Proceedings of the 1996 IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems

Instrumented Sensor Systems

Mohamed Dekhil and Thomas C. Henderson
Department of Computer Science
University of Utah
Salt Lake City, UT 84112

Abstract

In modeling and designing sensor systems, there is always a tradeoff between increasing the robustness of the sensor system (i.e., increasing the accuracy and reliability of the data) and increasing the efficiency of sensor use. To compare alternative solutions, we need to define quantitative measures that capture the required sensor system characteristics such as time and space complexity, robustness, error variance, etc. Our goal is to develop tools to assist selecting between the tradeoffs and defining models of global data values which establish a closer tie between data and its use. We propose to introduce the notion of instrumented sensor system *for online monitoring and debugging. To achieve our goal, we start by defining a sensor model using formal semantics techniques.*

In this paper, we show that an operational semantics of logical sensor systems provides a strong computational and operational engine that can be used to define and propagate several quantitative measures to evaluate and compare between design alternatives. The application of the proposed modeling approach is illustrated by an example in which two sensing strategies for determining wall pose using sonar sensors are compared.

1 Introduction

Increasing the robustness and reliability of a multisensor system requires adding redundant components and increasing the number of readings. This results in increased system cost. In contrast, increasing the efficiency of the system means less redundant components, fewer readings, and lower cost. Therefore, there is always a tradeoff between robustness and efficiency. It is desirable to find a general model for the different types of sensor systems that allows modeling these systems independent of the physical sensors used, and enables studying the performance and robustness of such systems. There have been many attempts to provide "the" general model along with its mathematical basis and description. Some of these modeling techniques concern error analysis and fault tolerance of multisensor systems [3, 16, 17]. Other techniques are "model-based" which requires a priori knowledge of the scanned object and its environment [6, 9]. These techniques help fit data to a model, but do not provide the means to compare alternatives. Task-directed sensing is another approach to devise sensing strategies [2, 10], but again, it does not provide any measures to evaluate the sensor system in terms of robustness and efficiency.

Another approach to modeling sensor systems is to define sensori-computational systems associated with each sensor [5]. This approach utilizes the concept of "information invariants" to define some measure of information complexity. However, as stated by Donald, the measures for information complexity are fundamentally different from performance measures and it does not permit one to judge which system is "simpler," "better," or "cheaper."

To compare between the performance of different sensor systems, there must be some quantitative bases for comparison. Doebelin [4] divides the performance characteristics of measurement systems into *static characteristics* and *dynamic characteristics*. Static characteristics involve the measurements which are constant or vary only quite slowly such as accuracy, precision, and span. On the other hand, dynamic characteristics concern the measurement of rapidly varying quantities. The dynamic description of the system involves differential equations that represent the response of the system to different input functions.

In this paper, we propose a new approach for sensor system modeling which allows defining and propagating the required measures. This approach is based on defining the formal semantics of a virtual sensing machine. Formal semantics of programming languages provides techniques to describe the meaning of a language based on precise mathematical principles. These formal techniques provide precise machine-independent concepts, unambiguous specification techniques, and a rigorous theory to support reliable reasoning [8].

The main types of formal semantics are *Denotational Semantics* which concerns designing denotations for con-

This work was supported in part by the Advanced Research Projects agency under Army Research Office grants number DAAH04-93-G-0420 and by NSF grant CDA 9024721.

0-7803-3700-X/96 $5.00 ©1996 IEEE

structs, *Operational Semantics* which concerns the specification of an abstract machine together with the machine behavior when running the program, and *Axiomatic Semantics* which concerns axioms and rules of inference for reasoning about programs [1].

Operational semantics is the most suitable type to use as a basis for modeling sensor system since it provides a precise description of the dynamic behavior of the system. Furthermore, operational semantics answers the question of "how" the system is computing while denotational semantics refers to "what" the system is computing. In sensor systems, we know what is being computed, but we need to know how it was computed which allows us to quantify some of the measures of interest such as efficiency, robustness and accuracy.

To this end, the notion of *instrumented sensor systems* is introduced, which is a modeling and design methodology that facilitates interactive, on-line monitoring for different components of the sensor system. It also provides communication and debugging tools and analysis measures for the sensor system. The instrumented sensor approach is based on a sensori-computational model which defines the components of the sensor system in terms of their functionality, accuracy, robustness, and efficiency. This model provides a strong computational engine for propagating these measures when constructing logical sensors and multisensor systems.

2 Logical Sensor Systems

The notion of *Logical Sensor Systems* (LSS) was introduced by Henderson and Shilcrat [14]. LSS is a methodology to specify any sensor in such a way that hides its physical nature. The main goal behind LSS 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.

A logical sensor does not have to be a physical (hardware) entity; it can be a software program (e.g., for pattern recognition or edge detection), or it can be a combination of software and hardware. In any case, it should be viewed as a distinct sensor which can be replaced by special hardware. Figure 1 shows an LSS for measuring 3D points of an object. There are two options in that system; using two 2D cameras and a stereo program (LSS4), or using an active range camera (LSS3).

The scope of logical sensors was extended to include a *control command interpreter* which forms the operational part of the LSS [13]. Each logical sensor receives some control commands from a higher level in the LSS hierarchy, and sends commands to the logical sensors in the lower levels.

Our approach is based on defining the sensor system as logical components and defining the operational semantics in terms of these components. This is explained in detail in the following section.

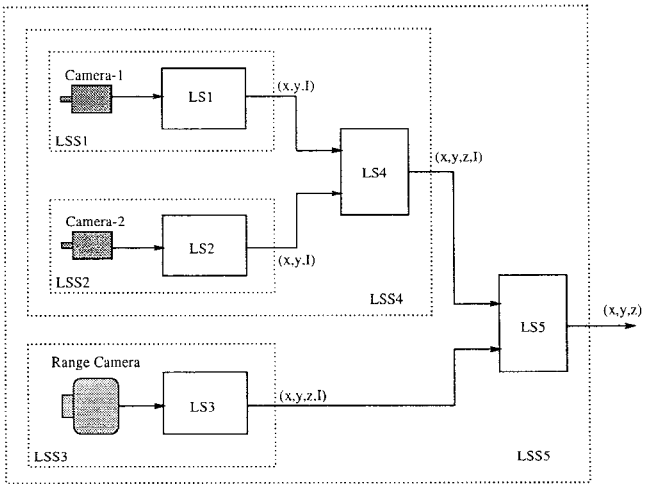

Figure 1: An example of a logical sensor system.

3 The Model

As mentioned earlier, our modeling approach is based on defining an abstract sensing machine and then defining its operational semantics. The rationale behind using operational semantics is that operational semantics:

- Provides a uniform definition for diverse sensor systems.
- Provides a precise description of the dynamic behavior of the system.
- Allows us to define the required quantitative measures and to include them in the system state, and provides a systematic procedure for propagating these measures across the sensor system.
- Provides a strong theoretical basis for online analysis and debugging of a sensor system.

The following notation is used to describe an operational step:

$$\frac{Condition}{(Command_1, State_1) \triangleright (Command_2, State_2)}$$

where $Condition$ is a conditional statement which determines if the operational step can be executed or not. $Command_1$ is the command to be executed, $Command_2$ is the next command after executing $Command_1$, $State_1$ is the current state and $State_2$ is the new state after executing $Command_1$.

By decomposing the logical sensor system into functional components, and defining the system state to include the measured data and the associated quantitative measures such as error variance and time complexity, we can use similar notation to define the operational semantics of the sensor system.

As an example, we can define the semantic statement:

$$\frac{sensor\_ready}{(read\_sensor, s) \triangleright (filter\_data, s')}$$

which says that under the condition that the sensor is ready to receive data (e.g., it is turned on), the command *read_sensor* can be executed yielding a new command *filter_data* and a new state that contains the sensor data (raw data)

In the same way we can define all the necessary operations that define our abstract sensor architecture. By including some information in the state s such as the noise function, the cost of the operation in terms of time and space, the rate of failure, etc., we can quantify some of the measures we are interested in such as accuracy, efficiency, and robustness. This approach provides a systematic procedure for propagating these measures across the sensor system.

The error statistics associated with each component of the system need to be propagated to formulate the statistics for the overall system. System robustness and efficiency vary according to the system layout and the types of sensors used in constructing the system. We propose to use a simple approach for error propagation which does not require carrying a lot of information throughout the system. This approach is based on the uncertainty propagation described in [7, 15]. Assume that we have a certain module with n inputs $X = (x_1, x_2, \ldots, x_n)$ and m outputs $Y = (y_1, y_2, \ldots, y_m)$ such that $Y = f(X)$, and assume that we know the error variance associated with the input vector $\Lambda_X = (\Lambda_{x_1}, \Lambda_{x_2}, \ldots, \Lambda_{x_n})$ (see Figure 2), then the error variance can be calculated using the equation:

$$\Lambda_Y = \left(\frac{\partial Y}{\partial X}\right) \Lambda_X \left(\frac{\partial Y}{\partial X}\right)^T$$

where $\frac{\partial Y}{\partial X}$ is the partial derivative of Y with respect to X evaluated at the measured value of the input vector X. If all the elements in X are independent variables, then this equation can be written as:

$$\Lambda_{y_i} = \sum_{j=1}^{n} \left(\frac{\partial y_i}{\partial x_j}\right)^2 \Lambda_{xj} \ , i = 1, 2, \ldots, m$$

4 Example: Wall Pose

The following example uses the proposed modeling approach to compare alternatives based on quantitative

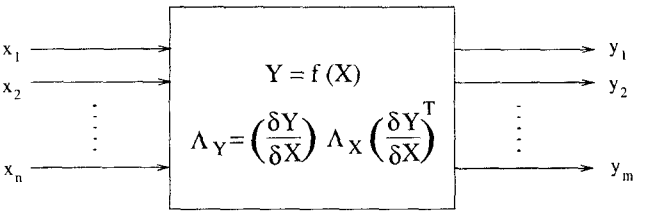

Figure 2: A simple approach for error propagation.

measures for accuracy and efficiency. This example is based on a mobile robot called "LABMATE" designed by Transitions Research Corporation. In the example, we are interested in finding the position and orientation of walls in the surrounding environment using sonar sensors. We consider two different logical sensors that produce wall pose and find the corresponding errors and time complexity for each. The first logical sensor considers the sonar sensor as a point sensor and the location of the return point is at the middle of the sonar spread wedge. The second logical sensor deals with the sonar sensor as a wedge sensor; that is, it it returns a wedge centered at the sonar sensor and spread by an angle 2θ. Both logical sensors use two sonar readings to calculate the wall position. Figure 3 shows the two logical sensors.

In this figure, r_1 and r_2 are the two sonar readings, *point_sensor* determines the location of the sonar reading as the mid-point of its spread wedge, *2_point_line* returns two 2D points, (x_1, y_1) and (x_2, y_2), on the line representing the wall, while *wedge_sonar_line* takes the two range values r_1 and r_2, and the spread angle of the sonar beam θ, and returns two 2D points on the line representing the wall.

Modeling LSS1

Using the operational semantics approach to model the first sensor system, we have:

$$\frac{sonar\_ready}{(read\_sonar, s_0) \triangleright (point\_sonar, s_1)}$$

$$\frac{}{(point\_sonar, s_1) \triangleright (2\_point\_line, s_2)}$$

$$\frac{}{(2\_point\_line, s_2) \triangleright (skip, s_3)}$$

where s_i is the system state which, in this example, can be written as:

$$s_i = \{D_i, t_i, e_i\}$$

where D_i is the data perceived (or calculated), t_i is the time complexity, and e_i is the error associated with the data represented by the error variance. Assume that $s_0 = \{NULL, 0, 0\}$, and let's determine the values for s_1, s_2, and s_3.

At the first level, we have the physical sonar sensor. The data in this case is the returned value representing the distance from the sonar to the nearest object in

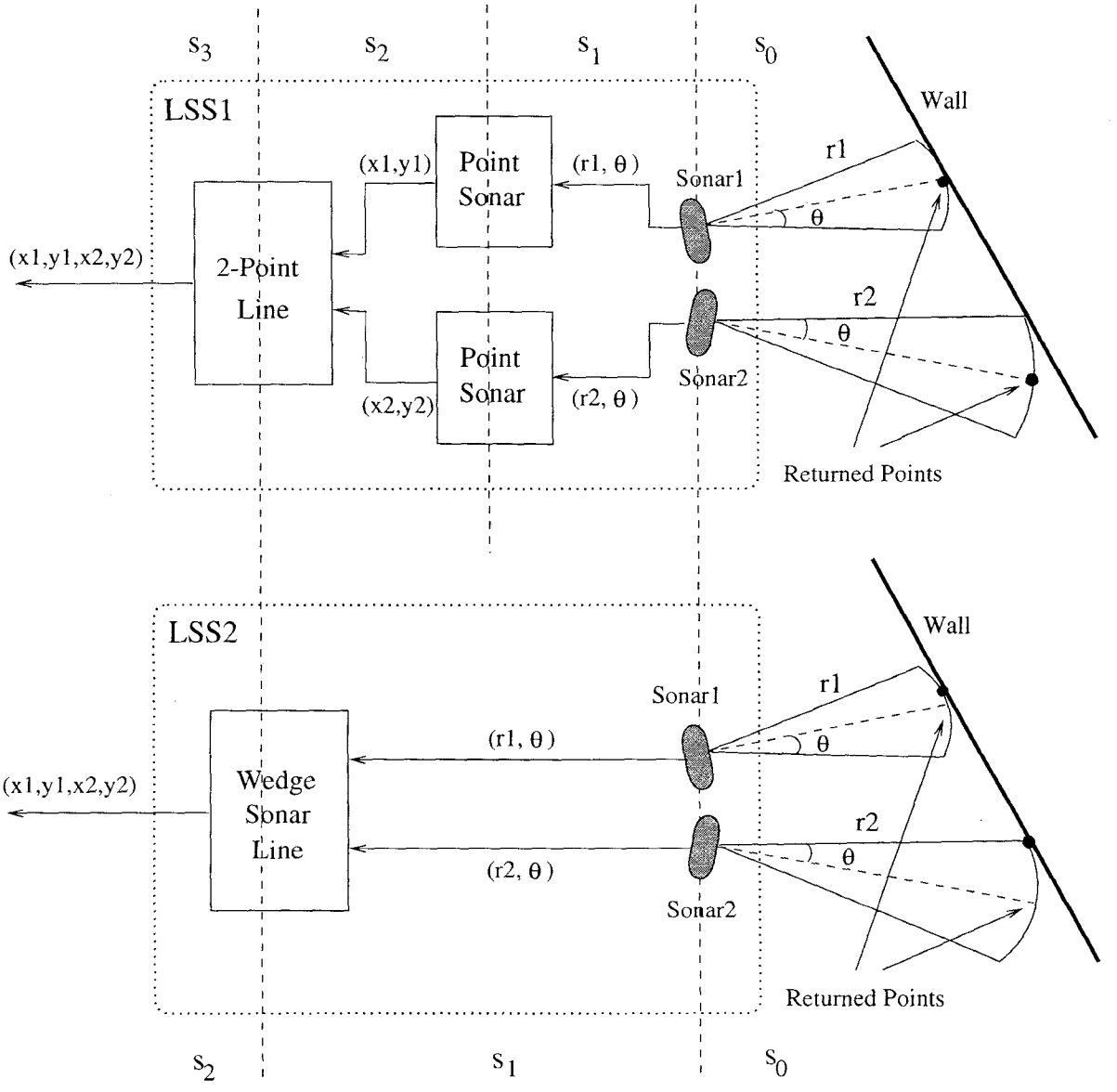

Figure 3: Two logical sensors for determining wall position.

its range, r_1 and r_2. The time complexity is the time required between issuing the *read_sonar* command and getting the data back. This time depends on the range returned by the sonar, the architecture running the program and the operating system overhead. Let's assume the time for the first level is $0.30ms$.

The error can be determined either from the manufacturer specs, or from experimental data. In this example we will use the error analysis done by Schenkat and Veigel [18] in which there is a Gaussian error with mean μ and variance σ^2. From this analysis, the variance is a function of the returned distance r. To simplify the problem let's assume that the variance in both sensors is equal to $4.0mm^2$. Also, in this logical sensor there is an algorithmic error due to the fact that the actual point can be anywhere on the sonar wedge, and not just at the center as is returned. This error can be calculated by assuming that θ is a random variable uniformly distributed over the range $(-11^o, 11^o)$ given that the beam spread of the sonar is 22^o (see Figure 4). The variance of the angle θ is therefore equal to:

$$\Lambda_\theta = \frac{(22\pi/180)^2}{12} = 0.0123rad^2$$

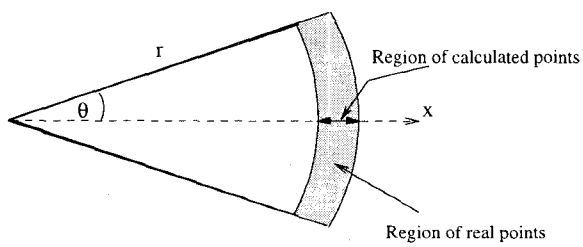

Figure 4: The error in the point-sonar module.

In this case we have:

$$s_1(sonar1) = \{(r_1, \theta), 0.30ms, (4.0mm^2, 0.0123rad^2)\}$$

$$s_1(sonar2) = \{(r_2, \theta), 0.30ms, (4.0mm^2, 0.0123rad^2)\}$$

At the second level, we have the *point_sonar*. In this case, the generated data is the estimated location of the perceived point, x and y. The time complexity is the time required to calculate x and y based on the sensor location and the value of r. Let's assume that the time required is $t_{point\_sonar} = 0.9ms$.

There are two sources of error in this case; error due to the uncertainty in r, and the algorithmic error represented by the variance in θ. The output of this module is the estimated (x, y) location of the point where the sonar beam hits the wall. This can be written as:

$$x_1 = r_1 \cos\theta, \;\; y_1 = r_1 \sin\theta$$

$$x_2 = r_2 \cos\theta, \;\; y_2 = r_2 \sin\theta$$

Since r_i and θ are independent, then the error associated with x_i and y_i can be calculated as follows:

$$\Lambda_{x_i} = \left(\frac{\partial x_i}{\partial r}\right)^2 \Lambda_r + \left(\frac{\partial x_i}{\partial \theta}\right)^2 \Lambda_\theta$$

$$\Lambda_{y_i} = \left(\frac{\partial y_i}{\partial r}\right)^2 \Lambda_r + \left(\frac{\partial y_i}{\partial \theta}\right)^2 \Lambda_\theta$$

To calculate the worst case error, we evaluate these expressions at $\theta = 11^o$, assuming that $r_1 = 1000mm$ and $r_2 = 1100mm$, we get:

$$\Lambda_{x_1} = \cos^2\theta * 4.0 + (-r_1 \sin\theta)^2 * 0.0123 = 451.67$$

$$\Lambda_{y_1} = \sin^2\theta * 4.0 + (r_1 \cos\theta)^2 * 0.0123 = 11852.18$$

$$\Lambda_{x_2} = \cos^2\theta * 4.0 + (-r_2 \sin\theta)^2 * 0.0123 = 545.72$$

$$\Lambda_{y_2} = \sin^2\theta * 4.0 + (r_2 \cos\theta)^2 * 0.0123 = 14341.28$$

So, we can write s_2 for the sonars as:

$$s_2(point\_sonar1) = \{(x_1, y_1), 0.3ms + 0.9ms, (451.67, 11852.18)\}$$

$$s_2(point\_sonar2) = \{(x_2, y_2), 0.3ms + 0.9ms, (545.72, 14341.28\}$$

Finally, the last level, *2_point_line*, is just combining the two outputs of the two point sonars to form two 2D points on the line representing the wall. Assume that this operation takes $0.01ms$, there for, the total time required is:

$$t_3 = 1.20 + 0.01 = 1.21ms$$

then the final state s_3 will be:

$$\{(x_1, y_1, x_2, y_2), 1.21ms, (451.67, 11852.18, 545.72, 14341.28)\}$$

where the error in x and y is im mm^2.

Modeling LSS2

We can model the second logical sensor in the same way we did for $LSS1$. Starting by defining the operational semantics of this sensor we get:

$$\frac{sonar\_ready}{(read\_sonar, s_0) \triangleright (wedge\_sonar\_line, s_1)}$$

$$\frac{}{(wedge\_sonar\_line, s_1) \triangleright (skip, s_2)}$$

At the first level we have the same physical sonar as in $LSS1$, therefore we have the same time complexity and same error in r_1 and r_2. However, we don't have the algorithmic error as in $LSS1$ which introduced an error in the beam spread θ. Therefore, we can set the error in θ to zero (this corresponds to the worst case error of 11^o for LSS1.) In this case, s_1 can be written as:

$$s_1 = \{(r_1, r_2, \theta), 0.30ms, (4.0mm^2, 4.0mm^2, 0)\}$$

At the last level we have the *wedge_sonar_line* module that takes the two sonar readings r_1 and r_2 and the spread angle θ, and returns two 2D points on the line

representing the wall. Let's assume that the time complexity of this module is $1.22ms$. As described in [11, 12], there are five possible cases for that line depending on the values of r_1 and r_2. In any case, the two points laying on the line can be written as:

$$x_1 = r_1 \cos\alpha_1, \; y_1 = r_1 \sin\alpha_1$$

$$x_2 = r_2 \cos\alpha_2, \; y_2 = r_2 \sin\alpha_2$$

where the values of $\alpha 1$ and α_2 are between $-\theta$ to θ (see Figure 5).

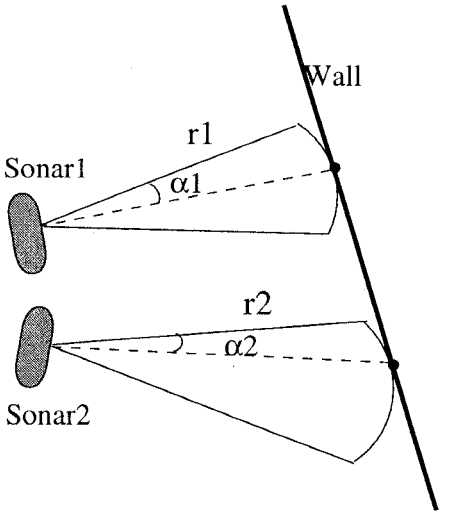

Figure 5: The general case for the points returned by the wedge_sonar_line.

Considering the worst case error, we can set $\alpha_1 = \alpha_2 = \theta$. And since error in θ is zero, then the error in the calculated points can be calculated as follows:

$$\Lambda_{x_i} = \left(\frac{\partial x_i}{\partial r}\right)^2 \Lambda_r$$

$$\Lambda_{y_i} = \left(\frac{\partial y_i}{\partial r}\right)^2 \Lambda_r$$

Evaluating these expressions using the same values we used for $LSS1$ we get:

$$\Lambda_{x_1} = \cos^2\theta * 4.0 = 3.85$$

$$\Lambda_{y_1} = \sin^2\theta * 4.0 = 0.15$$

$$\Lambda_{x_2} = \cos^2\theta * 4.0 = 3.85$$

$$\Lambda_{y_2} = \sin^2\theta * 4.0 = 0.15$$

So, we can write s_2 as:

$$s_2(LSS2) = \{(x_1, y_1, x_2, y_2), 1.52ms, (3.85, 0.15, 3.85, 0.15)\}$$

Comparing $s_3(LSS1)$ and $s_2(LSS2)$ we can select between these two alternatives based on their time complexity and their error variance for the returned points. These two logical sensors were used with the LABMATE to find the location of walls using synthetic and real data. Figure 6 shows the error in both cases using synthetic data representing 300 lines at different angles from the sonars, the results show that the error using $LSS2$ was much smaller than the error using $LSS1$ which supports the analysis conducted using our proposed approach. In this figure, the error in ρ and θ varies based on the region the line belongs to.[1] For example, we can see transition at line 100 which corresponds to moving from region 1 to region 2.

Note that in this example we did not consider the error in the sonar location and orientation just for simplicity. However, these errors can be incorporated in the model in the same manner.

5 On-line Monitoring and Embedded Testing

In the two previous examples, a lot of details were ignored to simplify the problem. For example, the material of the wall surface can affect the accuracy of the sonar sensors. Also, the computer used to run the programs and its operating system can affect the timing. The numerical errors due to quantization and rounding affect the accuracy of the results. For real applications in real environments, we have to consider these details and study their impact on the different measures used to evaluate the system. In some cases this cannot be determined using analytical techniques due to lack of information, varying environmental conditions, unknown relation between the factor and some of the measures, or complex physics that requires long computation which cannot be evaluated in real time.

To solve this problem, on-line monitoring is used when the analysis fails to determine one or more of the required measures. By monitoring the output at certain tapping locations, the user can view the effect of modifying some of the system parameters on the output and compare several alternatives. The main contribution here is providing the user with enough information to know what to look for and where. Therefore, on-line monitoring is a supplemental component in the proposed modeling approach.

Embedded testing is used for on-line checking and debugging proposes. Weller et al. [19] proposed a sensor processing model with the ability to detect measurement errors and to recover from these errors. 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. This approach will

[1] There are five different regions that a line representing the wall can fall in. (See [12] for more details.)

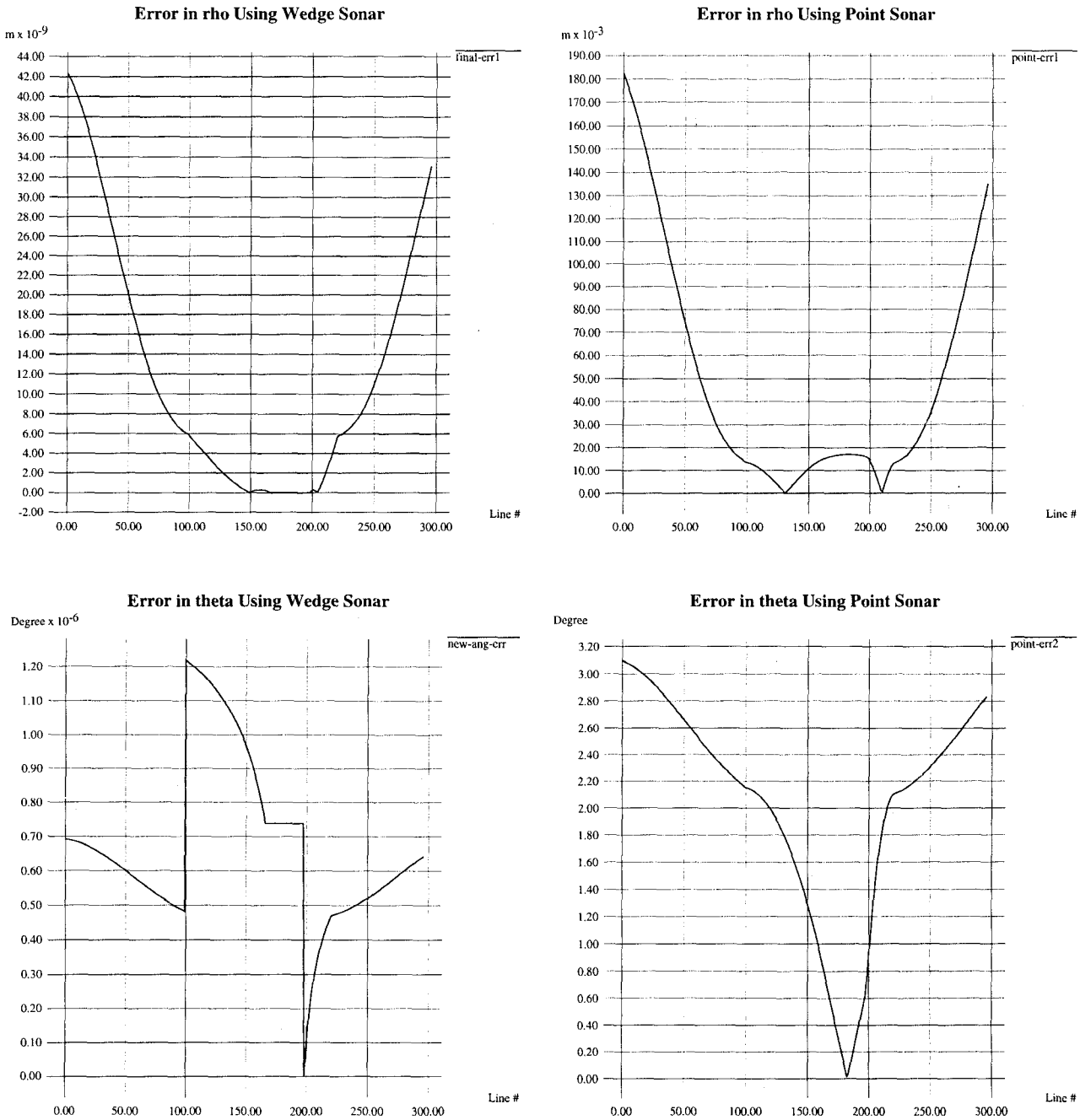

Figure 6: The error in both cases using synthetic data.

be used in our framework to provide the user with possible locations to tap into when there is a problem with the system.

6 Conclusion

In this paper, we introduced the notion of *instrumented sensor systems*, which is a modeling and design methodology that facilitates interactive, on-line monitoring for different components of the sensor system. It also provides debugging tools and analysis measures for the sensor system. The instrumented sensor approach can be viewed as an abstract sensing machine which defines the semantics of sensor systems. this provides a strong computational and operational engine that can be used to define and propagate several quantitative measures to evaluate and compare design alternatives. These measures are integrated with the system state and are modified after each operational step to reflect the affect of each operational component of the sensor system on these measures.

The application of the proposed approach on a real problem was described and the results were presented. We believe that an operational semantics of logical sensor systems can adequately model diverse types of sensors in a systematic way for practical use.

References

[1] ASHCROFT, E. A. R for semantics. *ACM Transactions on Programming Languages and Systems 4*, 2 (1982), pp. 283–295.

[2] BRIGGS, A., AND DONALD, B. Automatic sensor configuration for task-directed planning. In *IEEE Int. Conf. Robotics and Automation* (May 1994), pp. 1345–1350.

[3] BROOKS, R. R., AND IYENGAR, S. Averaging algorithm for multi-dimensional redundant sensor arrays: resolving sensor inconsistencies. Tech. rep., Louisiana State University, 1993.

[4] DOEBELIN, E. O. *Measurments systems application and design.* McGraw Hill, 1983.

[5] DONALD, B. R. On information invariants in robotics. *Artificial Intelligence*, 72 (1995), pp. 217–304.

[6] DURRANT-WHYTE, H. F. *Integration, coordination and control of multisensor robot systems.* Kluwer Academic Publishers, 1988.

[7] FAUGERAS, O. *Three-dimensional computer vision - a geometric viewpoint.* The MIT Press, 1993.

[8] GORDON, M. J. C. *Denotational description of programming languages.* Springer-Verlag, 1979.

[9] GROEN, F. C. A., ANTONISSEN, P. P. J., AND WELLER, G. A. Model based robot vision. In *IEEE Instrumentation and Measurment Technology Conference* (1993), pp. 584–588.

[10] HAGER, G., AND MINTZ, M. Computational methods for task-directed sensor data fusion and sensor planning. *Int. J. Robotics Research 10*, 4 (August 1991), pp. 285–313.

[11] HENDERSON, T. C., BRUDERLIN, B., DEKHIL, M., SCHENKAT, L., AND VEIGEL, L. Sonar sensing strategies. In *IEEE Int. Conf. Robotics and Automation* (April 1996), pp. 341–346.

[12] HENDERSON, T. C., DEKHIL, M., BRUDERLIN, B., SCHENKAT, L., AND VEIGEL, L. Flat surface recovery from sonar data. In *DARPA Image Understanding Workshop* (February 1996), pp. 995–1000.

[13] HENDERSON, T. C., HANSEN, C., AND BHANU, B. The specification of distributed sensing and control. *Journal of Robotic Systems* (Mar. 1985), pp. 387–396.

[14] HENDERSON, T. C., AND SHILCRAT, E. Logical sensor systems. *Journal of Robotic Systems* (Mar. 1984), pp. 169–193.

[15] HOLMAN, J. P., AND W. J. GAJDA, J. *Experimental methods for engineers.* McGraw-Hill, 1978.

[16] IYENGAR, S. S., AND PRASAD, L. A general computational framework for distributed sensing and falt-tolerant sensor integration. *IEEE Trans. Systems Man and Cybernetics* (May 1994).

[17] PRASAD, L., IYENGAR, S. S., RAO, R. L., AND KASHYAP, R. L. Fault-tolerence sensor integration using multiresolution decomposition. *The American Physical Society* (April 1994), pp. 3452–3461.

[18] SCHENKAT, L., VEIGEL, L., AND HENDERSON, T. C. Egor: Design, development, implementation – an entry in the 1994 AAAI robot competition. Tech. Rep. UUCS-94-034, University of Utah, Dec. 1994.

[19] WELLER, G. A., GROEN, F. C. A., AND HERTZBERGER, L. O. A sensor processing model incorporating error detection and recoverry. In *Traditional and non-traditional robotic sensors.* Edited by T. C. Henderson. (1990), Springer-Verlag, pp. 351–363.