The Specification of Distributed Sensing and Control*

Tom Henderson, Chuck Hansen,† and Bir Bhanu
Department of Computer Science, The University of Utah, Salt Lake City, Utah 84112
Received February 8, 1985; accepted March 7, 1985

Logical Sensor System Specification (LSS) has been introduced as a convenient means for specifying multi-sensor systems and their implementations. In this article 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.

多重センサーシステム及びその実行のための論理的セイサーシステム仕様（ＬＳＳ）を先に提案したが、本論文ではＬＳＳにおいてどう制御が行なわれるか例示する。特に論理的センサ仕様が拡張され、（１）中央プロセスから周辺プロセスへの情報の流れや（２）与えられた論理的センサのインターフェースに関するエキスパートによって発生された情報の流れを制御する機構を具備している。実例としてユタ／ＭＩＴハンド制御のための方式などを含む。

I. INTRODUCTION

Multi-sensor systems are becoming much more common and require an appropriate system organization to efficiently and correctly integrate their data. We have previously proposed the Logical Sensor Specification (LSS) as one such methodology.[1] There we discussed two major issues:

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

*This work was supported in part by the System Development Foundation and NSF Grants ECS-8307483 and MCS-82-21750.

†Chuck Hansen is an ARO Fellow.

Journal of Robotic Systems, **2**(4), 387–396 (1985)

0471-2223/85/040387-10$04.00

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

In this article, we propose a solution to the crucial problem: How to control the sensors.

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. 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.

Logical sensors are then a means by which to insulate the user from the peculiarities of input devices, which in this case are (generally) physical sensors. Thus, for example, a sensor system could be designed to deal with camera input, without regard to the kind of camera being used. However, in addition to providing insulation from the peculiarities of physical devices, 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 similiarity 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.

A 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] In that system a graphics CRT will be used to interactively program an image analysis sequence. An operator will be able to use symbolic representations for common functions to configure particular applications. The GIPL system will produce technology independent process definitions and executable process modules. A separate system then will map these onto the physical architecture. Thus, the GIPL system is very similar in spirit to the logical sensor system described here.

II. LOGICAL SENSOR SPECIFICATION

A logical sensor comprises four parts: a logical sensor name, a characteristic output vector, alternate subnets, each of which is made up of a set of input sources and a computation unit over the input sources, and finally, a selector whose inputs are alternate subnets and an acceptance test name. The role of the selector 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.

A logical sensor can be viewed as a network composed of subnetworks which are themselves logical sensors. Communication within a network is controlled via the flow of data from one subnetwork to another. Hence, such networks are *data flow* networks. Figure 1 gives a pictorial presentation of this notion. The program of an alternate subnet accepts input from the source logical sensors, performs some computation on them, and returns as output a set (stream) of vectors of the type defined by the characteristic output vector.

It should be noted that there may be alternate input paths to a particular sensor, and these correspond to the alternate subnets. But even though there may be more than one path through which a logical sensor produces data, the ouput will be of the type declared by the logical sensor's characteristic output vector.

Logical sensor specifications are stored as *s*-expressions, the database. 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). C-LSS produces a UNIX shell script from the specification.

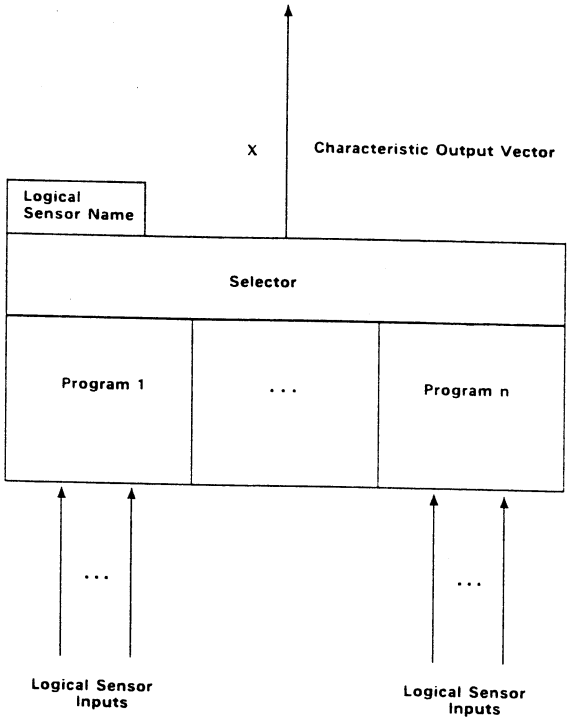

Figure 1. Graphical view of a logical sensor.

III. SPECIFICATION OF CONTROL IN LOGICAL SENSORS

The logical sensor specification system described so far provides a reasonable methodology for the specification of sensing systems. In this paper, we extend the scope of logical sensors to a form operationally and methodologically more appropriate for distributed sensing and control. 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, e.g., 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.

LSS provides a design methodology for sensor systems. We have seen that such a methodology is necessary due to the emergence of multi-sensor systems of great complexity. Moreover, many systems require that the sensors be widely distributed both physically and computationally. It may also be necessary to define a hierarchical relationship between the sensors and/or the algorithms applied to them. For example, a dextrous hand is made up of several fingers each of which is composed of several phalanges requiring position and tactile sensors. LSS exploits many well-known software design principles including abstraction, modularity, and separates implementation from specification. Finally, these features permit the dynamic reconfiguration of sensing resources for either fault tolerance or for a sort of adaptive sensing in response to the situation.

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. We therefore propose that the logical sensor schema be modified as shown in Figure 2.

Each logical sensor now has a program 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

Figure 2. Lo

controller." I
commands b
depend on w
exampl up
range
from a given
camera and a
region must t

Viewed in
knowledge re
transfer and
network. Fina
of the specifi
properties are

For exampl
comprise the
robot hand. T
high level cor
level right on
the configurat

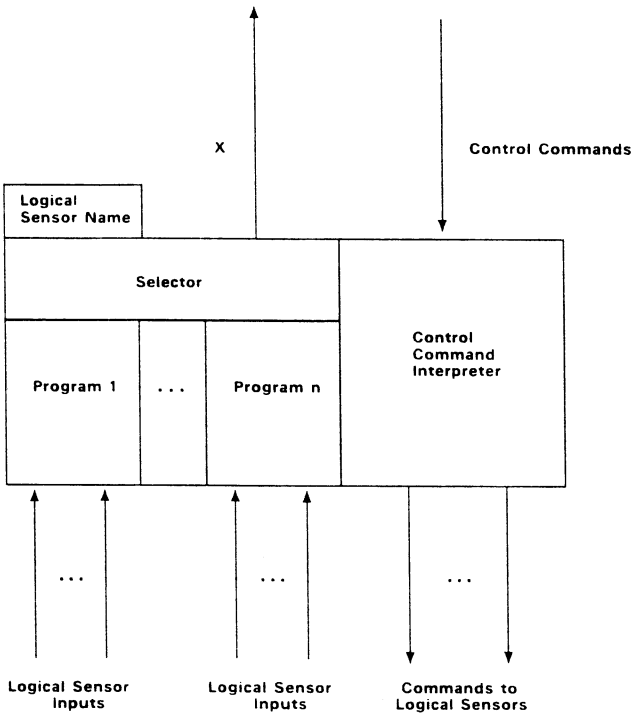

Figure 2. Logical sensor specification with control.

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.

Viewed in this way, it is possible that the selector function can learn some of the knowledge required to successfully administer its task. Also, it becomes possible to transfer and coordinate logical sensors at the same relative level and position in a network. Finally, it is possible for some very high level system (which has knowledge of the specification of a given selector) to modify the selector. We believe that these properties are crucial for successful sensor system schemes.

For example, consider Figure 3. Shown here 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 4 degrees of freedom.[3] 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.

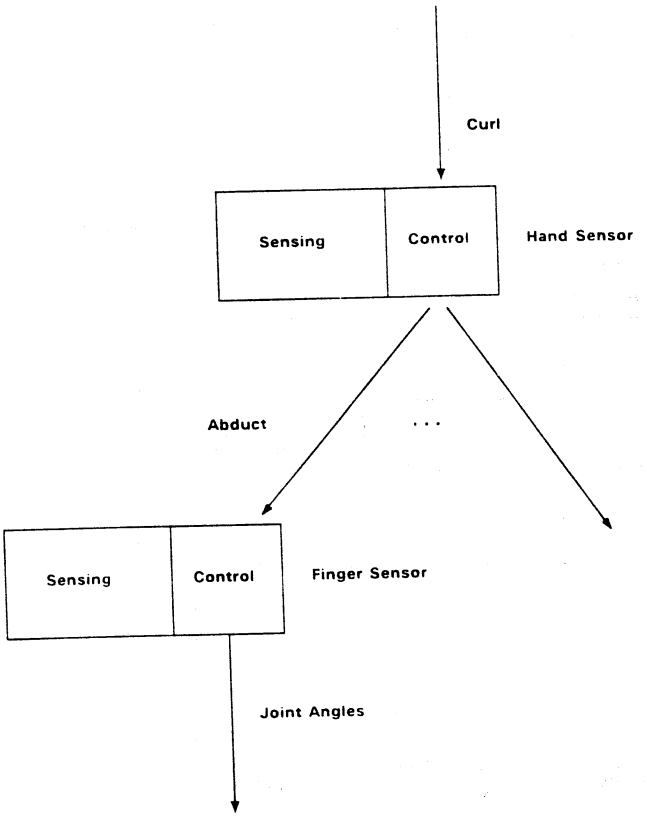

Figure 3. Part of a robot hand specification.

The hierarchical robot control system described by Albus[4] is a precursor to the scheme proposed here. However, the two differ in several ways:

- Dataflow networks define the organizational hierarchy of logical sensors.
- Each logical sensor combines the features of the sensory processing, world model, and task decomposition modules of the computational hierarchy of Albus.
- Logical sensors permit dynamic reconfiguration due to either high-level reasons (an alternate subnetwork is chosen due to some relevant feature, e.g., its speed) or report of failure from some lower-level sensor.
- Any language can be chosen to define the control interface between two logical sensors.
- Some form of production system is used to select the control activity as well as the alternate subnetwork.

Thus, Albus' system can be specified using a restricted logical sensor system with control.

IV. A DETAILED EXAMPLE

With logical sensor specifications, it is possible to develop either general image analysis functions (e.g., edge detectors, texture analyzers, etc.) as logical sensors or to describe logical sensors for very specific objects or shapes. In this example, we give the complete description of a logical sensor which, when invoked, returns the position of a fire extinguisher in a room. Any firefighting robot could use such a capability. Figure 4 gives a graphical version of the logical sensors involved in the definition of the "fire extinguisher finder" logical sensor.

The top level logical sensor, "fire extinguisher finder," returns a characteristic output vector comprised of the direction of the camera and the centroid of the fire extinguisher. The only control parameter for this logical sensor is the direction of the camera. This logical sensor produces its result by processing the output of two other logical sensors: a red object detector, and a feature detector. A fire extinguisher is assumed present if enough red blobs are adjacent to a feature.

The red object detector, "red," has a characteristic output vector of the centroid and area of any reasonably large red object. The control parameters consist of a camera

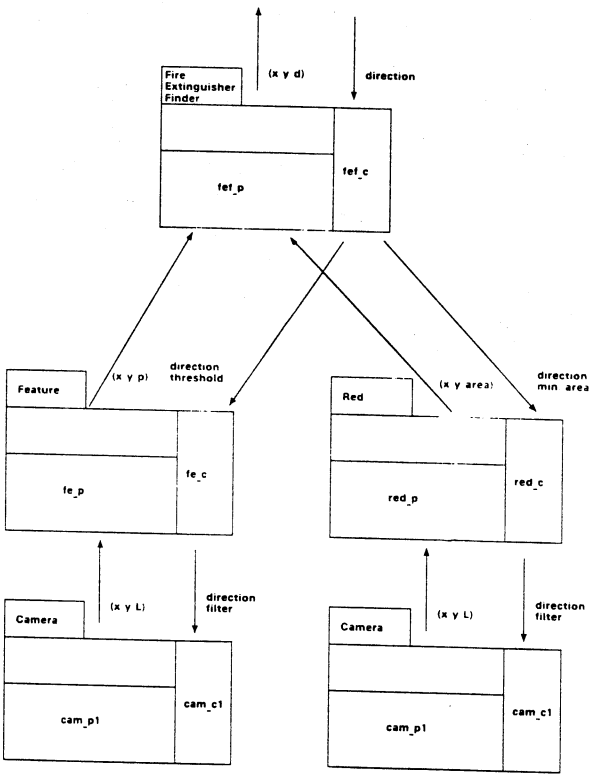

Figure 4. Logical sensor network for the fire extinguisher finder.

Figure 5. Camera output to "feature."

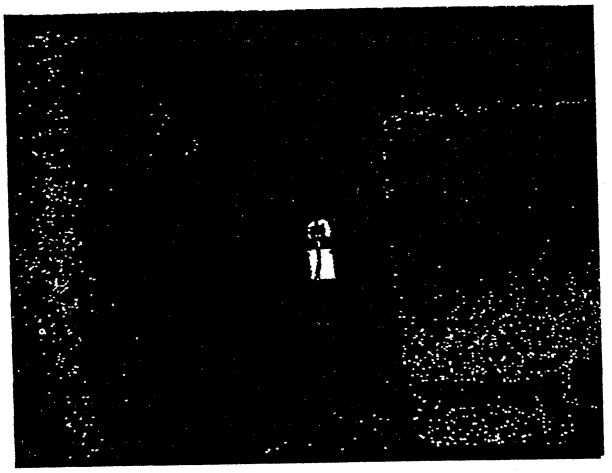

Figure 6. Camera output to "red."

```
(fire extinguisher finder ((fef_p fef_c  (red feature)))
     (× 0..511  y 0..479  direction 0..360  likelihood 0.0..100.0))

(feature ((fe_p fe_c (camera)))
     (× 0..511  y 0..479  likelihood 0.0..100.0))

(red ((red_p red_c (camera)))
     (× 0..511  y 0..479  area integer))

(camera ((cam_p1 cam_c1 ( )))
     (× 0..512  y 0..480  intensity 0..255))
```

Figure 7. The *s*-expressions for the logical sensors.

Table I. Output vectors from "red."

x mean	y mean	Area	Control Parameters
279	199	218	direction: 15°
267	206	59	min area: 30 pixels
278	240	34	

Table II. Output vectors from "feature."

x mean	y mean	likelihood	Control Parameters
281	187	95.8%	direction: 15°
269	213	98.8%	likelihood: 94.3%
269	220	94.3%	
93	432	94.7%	
93	443	94.3%	

direction and a threshold for the minimal size of the object. The feature detector, "feature," actually is a general purpose logical sensor which detects horizontal lines (or bands) of a certain breadth. It returns the centroid of the band and the likelihood that it is a band of the right kind. It takes as control parameters a direction for the camera and a threshold for the minimum acceptable likelihood.

Finally, both "red" and "feature" use a logical sensor called "camera." The result from "camera" is a 2D image, while the control parameters are direction to aim the camera and a filter color to use (red, blue, green, and none). "Red" always requires a red filter, while "feature" requests none.

Figure 5 shows an image from "camera" which was returned to "feature" and Figure 6 shows an image from "camera" (in the same direction) which was returned to "red." The output vectors from "red" are given in Table I, and the output vectors from "feature" are given in Table II. Finally, the output vector from "fire extinguisher finder" is given in Table III. The *s*-expressions for all these logical sensors are given in Figure 7. A fire extinguisher is detected since the three red blobs are all adjacent to feature 2, and this constitutes enough evidence. On other images of the room where no fire extinguisher was present, none was detected by the logical sensor. The template for a logical sensor *s*-expression is:

Table III. Output vectors from "fire extinguisher finder."

x mean	y mean	Direction	Control Parameter:
269	213	15°	direction: 15°

(logical sensor name ((computation—unit control—unit (input sources))) (characteristic output vector)),

where the characteristic output vector is a sequence of variable/type pairs.

V. CONCLUSIONS AND FUTURE RESEARCH

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.

We are currently exploring several aspects of LSS. In the present version of LSS, we treat the control interface as a separate function of the logical sensor. Each control command interpreter is designed using standard parser tools (e.g., lex and yacc under UNIX). We must determine the relation between the type language and the physical nature of the control at a given level in the hierarchy.

We must also determine the exact role and content of the select function. How should its local expertise be represented? (Another interesting side problem is how learning can be achieved.) The major problem is to give only the select function knowledge relevant to its interface. Much of the work has been done, however, in that the select function depends directly on: (1) the sensed data coming in from the alternate subnetwork currently selected, (2) the result of the acceptance test, and (3) the commands being interpreted by the control part of the logical sensor.

Finally, we must apply the methodology to some interesting and hard problems. In particular, we intend to develop and test a specification for the UTAH/MIT Dextrous Hand. This will also give us a chance to try out the method on a distributed multiprocessor system as the Hand is controlled by four (ultimately six) M68000s. We will develop the controllers bottom-up, starting with the finger controllers. These will then be integrated with a hand controller, and finally, this will be integrated with an arm controller. The construction of the Hand is already complete, and tactile sensors are currently being installed. We will first develop the logical sensors for the Hand off-line, then try it out as the hardware is available.

References

1. T. C. Henderson and E. Shilcrat, "Logical Sensor Systems," *J. Robotic Syst.*, **1(2)**, 169–193 (1984).
2. R. Bajscy, *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. S. Jacobsen, 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.
4. J. Albus, *Brains, Behavior and Robotics*, BYTE Books, Peterborough, New Hampshire, 1981.