A Framework for Distributed Sensing and Control[1]

Tom Henderson, Chuck Hansen, and Bir Bhanu

Department of Computer Science
The University of Utah
Salt Lake City, Utah 84112

Abstract

Logical Sensor Specification (LSS) has been introduced as a convenient means for specifying multi-sensor systems and their implementations. In this paper, we demonstrate how control issues can be handled in the context of LSS. In particular, the Logical Sensor Specification is extended to include a control mechanism which permits control information to (1) flow from more centralized processing to more peripheral processes, and (2) be generated locally in the logical sensor by means of a micro-expert system specific to the interface represented by the given logical sensor. Examples are given including a proposed scheme for controlling the Utah/MIT dextrous hand.

1. Introduction

Both the availability and need for sensor systems is growing, as is their complexity in terms of the number and kind of sensors within a system. But most robotic sensor-based systems to date have been designed around a single sensor or a small number of sensors, and *ad hoc* techniques have been used to integrate them into the complete system and for operating on their data. In the future, however, such systems must operate in a reconfigurable multi-sensor environment; for example, there may be several cameras (perhaps of different types), active range finding systems, tactile pads, and so on. In addition, a wide variety of such sensing devices, including mechanical, electronic, and chemical, are available for use in sensor systems, and a sensor system may include several kinds of sensing devices. Thus, at least three issues regarding the configuration of sensor systems arise:

1. How to develop a coherent and efficient treatment of the information provided by many sensors, particularly when the sensors are of different kinds.

2. How to allow for sensor system reconfiguration, both as a means of providing greater tolerance for sensing device failure, to permit dynamic allocation of sensing resources, and to facilitate future incorporation of additional sensing devices.

3. How to control the sensors.

We have previously proposed the Multi-sensor Kernel System [5, 7] and Logical Sensor Specification [4] as solutions for the first two problems, respectively. The rest of this paper gives our method for answering the third.

The purpose of the logical sensor specification is to permit an implementation independent description of the required data and the nature (type) of that data. In addition, alternative ways of producing the same output can be defined. This makes it possible to recover if some sensor fails. One can also choose an alternative based on higher level considerations (e.g., speed, resolution, etc.). Thus, a use for logical sensors is evident in any sensor system which is composed of several sensors, where sensor reconfiguration is desired, and/or where the sensors must be actively controlled.

As described in more detail elsewhere [4], the principal motivations for logical sensor specification are: the emergence of significant multi-sensor systems, the benefits of data abstraction, and the availability of smart sensors (thus, the substitution of hardware for software, and vice versa, should be transparent above the implementation level; see also Organick et. al [10]).

Logical sensors are then a means by which to insulate the user from the peculiarities of input devices. Thus, for example, a sensor system could be designed to deal with camera input, without regard to the kind of camera being used. In addition, logical sensor specification is also a means to create and package "virtual" physical sensors. For example, the kind of data produced by a physical laser range finder sensor could also be produced by two cameras and a stereo program. This similarity of output result is more important to the user than the fact that the information may be obtained by using one physical device, or by using two physical devices and a program. Logical sensor specification allows the user to ignore such differences of how output is produced, and treat equivalent means of obtaining data as logically the same.

Related work has been done in several areas. The need for some device-independent interactive system has resulted in the Graphical Kernel System (GKS) which is now a Draft International Standard. The main idea behind GKS is to provide "a means whereby interactive graphics

applications could be insulated from the peculiarities of the input devices of particular terminals, and thereby become portable" [11]. Some encouraging results reported in the robotics literature including a systematic study of robotic sensor design for dynamic sensing undertaken by Beni et al. [3]. Another related research effort is the programming environment (called the Graphical Image Processing Language) under development as part of the IPON project (an advanced architecture for image processing) at the University of Pennsylvania [2]. The hierarchical robot control system described by Albus [1] is a precursor to the logical sensor scheme proposed here.

2. Logical Sensor Specification

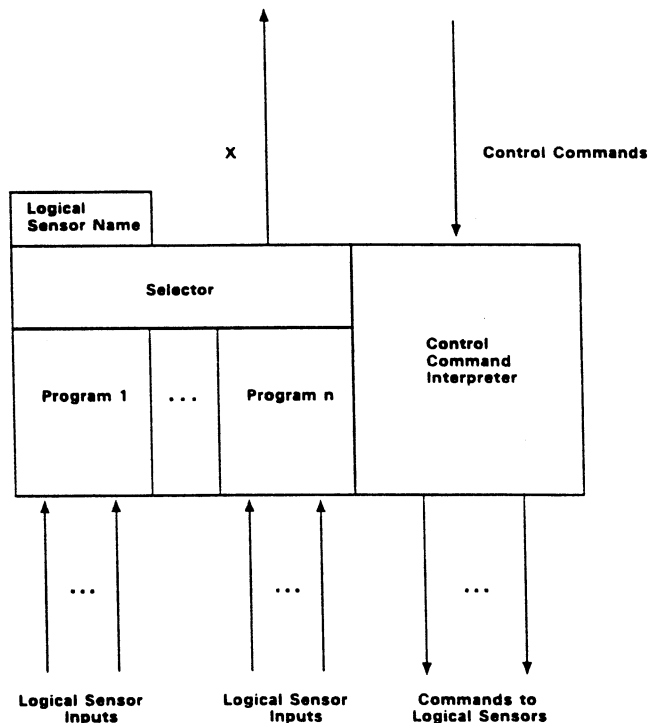Figure 1 shows the basic components of a logical sensor.

Figure 1. Logical Sensor Specification with Control

The **logical sensor name** uniquely identifies the logical sensor. The **characteristic output vector** is a vector of types which serves as a description of the output vectors that will be produced by the logical sensor. Thus, the output of a logical sensor is a set (or stream) of vectors, each of which is of the type declared by that logical sensor's characteristic output vector. Programs 1 to n represent alternative ways in which to obtain data with the

same characteristic output vector. Hence, each alternate subnet is equivalent, with regard to type, to all other alternate subnets in the list. Each alternate subnet in the list is itself composed of: (1) a set of **input sources**. Each element of the set must be a logical sensor (or the set may be empty). Allowing null input permits a physical sensor, which has only an associated program (the device driver), to be described as a logical sensor, thereby permitting uniformity of sensor treatment. (2) a **computation unit** over the input sources. Currently such computation units are software programs, but in the future, hardware units may also be used. Finally, the role of the **selector** (whose inputs are alternate subnets and an acceptance test name) is to detect failure of an alternate and switch to a different alternate. If switching cannot be done, the selector reports failure of the logical sensor.

In order to solve most recognition and manipulation problems, however, it is necessary to be able to reposition sensors (e.g., aim cameras) and adapt rapidly to changing conditions (e.g., if an object is slipping from the grasp of a robot hand, perhaps more force should be applied). Thus, in addition to a stream of sensed data flowing from physical sensors on up through some hierarchy of logical sensors, there may also be a stream of **control commands** (or signals) flowing in the reverse direction.

Each logical sensor has a **control command interpreter** to interpret the control commands coming from a level up in the hierarchy and to send commands down to logical sensors lower in the hierarchy. Moreover, the select function now plays a more sophisticated role in the logical sensor. Namely, the select function monitors both the sensor data going up and the command stream to be issued. Given the command (or commands) to be executed and the sensor data being produced locally, the select function is able to short circuit the path back to the root logical sensor and to modify the commands to be issued. Such a function may be viewed as a micro-expert system which knows all about the interface represented by the logical sensor in which it is located. Thus, a logical sensor acquires some of its meaning now not simply as a sensor/algorithm combination, but also as an interface between two layers of sensing and analysis.

Another requirement on the logical sensor is that it now also acts as a "logical controller." If the control command received at a particular sensor requires that control commands be sent to the source input logical sensors, then those commands will depend on which alternate subnet is currently selected by the selector function. For example, suppose range data can be obtained from a stereo camera system, a laser range finder system or a robot hand with tactile sensing. Then to obtain range data from a given region in space requires aiming and focusing two cameras, or aiming a camera and a laser, or positioning a robot arm. The high level command to scan a region must then be broken down into the appropriate lower level commands.

A logical sensor can be viewed as a network composed

of sub-networks which are themselves logical sensors. Communication within a network is controlled via the flow of data from one sub-network to another. Hence, such networks are data flow networks.

Once the logical sensors are specified, they are stored as s-expressions. In order to actually obtain an executable system from the logical sensor specification, it is necessary to translate the database expressions into some executable form, e.g., to produce source code for some target language, and then either interpret or compile and run that source. We currently have two implementations of the logical sensor specification language running: a C version (called C-LSS) running under UNIX, and a functional language version (called FUN-LSS) which produces FEL code (Function Equation Language) [9]. These have been described elsewhere [4]. C-LSS produces a UNIX shell script from the specification.

3. An Example

We are currently applying the methodology to some interesting and hard problems. In particular, we are developing and testing a specification for the UTAH/MIT Dextrous Hand. This gives us the opportunity to try out the method on a distributed multi-processor system, as the Hand is controlled by six M68000s.
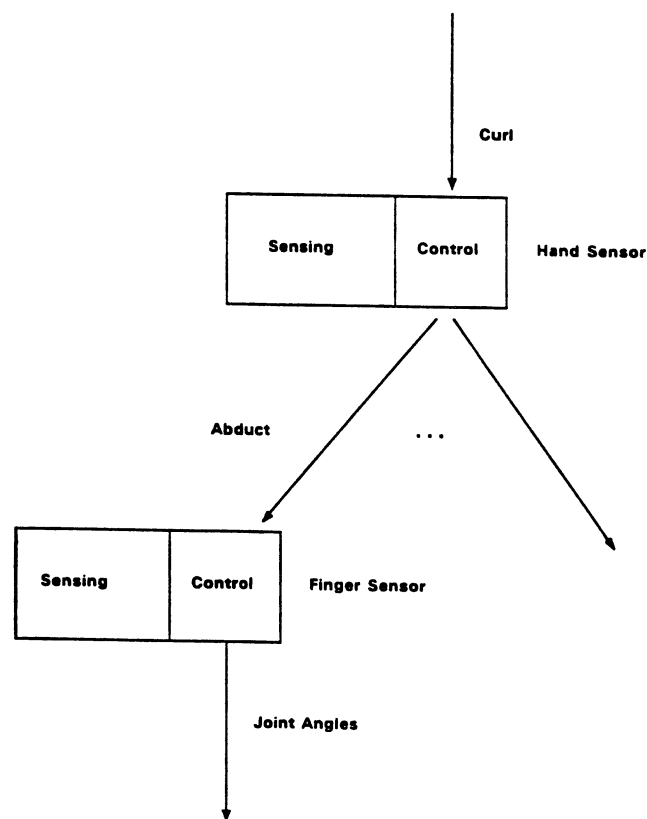
Shown in Figure 2 are some of the logical sensors which comprise the specification of a sensor and control scheme for the UTAH/MIT dextrous robot hand. The robot hand has four fingers each with four degrees of freedom [8]. The high level commands for hand control are interpreted as a set of commands to a lower-level right on down to the control of the joint positions of each finger which define the configuration of the robot hand.

A grasping action requires several hand operations, including the attainment of an approach configuration. One of these is the "curl" position (the control command to the hand logical sensor). To curl the hand requires that each finger move away from the median axis of the hand (the control command "abduct" to each of the finger logical sensors). Finally, the abduct command requires that each joint achieve a specific angle (the θ_{ij} control commands to the joint logical sensors). Thus, the feedback loop for position control can be located in the programs which are part of the joint logical sensor specification. Moreover, concise local knowledge for what to do in case of error conditions (slipping, too much force, etc.) can be embedded in the appropriate select function.

4. Conclusions

We have presented a framework for the specification of sensing and control systems. Moreover, the methodology lends itself nicely to distributed processing. The method permits the specification of fault tolerance (both software and hardware) and dynamic reconfiguration of the sensing system. The incorporation of control now permits closed loop operation and adaptation to changing conditions.

Our specific accomplishments include:

1. The development of a methodology for the specification of distributed sensing and control. In particular, one based on a reasonably well understood underlying computational model, i.e., dataflow.

2. The development of an operational environment for computing with respect to the methodology.

The successful implementation of such a methodology provides a very significant and fundamental tool for the specification of distributed sensing and control systems. Moreover, we believe that our approach permits an effective conceptual decomposition of the problem into manageable units. For a more complete treatment of logical sensors as a framework for distributed sensing and control (with detailed examples), see Henderson et. al [6].

Figure 2. Part of a Robot Hand Specification

References

[1]   Albus, J.
        Brains, Behavior and Robotics.
        BYTE Books, Peterborough, New Hampshire, 1981.

[2]   Bajscy, R.
        GRASP:NEWS Quarterly Progress Report.
        Technical Report Vol. 2, No. 2, The University of
            Pennsylvania, School of Engineering and Applied
            Science, 2nd Quarter, 1984.

[3]   Beni, G., S. Hackwood, L.A. Hornak and J.L. Jackel.
        Dynamic Sensing for Robots: An Analysis and
            Implementation.
        Robotics Research 2(2):51-60, Summer, 1983.

[4]   Henderson, T.C. and E. Shilcrat.
        Logical Sensor Systems.
        Journal of Robotic Systems 1(2):169-193, 1984.

[5]   Henderson, T.C. and Wu So Fai.
        MKS: A Multi-sensor Kernel System.
        *IEEE Transactions on Systems, Man, and
            Cyberbetics* SMC-14(5):784-791,
            September/October, 1984.

[6]   Henderson, T.C., C.D. Hansen, and Bir Bhanu.
        The Specification of Distributed Sensing and
            Control.
        Journal of Robotic Systems :to appear, 1985.

[7]   Henderson, Thomas C. and Wu So Fai.
        A Multi-sensor Integration and Data Acquisition
            System.
        In *Proceedings of the IEEE Conference on
            Computer Vision and Pattern Recognition,* pages
            274-280. IEEE, June, 1983.

[8]   Jacobsen, S., D.F. Knutti, K. Biggers, E.K. Iverson and
        J.E. Wood.
        An Electropneumatic Actuation System for the
            Utah/MIT Dextrous Hand.
        In *Proceedings of the Fifth CISM-IFToMM
            Symposium on Theory and Practice of Robots
            and Manipulators.* Udine, Italy, June, 1984.

[9]   Keller, R.M.
        FEL Programmer's Guide.
        Technical Report AMPS Tech. Memo 7, The
            University of Utah, Department of Computer
            Science, April, 1982.

[10]   Organick, E.I., M. Maloney, D. Klass and
        G. Lindstrom.
        *Transparent Interface between Software and
            hardware Versions of Ada Compilation Units.*
        Technical Report UTEC-83-030, University of Utah,
            Salt Lake City, Utah, April, 1983.

[11]   Rosenthal, D.S., J.C. Michener, G. Pfaff, R. Kessener
        and M. Sabin.
        The Detailed Semantics of Graphics Input Devices.
        Computer Graphics 16(3):33-38, July, 1982.

IJCAI
85

Proceedings of the Ninth International Joint Conference on Artificial Intelligence

18-23 August 1985, Los Angeles, California

Volume 2

IJCAI 85

Proceedings of the
Ninth International Joint Conference
on
Artificial Intelligence

Volume 2

August 18–23, 1985
sponsored by
**International Joint Conferences on
Artificial Intelligence, Inc.**
cosponsored and hosted by
**American Association for Artificial
Intelligence**
in cooperation with
University of California at Los Angeles