Instrumented Sensor System – Practice

Mohamed Dekhil and Thomas C. Henderson

UUSC-97-014

Department of Computer Science
University of Utah
Salt Lake City, UT 84112 USA

March 1997

Abstract

In previous work, we introduced the notion of Instrumented Logical Sensor Systems (ILSS) that are derived from a modeling and design methodology [2, 4]. 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 approach provides a uniform specification language to define sensor systems as a composition of smaller, predefined components. From a software engineering standpoint, this addresses the issues of modularity, reusability, and reliability for building complex multisensor systems.

In this report, we demonstrate the practicality of this approach and discuss several design and implementation aspects in the context of mobile robot applications.

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.

1 Introduction

In any closed-loop control system, sensors are used to provide the feedback information that represents the current status of the system and the environmental conditions. Building a sensor system for a certain application is a process that includes the analysis of the system requirements, a model of the environment, the determination of system behavior under different conditions, and the selection of suitable sensors. The next step in building the sensor system is to assemble the hardware components and to develop the necessary software modules for data fusion and interpretation. Finally, the system is tested and the performance is analyzed. Once the system is built, it is difficult to monitor the different components of the system for the purpose of testing, debugging and analysis. It is also hard to evaluate the system in terms of time complexity, space complexity, robustness, and efficiency, since this requires quantitative measures for each of these aspects.

In addition, designing and implementing real-time systems are becoming increasingly complex because of many added features such as fancy graphical users interfaces (GUIs), visualization capabilities and the use of many sensors of different types. Therefore, many software engineering issues such as reusability and the use of COTS (Commercial Off-The Shelf) components [20], reliability [13, 14, 22], and embedded testing [23] are now getting more attention from system developers.

In previous work, we proposed to use formal semantics to define performance characteristics of sensor systems [2]. Then, we presented a theoretical framework for modeling and designing sensor systems based on a formal semantics in terms of a virtual sensing machine [4]. This framework defines an explicit tie between the specification, robustness and efficiency of the sensor system by defining several quantitative measures that characterize certain aspects of the system's behavior. Figure 1 illustrates our proposed approach which provides static analysis (e.g., time/space complexity, error analysis) and dynamic handles that assist in monitoring and debugging the system.

In this report, we show how to use the proposed framework for real-time monitoring and testing. Several experiments conducted on a mobile robot platform are presented and the results are discussed.

2 Related Work

Each sensor type has different characteristics and functional description. Therefore it is desirable to find a general model for these different types that allows modeling sensor systems that are 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 [1, 5, 11, 18, 19]. Other techniques are model-based and

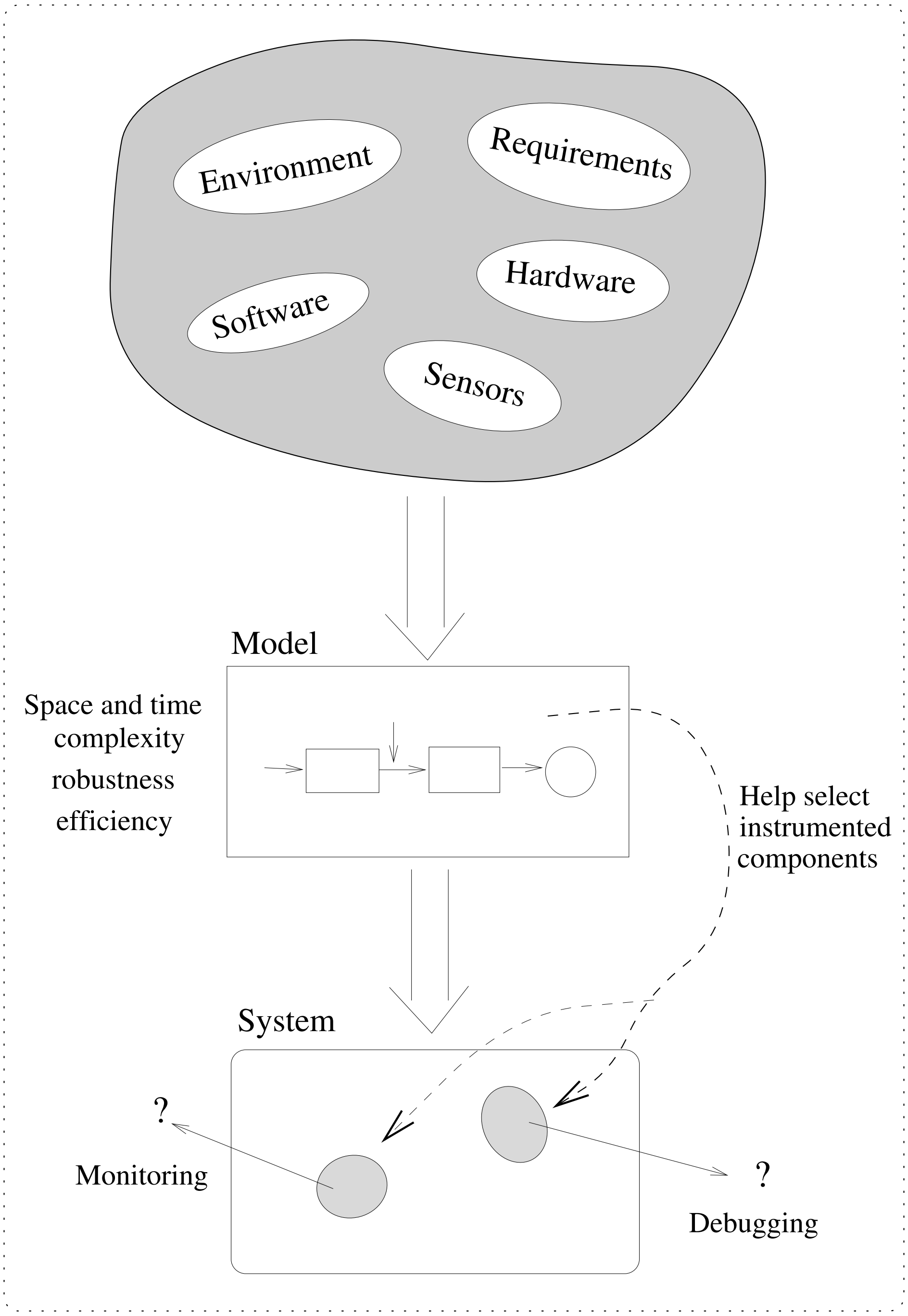

Figure 1: The proposed modeling approach.

require a priori knowledge of the scanned object and its environment [8, 9, 12]. These techniques help fit data to a model, but do not provide the means to compare alternatives.

Another approach to modeling sensor systems is to define sensori-computational systems associated with each sensor to allow design, comparison, transformation, and reduction of any sensory system [7]. In this approach the concept of information invariants is used to define some measure of information complexity. This approach provides a very strong computational theory which allows comparing sensor systems, reducing one sensor system to another, and measuring the information complexity required to perform a certain task. However, as stated by Donald, the measures for information complexity are fundamentally different from performance measures. Also, this approach does not permit one to judge which system is "simpler," "better," or "cheaper."

In most applications, performance analysis is essential to evaluate the system and compare alternative solutions. Measuring the performance of any system requires identifying a set of metrics that captures the desired characteristics of the system. In the robotics field, several research efforts have been directed towards defining such metrics and evaluating the performance of new control algorithms.

Lee and Resse proposed a quantitative evaluation approach for navigation and planning strategies for mobile robots [15, 16]. They conducted an experimental investigation into the robots exploration capabilities using a novel metric that predicts the effectiveness of the robot in executing a set of tasks using a map that is built using a Polaroid ultrasonic range sensor. This approach matches our view of performance evaluation of sensor systems in terms of providing a set of metrics and conducting experimental evaluation of the system to capture the dynamics and variations in the system and its environment.

3 The ILSS Approach

The *Instrumented Logical Sensor System* (ILSS) approach represents our methodology for incorporating design tools and allows static and dynamic performance analysis, on-line monitoring, and embedded testing. Figure 2 shows the components of our framework. First (on the left), an Instrumented Logical Sensor Specification is defined, as well as $\mathcal{F}$, a set of functions which measure system properties of interest. This specification is derived from a mathematical model, simulation results, or from descriptions of system components. Analysis of some aspects of the ILSS are possible (e.g., worst-case complexity of algorithms). Next (the center of the figure), an implementation of the system is created; this can be done by hand or automatically generated in a compile step (note that the original Logical Sensor Specifications[10] could be compiled into Unix shell script or Function Equation Language (FEL), an applicative language). Either way, the monitoring, embedded testing or taps are incorporated into the system implementation. Finally (the right hand side), validation is achieved by analyzing the system response and performance

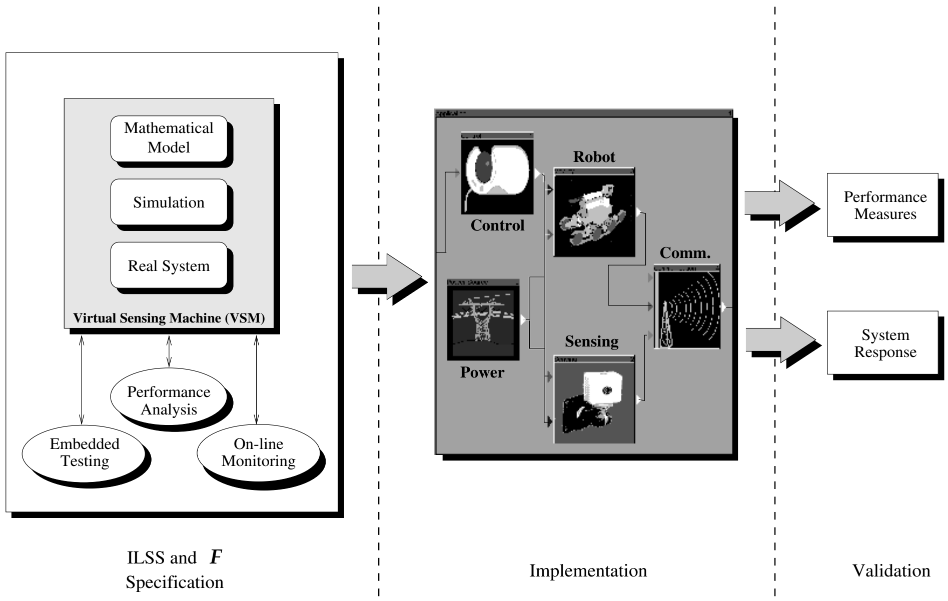

Figure 2: The Instrumented Logical Sensor System Components.

measures generated during system execution. In this way, there are some semantic constraints on the values monitored which relate the system output measures to the original question posed for the specification.

In our proposed framework, a sensor system is composed of several ILSS modules connected together in a certain structure. We defined operations for composing ILSS modules, and defined the semantics of these operations in terms of the performance parameters [4]. The semantics of these construction operations is defined as a set of functions that propagate the required performance measures. Several techniques can be used for propagation: best case analysis, worst case analysis, average, etc. Selecting among these depends on the application, hence it should be user defined.

4 ILSS Implementation

The main objective of this research project is to develop a modeling and prototyping environment with tools for analysis, debugging, and monitoring sensor systems with emphasis on mobile robot control applications. In this section, we present the specification of the ILSS and an implementation of the system in a multi-tasking shared-memory environment.

4.1 ILSS Specification

The ILSS is comprised of several components that 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. the ILSS components are (see Figure 3):

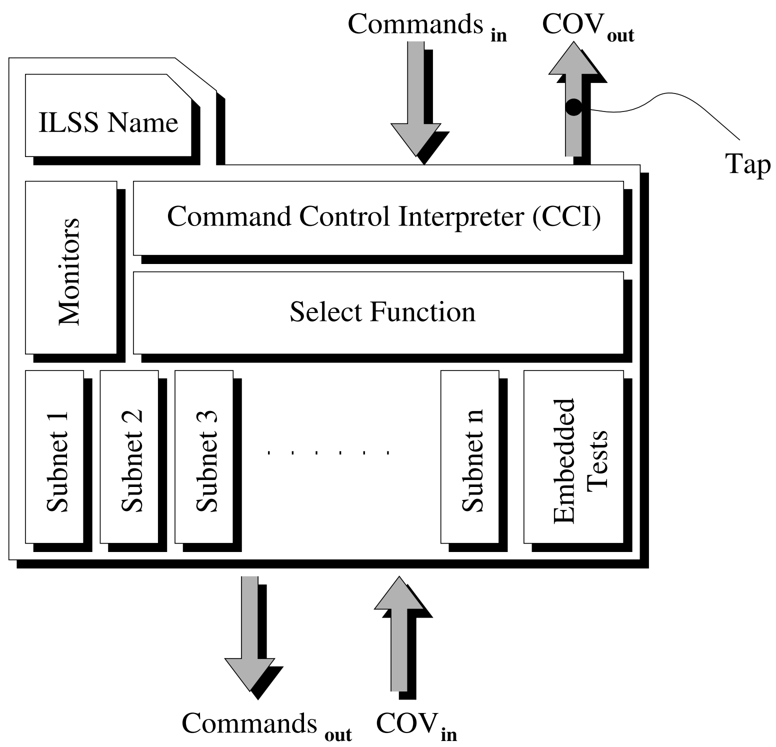

Figure 3: The ILSS components.

1. *ILS Name*: uniquely identifies a module.

2. *Characteristic Output Vector (COV)*: strongly typed output structure. We have one output vector (COV_{out}) and zero or more input vectors (COV_{in}).

3. *Commands*: input commands to the module ($Commands_{in}$) and output commands to other modules ($Commands_{out}$).

4. *Select Function*: selector which 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 which increase robustness and facilitate debugging.

8. *Monitors*: modules that check the validity of the resulting COVs.

9. *Taps*: hooks on the output lines to view different COV values.

Monitors are validity check stations that filter the output and alert the 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 situations.

Embedded testing is used for on-line checking and debugging proposes. Weller proposed a sensor processing model with the ability to detect measurement errors and to recover from these errors [23]. 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. Another approach to failure classification and recovery was proposed by Murphy [17]. We used a similar approach in our framework called *local embedded testing* in which each module is equipped with a set of tests based on the semantic 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 data from other modules if we need to check the operation for a group of modules.

4.2 Implementing ILSS Components

An object-oriented approach is used to develop the ILSS components. Each component is an object that possesses some basic features common to all components plus some additional features that are specific to each ILSS type. The following are some of the basic functions supported by all components:

Initialize: performs some initialization steps when the component is created.

Calibrate: starts a calibration routine.

Run: generates the COV corresponding to the current input and the component status.

Reset: resets all the dynamic parameters of the component to their initial state.

Test: performs one or more of the component's embedded tests.

Select: selects one of the alternate subnets. This allows for dynamic reconfiguration of the system.

Monitor: observes the COV and validates its behavior against some predefined characteristic criteria.

Tap: displays the value of the required variables.

We use these components to build complex sensor systems in a hierarchical structure. Each component can run as a separate process or several of them can run as one process depending on the application requirements and the degree of concurrency needed. Monitors, taps, and embedded tests are implemented as sub-processes generated from the main ILSS process. These components communicate through the COV and the Command lines using shared-memory structures. This shared memory architecture was developed to design and implement distributed control systems for mobile robots (see [3, 6] for more detail.) Figure 4 shows the structure of one ILSS component with its different modules and communication lines.

A complex sensor system may have tens or hundreds of these components connected together in a certain structure. This structure is kept in a small database that contains all the necessary information about each component and the way they are connected. This adds more flexibility to the system and allows for rapid construction and modification of the system components and its parameters. Figure 5 shows the database schema used for this purpose.

5 Experiment: Adaptive Wall Following

Several experiments has been conducted to demonstrate the usability of the proposed framework in modeling and designing sensor systems [4, 2]. In the following experiment we implement a simple wall-following algorithm using two alternatives; sonar sensors and a camera. The sonars and the camera are mounted on a LABMATE mobile robot designed by Transitions Research Corporation. The LABMATE was used for several experiments in the Department of Computer Science at the University of Utah. It was also entered in the 1994 and 1996 AAAI Robot Competition [21] and it won sixth and third place, respectively. For that purpose, the LABMATE was equipped with 24 sonar sensors, eight infrared sensors, a camera and a speaker.[1]

Figure 6 shows the ILSS structure used for this experiment along with the robot controller and the follow-wall components. This experiment illustrates the usefulness of the design tools provided by the ILSS architecture such as taps, monitors, and embedded tests. The idea of using two different (and independent) ILSs is to increase the reliability and the robustness of the system.

[1]The LABMATE preparations, the sensory equipments, and the software and hardware controllers were done by L. Schenkat and L. Veigel at the Department of Computer Science, University of Utah.

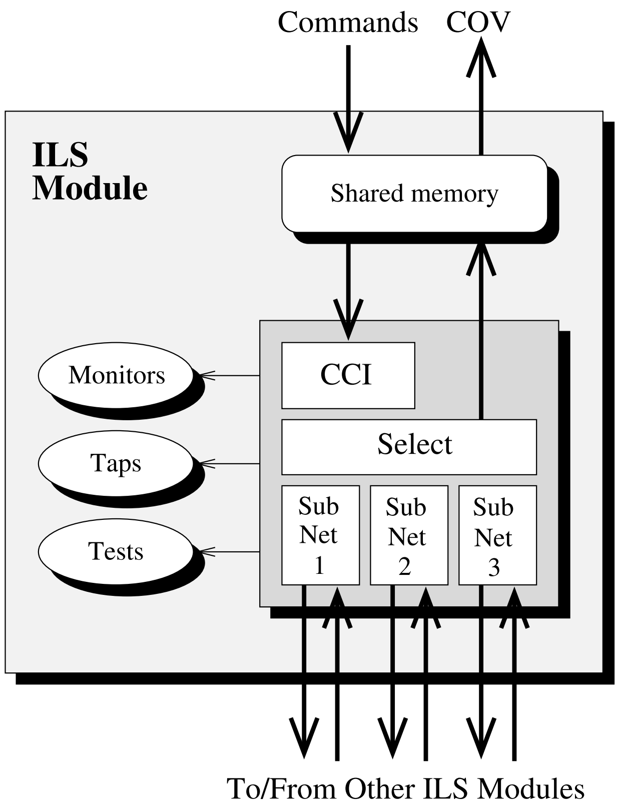

Figure 4: ILSS component structure.

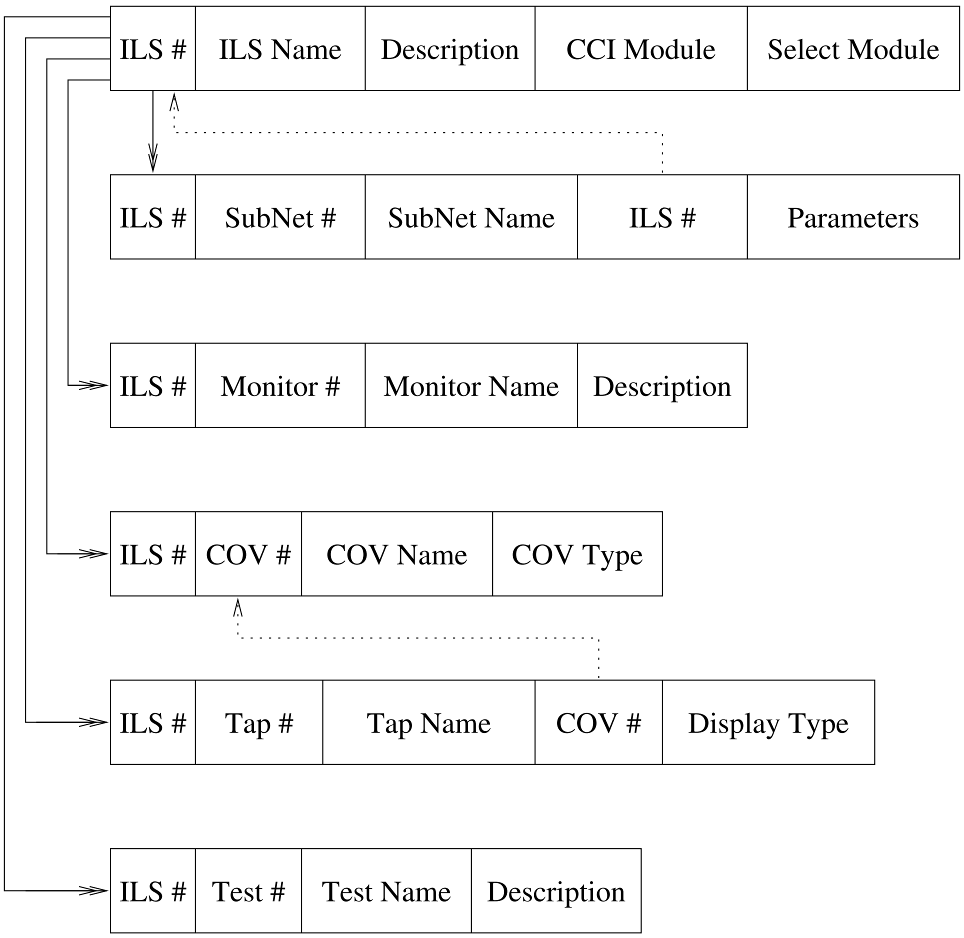

Figure 5: Database schema for ILSS Structures.

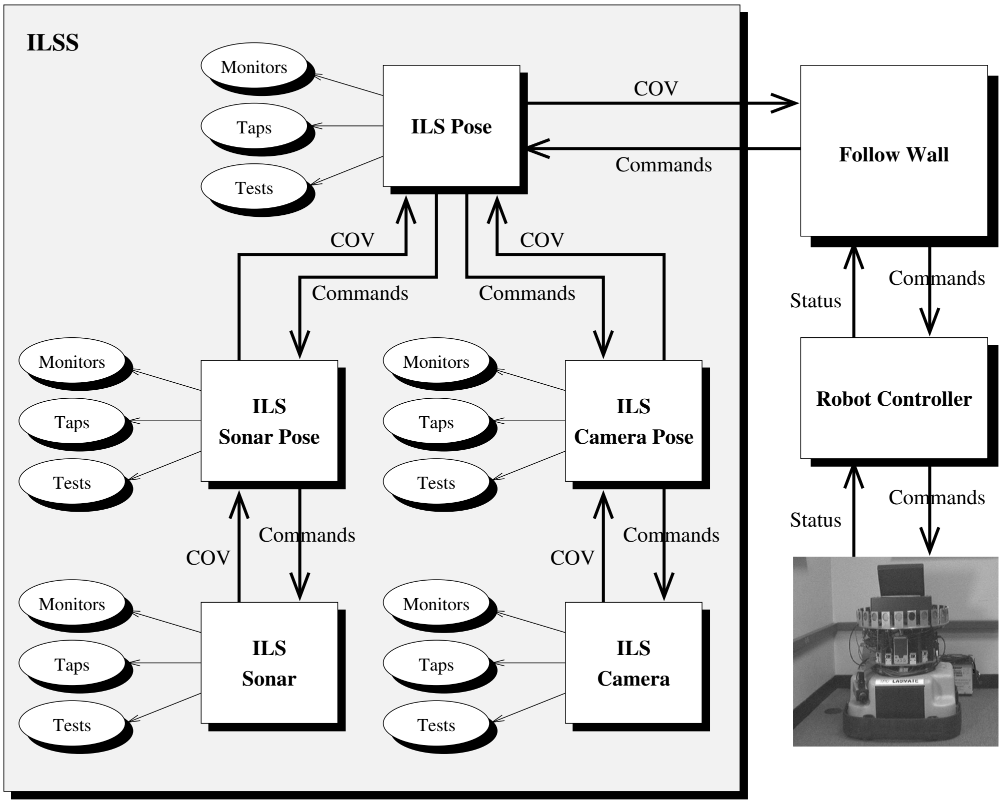

Figure 6: The wall-following system using sonars and a camera.

For example, if the sonar sensors are not working probably due to audio interference or damage, the system detects that through a certain monitor, and automatically switches to using the camera. Similarly, if the system detects any malfunction with the camera (e.g., lights off, occlusion, etc.) it switches to the sonars.

Having more than one independent means to get the same information increases the overall reliability of the system. This can be shown mathematically as follows: Let Pf_1 and Pf_2 be the probability of failure for two independent components, and R_1 and R_2 to be the reliability of the two components, respectively. We can define R_1 and R_2 in terms of Pf_1 and Pf_2 as follows:

$$R_1 = (1 - Pf_1) \times 100$$

$$R_2 = (1 - Pf_2) \times 100$$

The system will fail only when both components fail. This can happen with probability

$$Pf = Pf_1 * Pf_2$$

which is smaller than either of them. Therefore, the overall system reliability will be

$$R = (1 - Pf) \times 100$$

which is larger than either R_1 and R_2.

Using the above concept, the system can determine the reliability of the overall system at anytime given the reliability of each component and its current status.

5.1 Using Sonar Sensors

Two sonar sensors located on one side of the robot are used to determine the the orientation of the wall relative to the robot. We call it *ILS_Sonar_Pose*. It gets the sonar values from the *ILS_Sonar* component and generates one of three values: -1, 0, or 1 to represent if the robot should turn right, no turn, or left. The *ILS_Pose* component selects the sonar first since it is faster than the camera. If the *ILS_Sonar_Monitor* detects failure it reports that and the *ILS_Camera_Pose* is selected. In this case failure is detected if one of the sonars generates an out-of-range value.

5.2 Using the Camera

The camera is located on the same side as the two sonars. The orientation of the wall is determined using a horizontal dark line on the wall (we used electrical tape for that purpose). The idea is to find the image row number of both ends of the imaged line. By comparing the row numbers for both ends we can determine the way the robot should turn to be parallel to the wall. the *ILS_Camera_Pose* is used for that purpose. The *ILS_Camera_Monitor* checks for insufficient light or occlusion of the line.

5.3 Results

We started the experiment with both types of sensors working. The select function chooses the sonar first, then the camera. Figures 7 and 8 show two scenarios with two different initial orientations of the robot relative to the wall. In both scenarios we induced malfunction to the sonar (by covering it) to test the monitoring and debugging capabilities of the system. The system detected this malfunction and automatically switched to the camera. We also induced a malfunction to the camera (by turning off the lights) and the system detected that as well and started performing the appropriate embedded tests for both sensors to pinpoint the problem.

The following script shows parts of the system output while running the first experiment.

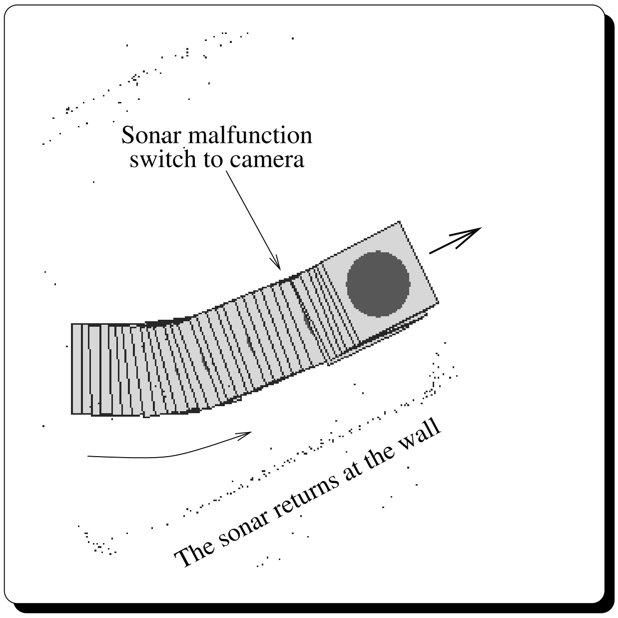

Figure 7: The first test run.

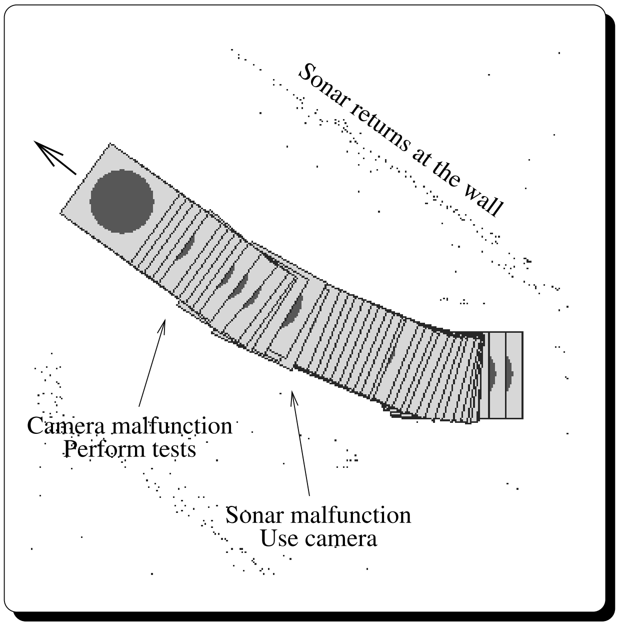

Figure 8: The second test run.

```
-- Start Initializations
-- Initialize ILS_Pose
-- In ILS_Sonar_Init
-- In ILS_Camera_Init
-- In Camera_Init
-- Start The Robot hardware

-- Starting main loop

TAP -- ILS_Sonar:
        Direction = 1, time = 0.236589 sec.
TAP -- ILS_Pose:
        Direction = 1, time = 0.243303 sec.
TAP -- ILS_Sonar:
        Direction = 1, time = 0.246920 sec.
TAP -- ILS_Pose:
        Direction = 1, time = 0.248939 sec.
 . . .

TAP -- ILS_Sonar:
        Direction = -1, time = 0.228644 sec.
Monitor -- ILS_Sonar:
        Sonar values out of range (541, 171, 174)

!!!!!! Switching to ILS_Camera !!!!!!

TAP -- ILS_Camera:
        Direction = 1, time = 0.004258 sec.
TAP -- ILS_Pose:
        Direction = 1, time = 0.246355 sec.
. . .

TAP -- ILS_Camera:
        Direction = 0, time = 0.003417 sec.
Monitor -- ILS_Sonar:
        Image too dark! Lights might be off!

-- Start embedded testing

-- In ILS_Pose_Test
```

```
TEST -- ILS_Sonar:
        Place the robot parallel to the wall at
        about 1 meter distance
        and press any key when ready ...

         --> Sonar 5 is out of range (137)

!!! ILS_Sonar_Test Failed !!!

TEST -- ILS_Camera:
        Place the robot parallel to the wall at
        about 1 meter distance
        and press any key when ready ...

-- Camera test is Ok!
```

6 Conclusion

In this report we presented 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. This framework was applied to several mobile robot applications, and a wall-following experiment was presented along with a discussion of the results.

Currently, we are working on building an ILSS library with several design tools which will assist in rapid prototyping of sensor systems and will provide an invaluable design tool for monitoring, analyzing and debugging robotic sensor systems.

Acknowledgment

We would like to thank Professor Robert Kessler and Professor Gary Lindstrom for their helpful discussions and suggestions.

References

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

[2] DEKHIL, M., AND HENDERSON, T. C. Instrumented sensor systems. In *IEEE International Conference on Multisensor Fusion and Integration (MFI 96), Washington D.C.* (December 1996), pp. 193–200.

[3] DEKHIL, M., AND HENDERSON, T. C. Optimal wall pose determination in a shared-memory multi-tasking control architecture. In *IEEE International Conference on Multisensor Fusion and Integration (MFI 96), Washington D.C.* (December 1996), pp. 736–741.

[4] DEKHIL, M., AND HENDERSON, T. C. Instrumented sensor system architecture. Tech. Rep. UUCS-97-011, University of Utah, March 1997.

[5] DEKHIL, M., AND SOBH, T. M. Embedded tolerance analysis for sonar sensors. In *Invited paper to the special session of the 1997 Measurement Science Conference,* Measuring Sensed Data for Robotics and Automation*, Pasadena, California* (January 1997).

[6] DEKHIL, M., SOBH, T. M., AND EFROS, A. A. Sensor-based distributed control scheme for mobile robots. In *IEEE International Symposium on Intelligent Control (ISIC 95), Monterey, California* (August 1995).

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

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

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

[11] 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).

[12] JOSHI, R., AND SANDERSON, A. C. Model-based multisensor data fusion: a minimal representation approach. In *IEEE Int. Conf. Robotics and Automation* (May 1994).

[13] KAPUR, R., WILLIAMS, T. W., AND MILLER, E. F. System testing and reliability techniques for avoiding failure. *IEEE Computer* (November 1996), pp.28–30.

[14] KIM, K. H., AND SUBBARAMAN, C. Fault-tolerant real-time objects. *Communications of the ACM 40*, 1 (January 1997), pp.75–82.

[15] LEE, D. C. *The map-building and exploration strategies of a simple sonar-equipped mobile robot.* PhD thesis, Cambridge University, 1996.

[16] LEE, D. C., AND RECCE, M. Quantitative evaluation of the exploration strategies of a mobile robot. *IJRR 16*, 4 (Aug. 1997), pp. 413–447.

[17] MURPHY, R. R., AND HERSHBERGER, D. Classifying and recovering from sensing failures in autonomous mobile robots. In *Proceedings of the AAAI-96* (Aug. 1996), pp. 922–929.

[18] NADIG, D., IYENGAR, S. S., AND JAYASIMHA, D. N. New architecture for distributed sensor integration. In *IEEE SOUTHEASTCON Proceedings* (1993).

[19] 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.

[20] PROFETA, J. A. Safety-critical systems built with COTS. *IEEE Computer* (November 1996), pp.54–60.

[21] 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.

[22] STEWART, D. B., AND KHOSLA, P. K. Mechanisms for detecting and handling timing errors. *Communications of the ACM 40*, 1 (January 1997), pp.87–93.

[23] 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.