connected to a digital readout. The rod is attached to the steam chamber that provides a constant energy source at the base. The steam is turned on, and temperature readings are taken every 20 seconds as the rod heats.

In analyzing this data, an alternate algorithm was developed. Given knowledge of the initial conditions once the steam is activated (namely, 100 degrees C at one end and room temperature elsewhere along the rod), it is possible to run a careful simulation to obtain temperature curves at a dense sample of points along the rod (e.g., 1,000). Code was developed for the Tmote Sky and Figure 9 shows the results of a mote calculation. Each sensor is then matched independently to determine the best fit location. Call this *Heat_1D_dense*.

$x = 0.005$ sim/measured	$x = 0.035$ sim/measured	$x = 0.095$ sim/measured
65.2/65.2	30.2/30.2	20.6/20.6
85.4/68.3	33.9/33.1	21.2/20.7
88.2/71.0	35.7/35.5	21.4/21.0
89.5/73.4	37.6/37.6	21.6/21.1
90.4/75.9	39.5/39.8	21.8/21.3
91.1/77.9	41.4/41.9	21.9/21.6
91.6/79.8	43.2/43.9	22.1/22.0

Table 4.1 Simulated and Measured Temperature Data for Heated Rod Experiment.

Table 4.1 gives the simulated and measured temperature values. The actual and recovered locations are: $(0.005, 0.035, 0.095)$ and $(0.011, 0.035, 0.100)$, respectively. As can be seen, the heat transfer model fits better away from the steam source.

Discussion and Future Work

Several issues arise in terms of the application of this method. First, there is a tradeoff between coarse simulation (e.g., large spacing and time step) versus high resolution simulation. Next, one approach is to compute a sparse solution using only the points where data is taken (or assumed taken). This involves running the coarse simulation for each separate guess as to the locations of the sensors. Alternatively, it is possible to run one fine-grained simulation, if the initial conditions are known and satisfied, and then simply match the sensor data to the location with the temperature trace which most closely matches the measured data. Note that even if the data is not taken from time 0 (i.e., when the steam is turned on), it is possible to find the best matching locations for the measured data considered simultaneously.

Another major issue is the determination of an adequate model of the phenomenon. We take as our starting point that this is possible, especially when the structure of the model is known, and all that remains is to identify parameters. For recent work on this, see [20]. They derive the system model and the measurement model by the finite spectral method and show how nonlinear phenomena with complex boundary conditions can be reconstructed and predicted. More work needs to be done to characterize the types of functions which allow unique solutions in these circumstances.

These preliminary results are very encouraging. However, there is much work to be done:

1) Analysis: The mathematical basis for the approach must be established. This is an instance of the Inverse Heat Transfer Problem [1]. We would

like to couple the Heat_1D method with some nonlinear solvers (e.g., Newton type methods). In this way Heat_1D would provide starting points for faster nonlinear solvers.

2) Stochastic Methods: Simple Monte Carlo techniques are used here; however, we intend to explore Markov Chain Monte Carlo, Bayesian methods, Quasi-Monte Carlo, etc.
3) Experimental Considerations: Further experimentation needs to be performed to establish the method. We plan to build and test an experimental apparatus for monitoring snow. 2D and 3D methods must be developed.

A few words are in order about the practicability of the method:

- Each sensor node can solve the problem independently in terms of sensor data from its neighbors.
- Generally, there will not be a large number of neighbors, and thus the system should be readily solvable.
- The only communication required is a time sequence sample of temperatures from the neighbors.
- Solutions can be shared between nodes to improve efficiency and accuracy.
- Temperature values can be averaged to reduce the effect of noise.
- Off-network computation of the numerical solution is also possible.

Acknowledgments

We would like to thank Mr. Konark Pakkala and Prof. Kent S. Udell for performing the heated rod experiment and collecting the data, and Mr. Dietrich Brunn and Prof. Uwe Hanebeck for discussions on this topic.

References

[1] O.M. Alifanov. *Inverse Heat Transfer Problems*. Springer Verlag, Berlin, 1994.

[2] G.J. Barlow, T.C. Henderson, A.L. Nelson, and E. Grant. Dynamic leadership protocol for s-nets. In *IEEE Conference on Robotics and Automation*, New Orleans, LA, April 2004.

[3] Y. Chen and Thomas C. Henderson. S-nets: Smart sensor networks. In *Proc International Symp on Experimental Robotics*, pages 85–94, Hawaii, Dec 2000.

[4] Yu Chen. Snets: Smart sensor networks. Master's thesis, University of Utah, Salt Lake City, Utah, December 2000.

[5] P. Corke, R. Peterson, and D. Rus. Localization and navigation assisted by cooperating networked sensors and robots. *International Journal of Robotics Research*, 24(9), September 2005.

[6] Mohamed Dekhil and Thomas C. Henderson. Instrumented logical sensors systems. *International Journal of Robotics Research*, 17(4):402–417, 1998.

[7] T.C. Henderson. Verification and validation of sensor networks. In *Schloss Dagstuhl Workshop on Form and Content of Sensor Networks*, September 2005.

[8] T.C. Henderson. Performance measures for sensor networks. In *NIST-ANS Workshop on Urban Search and Rescue Performance Measures for Intelligent Systems*, February 2006.

[9] T.C. Henderson and E. Grant. Gradient calculation in sensor networks. In *IEEE Conference on Intelligent Robots and Systems*, Sendai, Japan, October 2004.

[10] T.C. Henderson, E. Grant, K. Luthy, and J. Cintron. Snow monitoring with sensor networks. In *IEEE Workshop on Embedded Networked Sensors*, pages 558–559, Tampa, FL, November 2004.

[11] T.C. Henderson, R. Venkataraman, and G. Choikim. Reaction-diffusion patterns in smart sensor networks. In *IEEE Conference on Robotics and Automation*, New Orleans, LA, April 2004.

[12] Thomas C. Henderson. Leadership protocol for s-nets. In *Proc Multisensor Fusion and Integration*, pages 289–292, Baden-Baden, Germany, August 2001.

[13] Thomas C. Henderson, Mohamed Dekhil, Scott Morris, Yu Chen, and William B. Thompson. Smart sensor snow. *IEEE Conference on Intelligent Robots and Intelligent Systems*, October 1998.

[14] Thomas C. Henderson, Jong-Chun Park, Nate Smith, and Richard Wright. From motes to java stamps: Smart sensor network testbeds. In *Proc of IROS 2003*, Las Vegas, NV, October 2003. IEEE.

[15] J. Hill and D. Culler. A wireless embedded sensor architecture for system-level optimization. Ece, UC Berkeley, October 2002.

[16] <http://www.eecs.harvard.edu/noss>. NSF NOSS Workshop, Harvard University, Cambridge, MA. October 2005.

[17] J. T.-H. Lo and Jr. S.L. Marple. Observability conditions for multiple signal direction finding and array sensor localization. *IEEE-T Signal Processing*, 40(11):2641–2650, 1992.

[18] B.C. Ng and A. Nehorai. Active array sensor localization. *IEEE-T Signal Processing*, 44:309–327, 1995.

[19] W.L. Oberkampf, T.G. Trucano, and C. Hirsch. Verification, validation and predictive capability in computational engineering and physics. In *Foundations for Verification and Validation in the 21st Century Workshop*, Laurel, Maryland, October 2002.

[20] F. Sawo, K. Roberts, and U. D. Hanebeck. Bayesian estimation of distributed phenomena using discretized representations of partial differential equations. In *Proceedings Proceedings of the 3rd International Conference on Informatics in Control, Automation and Robotics (ICINCO 2006)*, pages 16–23, Setubal, Portugal, August 2006.

[21] A.J. Sommese and II C. Wampler. *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science*. World Scientific, New York, 2005.

[22] K. Whitehouse. The design of calamari: an ad-hoc localization system for sensor networks. Master's thesis, University of California, Berkeley, San Francisco, CA, 2002.

[23] K. Whitehouse and D. Culler. Macro-calibration in sensor/actuator networks. In *Proceedings of Mobile Networks and Applications*, volume 8, pages 463–472, 2003.

[24] L Zhang. Simple protocols, complex behavior. In *Proc. IPAM Large-Scale Communication Networks Workshop*, March 2002.

[25] F. Zhao and L. Guibas. *Wireless Sensor Networks*. Elsevier, Amsterdam, 2004.