Computational Sensor Networks

***Thomas C. Henderson*[1], *Christopher Sikorski*[1],**
***Edward Grant*[2] and *Kyle Luthy*[2]**

UUCS-07-003

[1]School of Computing
University of Utah
Salt Lake City, UT 84112 USA

[2] Dept of Electrical and Computer Engineering
North Carolina State University
Raleigh, NC 27695-7911

February 5, 2007

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 examples, we describe the use of the heat equation to solve (1) the sensor localization problem, and (2) the sensor bias problem. Simulation and physical experiments are described.

1 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[28]; 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 [39, 40] for an overview of the state of the art. Work has been done on: architecture [20], systems and security [9, 29, 38], and applications [26]. 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 [3, 5, 4, 12, 16, 13, 14, 15, 18]. In order to address sensor networks in a comprehensive manner, the sensor network community has initiated a research program[22] that includes work in the areas of sensor network architectures, programming systems, reference implementations, hardware 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 [10, 11]; in particular, the tools developed here can be highly leveraged across many scientific communities. CSNs will pro-

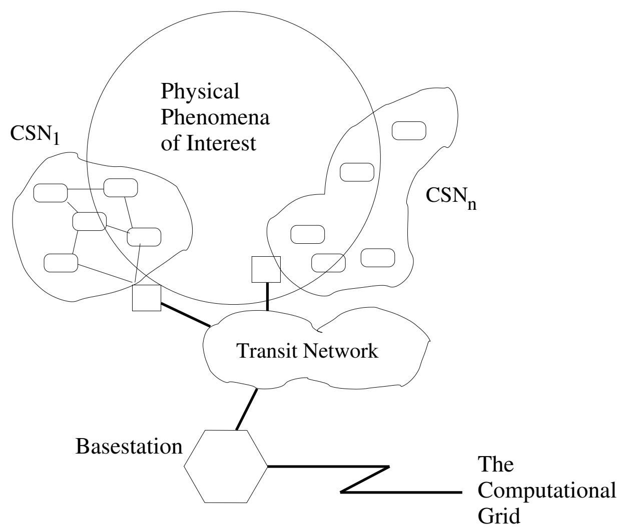

Figure 1: *Computational Sensor Network* Large-Scale Utilization Paradigm.

vide 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.

2 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.

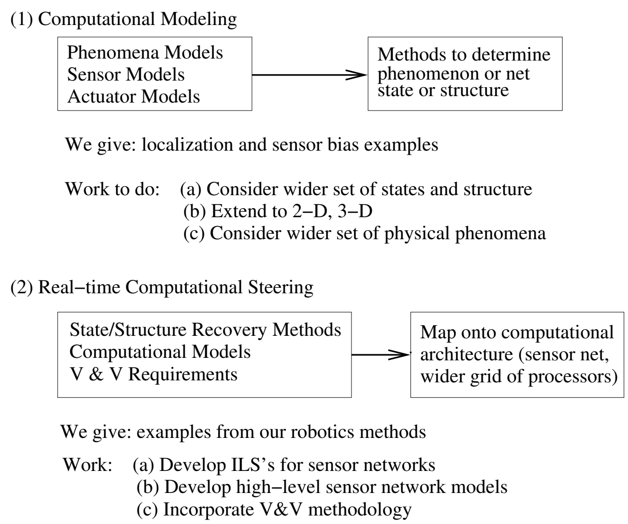

Figure 2: *Computational Sensor Network* System Development Framework.

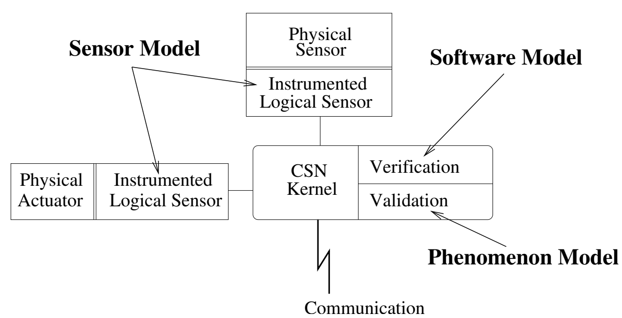

Figure 3: Basic *Computational Sensor Network* Layout.

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 (e.g., see [23]), and (2) the Instrumented Logical Sensor methodology[7]. 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.

3 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 the usefulness of the CSN approach by way of two examples:

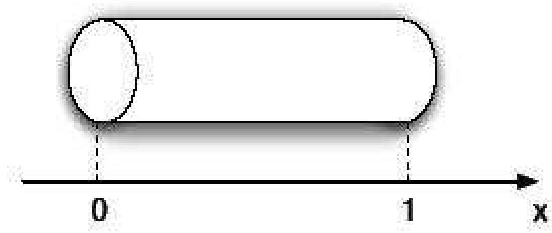

Figure 4: Heat Flow in a Uniform Rod.

- **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.
- **Sensor Bias Check**: Given a sensor model that includes a term for sensor bias, use the computational model to determine the amount of bias in each sensor.

3.1 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 [6, 24, 27]; see [35] for a survey. As one example, Whitehouse and Culler propose a macro-calibration method for localization [36]. 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*[37]):

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

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

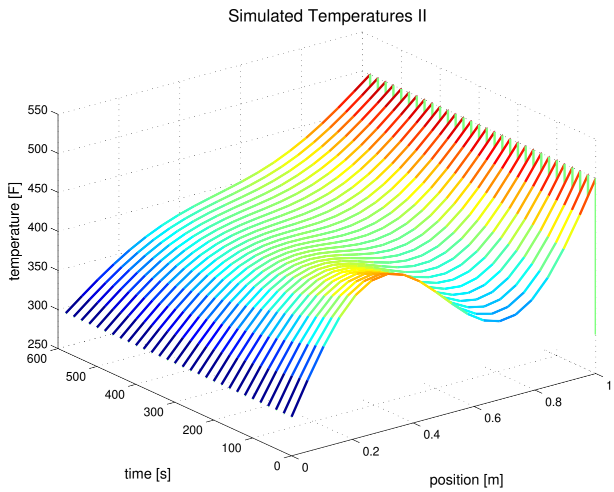

Figure 5: Simulation of Heat Flow Equation.

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 [2, 31] 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, $i = 1 \dots n$; 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 $D_min > thresh$
    $S_{1\dots10} \leftarrow$ choose best guesses
    $S_{11\dots100} \leftarrow$ 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 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

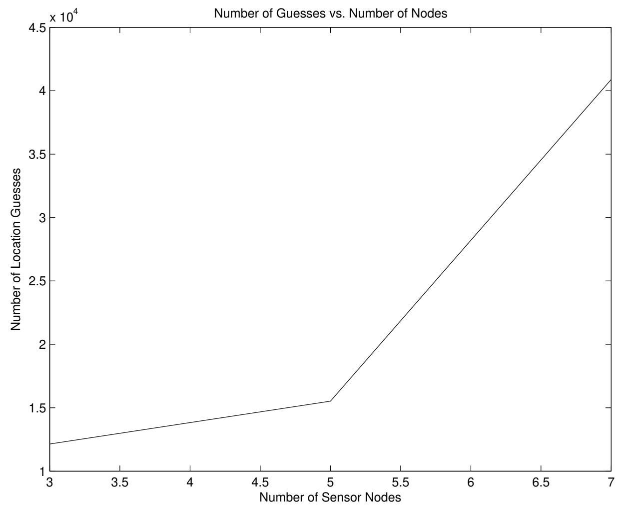

Figure 6: Number of Locations Expanded by *Algorithm Heat_1D*.

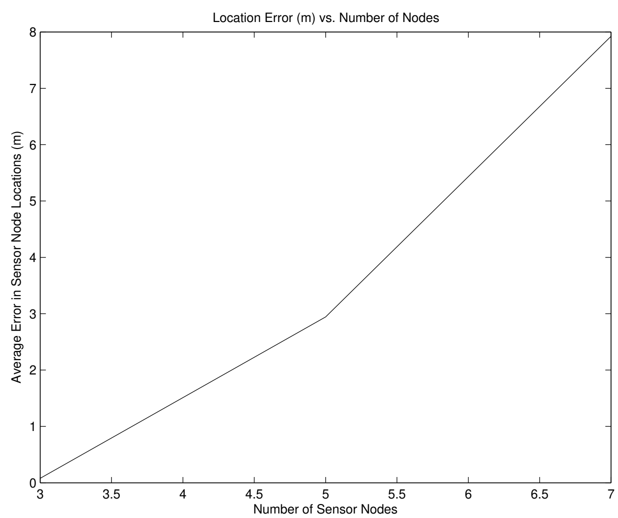

Figure 7: Error (m) in Solution Location by *Algorithm Heat_1D*.

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 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

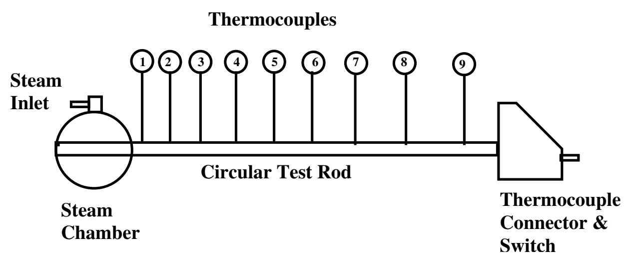

Figure 8: Layout of the Heated Rod Experiment.

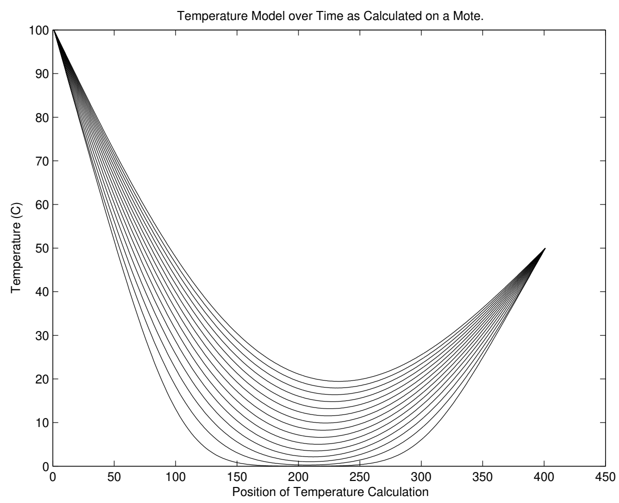

Figure 9: Forward Temperature Simulation from Tmote Sky Execution.

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

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 [30]. 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.

3.2 Sensor Networks and Bias Estimation

As another example of the computational sensor network approach, consider sensor bias estimation. Now assume that there are $n + 1$ sensors $S_i, i = 0, \ldots, n$ equi-spaced along the rod with S_i at position x_i ($x_0 = 0, x_n = 1$). Those sensors supply us with measurements of the actual temperatures where $z_{i,j}$ denotes the measurement of sensor S_i at time t_j. The measurements themselves are noisy and furthermore we assume that each sensor S_i has a bias of b_i:

$$z_{i,j} = y_{i,j} + b_i + v \qquad v \sim \mathcal{N}(0, \sigma^2)$$

Our goal is to estimate values for the b_i based solely on the sensor measurements and the system model. To do so we rewrite the last equation to

$$y_{i,j} \approx z_{i,j} - b_i$$

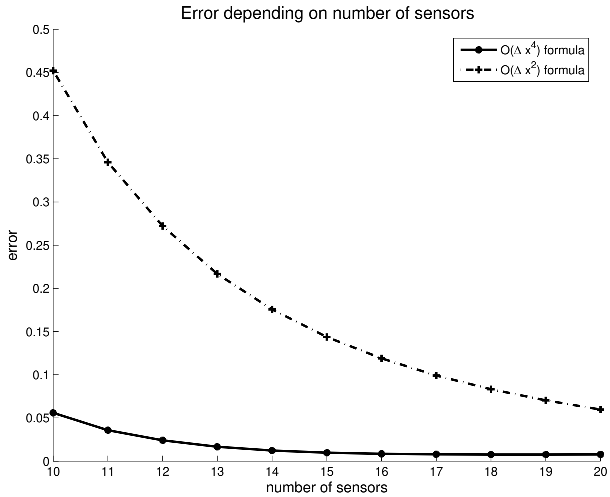

Figure 10: Errors Depending on Δx and the Approximation Formulas.

and insert this one into the finite difference approximation of the system model:

$$\begin{aligned}\frac{z_{i,j+1}-z_{i,j}}{\Delta t} &= \frac{D}{(\Delta x)^2}(z_{i+1,j}-b_{i+1} \\ &\quad +z_{i-1,j}-b_{i-1}-2z_{i,j}+2b_i) \\ \Leftrightarrow\ -b_{i-1}+2b_i-b_{i+1} &= \\ \underbrace{\frac{(\Delta x)^2}{D\cdot\Delta t}(z_{i,j+1}-z_{i,j})-z_{i+1,j}-z_{i-1,j}+2z_{i,j}}_{=:r_i}\end{aligned}$$

Since we know the correct temperatures for $x_0 = 0$ and $x_n = 1$ we can set $b_0 = b_n = 0$. This leads to the following tridiagonal system (since the rhs of the above equation is known):

$$\begin{pmatrix} -2 & 1 & & \\ 1 & \ddots & & \\ & & \ddots & 1 \\ & & 1 & -2 \end{pmatrix} \cdot \begin{pmatrix} b_1 \\ \vdots \\ b_{n-1} \end{pmatrix} = \begin{pmatrix} r_1 \\ \vdots \\ r_{n-1} \end{pmatrix} \tag{v}$$

which can be solved with direct methods (or in case of over-determined systems, can be solved using least squares).

We carried out some simulation experiments to see how our approach for bias detection performs. We used the same settings as before and simulated temperatures using spectral methods to generate the data for the experiments. The sensor located at the center of the rod was assigned a bias of $b_{n/2} = 5$ while all other sensors had no bias. There are two sources of error to the bias estimate: inaccuracies introduced by the finite difference approximations and noise. Figure 10 demonstrates the influence of the former. The figure compares true and estimated bias values; $\Delta t = 2s$ was

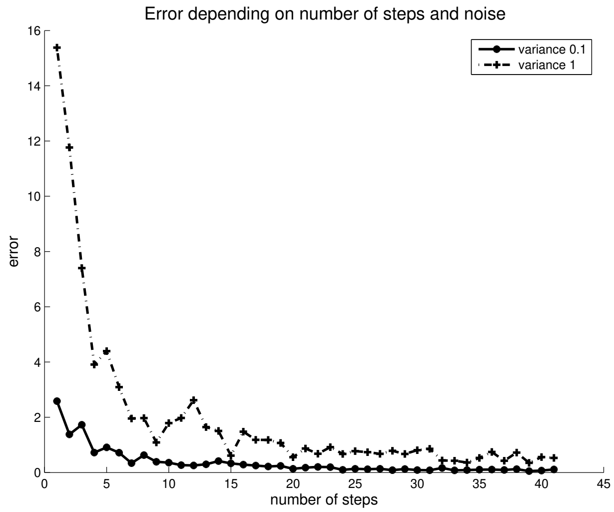

Figure 11: Estimates Depending on Noise and the Number of Measurements.

chosen as time step. For the estimation we used the approximation formula already introduced (error is $\in O((\Delta x)^2)$) as well as the following approximation formula which has error $\in O((\Delta x)^4)$:

$$\left(\frac{\partial^2 y}{\partial x^2}\right)_{i,j} = \frac{1}{12(\Delta x)^2}(-y_{i-2,j} + 16y_{i-1,j} + 30y_{i,j} + 16y_{i+1,j} - y_{i+2,j})$$

To apply this formula one has to assume that the two first and last bias values are known in advance (we set them to zero). As expected the higher accuracy of the latter formula also leads to better values for the b_i.

To demonstrate the influence of noise, we added white Gaussian noise to the (simulated) measurements. Now a single estimate is not sufficient any more since it is influenced by this noise. Therefore it becomes necessary to take into account more than just two consecutive measurements. This can be accomplished by using the equation several times for different measurements and solving this now over-determined system, or by solving the individual equations independently and computing the mean value. For different levels of noise Figure 11 shows how the bias estimate converges as more and more measurements are included (we used the same initial temperatures and bias values as above, 15 sensors and the $O((\Delta x)^4)$ finite-differences approximation).

So far we have done all of our sensor bias experiments with equi-spaced sensor nodes; this is a constraint that will rarely hold in real world applications and for unequally spaced sensors other approximations formulas have to be used, like derivatives of interpolating polynomials, for example (note that the resulting terms will still be linear in the b_i).

The technique presented here also has another drawback. The computation is performed globally so that the measurements of all sensors first have to be sent to a central node before computation can start. But consider the following scenario: the sensor network could frequently test itself by computing the bias of a *single* sensor under the assumption that the neighboring sensors have zero bias (this computation can be done locally since it involves only measurements from a constant number of sensor-nodes – 3 sensors for the $O((\Delta x)^2))$-approximation and 5 sensors for the $O((\Delta x)^4))$-approximation). Although the condition of zero-biased neighboring nodes may not hold, the sensor-network should still be able to detect malfunctioning nodes this way and can exclude those from further computations.

We have demonstrated the ideas already presented for a concrete example: we estimated sensor bias solely based on sensor measurements and the system model. The accuracy of the approximation formulas had a clear impact on the reliability and the convergence behavior of the estimates – but more accurate approximations will also result in higher computation costs. We furthermore discussed that this approach may still be of practical use even if the costs of global computation are not acceptable.

4 Real-Time Computational Steering

Given a computational method, such as one of those described previously, the next major goal is to develop a framework to facilitate mapping it onto a sensor network architecture. A strong requirement is to build on existing architectures (e.g., SNA, Tenet, etc.) and to provide value added for information analysis methods (e.g., COMPASS and WaveScope). Our approach to this is to extend our previous logical sensor system specification methodology (ILSS) into the CSN arena.

Recall that the SNA group has stated [21] that an architecture is not an end in itself, but that the task at hand is "the dense monitoring and analysis of sizable extents of the physical world," and that "sensor networks allow the understanding of the systematic behavior of various phenomena." The CSN approach allows a more quantified understanding of physical phenomena and the sensor network, too.

The most relevant aspect of the SNA architecture to this proposal is the programming architecture which is composed of two aspects:

- *internal programming interface (IPI)*: how processing is performed across the numerous nodes of the sensor network, and
- *external programming interface (EPI)*: how information extracted from the sensor network is processed on conventional computing resources.

We focus here on the IPI. CSN's require an organization of the task structure, as well as load balancing, communication minimization, etc. We have a great deal of experience with this in large-scale, multi-physics simulation [25], and propose to develop a similar approach for CSN's.

Certain aspects of CSN's require more detailed exchange of information through the interface: (1) error analysis, (2) physical phenomena models or computational methods, and (3) V & V requirements. Our particular approach to these interface problems is to exploit our previous work on Instrumented Logical Sensors to the sensor network context.

4.1 Internal Programming Interface

An early architectural approach which advocated strong programming semantics for multisensor systems is the Logical Sensor System (LSS) [19]. This approach exploits functional (or applicative) language theory to achieve that. The most developed version of LSS is Instrumented LSS [7].

The ILSS methodology is designed to specify any sensor in such a way that hides its physical nature. The main goal behind ILSS was to develop a coherent and efficient presentation of the information provided by many sensors of different types. This representation provides a means for recovery from sensor failure, and also facilitates reconfiguration of the sensor system when adding or replacing sensors [17]. We propose ILSS as a CSN specification.

ILSS is defined as an extension to LSS, and it has the following components (see Figure 12):

1. *ILS name*: uniquely identifies a module;
2. *Characteristic Output Vector (COV)*: strongly typed output structure, with one output vector and zero or more input vectors;
3. *Commands*: input commands to the module($Commands_{in}$), and output commands to the other modules ($Command_{out}$);
4. *Select function*: selector that detects the failure of an alternate and switches to another alternate if possible;
5. *Alternate subnets*: alternative ways of producing the COV_{out}; it is these implementations of one or more algorithms that carry the main functions of the module;
6. *Control Command Interpreter (CCI)*: interpreter of the commands to the module;
7. *Embedded tests*: self-testing routines that increase robustness and facilitate debugging;
8. *Monitors*: modules that check the validity of the resulting COVs; and
9. *Taps*: hooks on the output lines to view different COV values.

These components identify the system behavior and provide mechanisms for on-line monitoring and debugging. In addition, they give handles for measuring the run-time performance of the system.

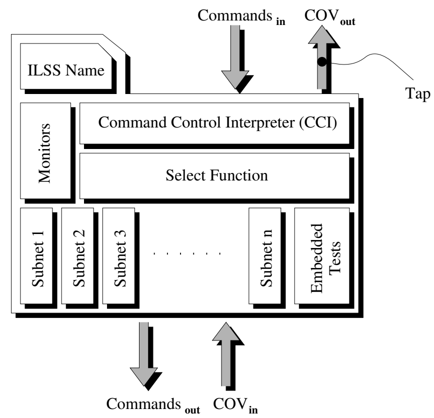

Figure 12: Instrumented Logical Sensor Module.

Monitors are validity check stations that filter the output and alert the CSN or user to any undesired results. Each monitor is equipped with a set of rules (or constraints) that governs the behavior of the COV under different conditions. For example, given the bias estimation method in the previous section, this would be incorporated as a monitor to be run on a regular basis.

Embedded testing is used for on-line checking and debugging purposes. Weller proposed a sensor-processing model with the ability to detect measurement errors and to recover from these errors (Weller, Groen, and Hertzberger 1990). This method is based on providing each system module with verification tests to verify certain characteristics in the measured data, and to verify the internal and output data resulting from the sensor-module algorithm. The recovery strategy is based on rules that are local to the different sensor modules. ILSS uses a similar approach called *local embedded testing*, in which each module is equipped with a set of tests based on the semantics definition of that module. These tests generate input data to check different aspects of the module, then examine the output of the module using a set of constraints and rules defined by the semantics. Also, these tests can take input from other modules to check the operation of a group of modules.

Specific issues include:

- Efficient methods for verifying sensor mote codes during execution. E.g., check that random samples are from the appropriate distribution, check convergence rates, etc. These may be compiled into the Instrumented Logical Sensors as automatic tests.
- Develop appropriate sensor mote framework for model validation. E.g., check that sensed values match model predictions, periodically check model parameters, compare results across motes for consistency.
- Implement sensor data integrity checks. E.g., Check that sensor models match actual data statistics, check for sensor bias, drift, hysteresis, etc. as described in the body of the proposal.
- Obtain performance data from application experiments.

4.2 Demonstration Application

We are engaged in the construction of a sensor network to monitor snow conditions and help determine avalanche probability in back country ski areas in the Wasatch mountains. These sensor networks will be comprised of devices having the ability to communicate, compute and sense the environment temperature, light intensity, pressure, and other properties. We have developed several distributed algorithms for such networks and demonstrated them in simulation and small-scale experiments. Our goal is to build and test a 50-60 node network through Fall 2007. Given our location in the Wasatch mountains in Utah, and the number of deaths each year from avalanche or snow-related conditions, we are developing a network for deployment in the mountains in popular cross-country skiing areas.

Several aspects of snowpack and snow crystal conditions may be modeled (see [34, 33] for a discussion of snowpack simulation, and [8] for the classification and identification of seasonal snow types). A more complete discussion of the physics of snow and its relation to avalanche prediction can be found in [32].

Snow generally occurs in layers of different types, and the goal is to sense the correct features and determine the likelihood of the existence of types of snow grains and slope conditions prone to fracture and sliding.

5 Future Work

1. As described in II, there is work to be done in terms of computational modeling, including: analysis, investigation of stochastic methods, analysis of sensitivity and recovery errors, and analysis of uniqueness of solutions to the sensor localization problem. We propose to investigate various formulations (1-D, 2-D, and 3-D) of the models of continuous phenomena to be monitored. In particular, finite difference, finite element and/or spectral formulations will be evaluated in terms of existence of solutions and complexity of solving resulting systems of nonlinear algebraic equations. We need to investigate properties of the solution classes to our model problems, in order to determine the existence of unique solutions, solutions that are located in small clusters, and/or non-structured solutions. The sensitivity of sensor positions to the errors in process measurements needs to be investigated. In the case of processes of high sensitivity (ill- conditioned) we need to investigate adaptive use of multi-precision computation in order to correctly determine the locations of sensors.

2. The details of the computational mapping methodology need to be worked out and demonstrated on applications. Given our success in mobile robotics applications, we belive that this is readily demonstrable.

3. A rigorous set of experiments must be carried out. We intend to monitor snow quality in the Wasatch mountains; this is an important application, and moreover, one which can di-

rectly exploit the heat equation to determine degradation of snow quality. In addition, the temperature at ground level under the snow should remain fairly constant.

Note that these are not simply theoretical issues, but require implementation and experimentation in the field in order to demonstrate their effectiveness.

Acknowledgments

We would like to thank Mr. Markus Westphal for contributions to the sensor bias solution, 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. Alifanov. *Inverse Heat Transfer Problems*. Springer Verlag, Berlin, 1994.

[2] E. Allgower and K. Georg. *Numerical Continuation Methods*. Springer-Verlag, Berlin, 1990.

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

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

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

[6] 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.

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

[8] S. C. et al. The international classification for seasonal snow on the ground. http://www.earthscape.org/r1/ES16817/Seasonal_Snow.pdf.

[9] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly resilient, energy efficient multipath routing in wireless sensor networks. *Mobile Computing and Communications Review*, 1(2), 2002.

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

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

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

[13] T. 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.

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

[15] T. C. Henderson, M. Dekhil, S. Morris, Y. Chen, and W. B. Thompson. Smart sensor snow. *IEEE Conference on Intelligent Robots and Intelligent Systems*, October 1998.

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

[17] T. C. Henderson, C. Hansen, and B. Bhanu. The specification of distributed sensing and control. *Journal of Robotic Systems*, pages pp. 387–396, Mar. 1985.

[18] T. C. Henderson, J.-C. Park, N. Smith, and R. Wright. From motes to java stamps: Smart sensor network testbeds. In *Proc of IROS 2003*, Las Vegas, NV, October 2003. IEEE.

[19] T. C. Henderson and E. Shilcrat. Logical sensor systems. *Journal of Robotic Systems*, pages pp. 169–193, Mar. 1984.

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

[21] <http://webs.cs.berkeley.edu/SNA/architecture.pdf>. NeTS-NOSS: Creating an Architecture for Wireless Sensor Networks. 2004.

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

[23] G. Karniadakis and R. M. Kirby. *Parallel Scientific Computing in C++ and MPI*. Cambridge University Press, Cambridge, UK, 2002.

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

[25] J. Luitjens, T. Henderson, and M. Berzins. Parallel space-filling curve generation for dynamic load balancing. In *SIAM Conference on Parallel Processing for Scientific Computing*, San Francisco, CA, February 2006.

[26] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor netwroks for habitat monitoring. In *WSNA 2002*, Atlanta, GA, September 2002.

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

[28] W. Oberkampf, T. 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.

[29] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar. SPINS: Security protocols for sensor networks. *Wireless Networks*, 8(5):521–534, Sept 2002.

[30] 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.

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

[32] B. Tremper. *Staying Alive in Avalanche Terrain*. The Mountaineers Books, Seattle, WA, 2001.

[33] B. Vehvilainen. Kluwer Academic Publishers.

[34] D. Weibel, S. Wunderle, and H. Kleindienst. Snow pack simulation in the swiss alps – combining gps and remote sensing to model snow cover in switzerland. In *Proceedings EARSeL-LISSING Workshop Observing our Crvosphere from Space*, pages 179–187, Bern, March 2002.

[35] 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.

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

[37] M. Woolfson and G. Pert. *An Introduction to Computer Simulation*. Oxford University Press, Oxford, UK, 1999.

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

[39] F. Zhao and L. Guibas. Preface. In *Proc of IPSN 2003*, pages v–vi, Palo Alto, CA, April 2003. LNCS.

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