Sensor-based Distributed Control Scheme for Mobile Robots

Mohamed Dekhil, Tarek M. Sobh, and Alexei A. Efros*

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

Abstract

In this paper we present a sensor-based distributed control scheme for mobile robots. This scheme combines centralized and decentralized control strategies. A server-client model is used to implement this scheme where the server is a process that caries out the commands to be executed, and each client is a process with a certain task. The clients are running in parallel issuing commands to the server which selects the command to be executed based on a predefined priority scheme. In this scheme, a collision avoidance client is running all the time with the highest priority. This improves the performance of the other clients since it removes the burden of avoiding obstacles and allows each client to concentrate on performing its task. The logical sensor approach is used to model the sensory system which provides different levels of data representation with tolerance measures and analysis. The simulation results of this model are presented with a brief discussion and conclusion on these results.

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 uncertainties. The main component in such systems is the transformation of sensor outputs to the decision space, then the computation of the error signals and the joint-level commands (see Figure 1). For example, the sensor readings might be the current tool position, the error signal the difference between the desired and current position at this moment, and finally, the joint-level command will be the required actuator torque/force.

The sensors used in the control scheme shown in Figure 1 are considered to be passive elements that provide raw data to a central controller. The central controller computes the next command based on the required task and the sensor readings. The disadvantage of this scheme is that the central controller may become a bottleneck when the number of sensors increases which may lead to longer response time. In some applications the required response time may vary according to the required task and the environment status. For example, in an autonomous mobile robot with the task of reaching a destination position while avoiding unknown obstacles, the time to reach to the required position may not be important, however, the response time for avoiding obstacles is critical and requires fast response. Fast response can be achieved by allowing sensors to send commands directly to the physical system when quick attention is required.

In this work, several controllers (clients) are working in parallel, competing for the server. The server selects the command to be executed based on a dynamically configured priority scheme. Each of these clients has a certain task, and can use the sensor readings to achieve its goal. A special client with the task of avoiding obstacles is assigned the highest priority. The clients need to know the current state of the system and the command history to update their control strategy, therefore, the server has to broadcast the selected command and the current state of the system.

The logical sensor approach, which we used to model the sensory system in our mobile robot, allows flexible and modular design of the controllers. It also provides several levels of data abstraction and tolerance analysis based on the sensor type and the required task. The initial work on this project is described in [5]. This approach is used to build high-level requests which may be used by the application programs.

2 Related Work

There has been a tremendous amount of research in the area of sensor-based control, including sensor modeling, multisensor integration, and distributed control schemes for robotic applications in general and mobile robots in particular. A sensor-based control using a general learning algorithm was suggested by Miller [10]. This approach uses a learning controller that learns to reproduce the relationship between the sensor outputs and the system command variables. Another technique for sensor-based obstruction avoidance for mobile robots was proposed by Ahluwalia and Hsu [1]. In their technique, the robot is able to move through an unknown environment while avoiding obstacles. Simulations were carried out assuming the robot had eight tactile sensors and the world is modeled as a two-dimensional occupancy matrix with 0's representing empty cells and 1's representing occupied cells. Several research activities for sensor-

*This work was supported in part by The Advanced Research Projects Agency under Army Research Office grant number DAAH04-93-G0420, and NSF grant CDA 9024721. All opinions, findings, conclusions or recommendations expressed in this document are those of the author and do not necessarily reflect the views of the sponsoring agencies.

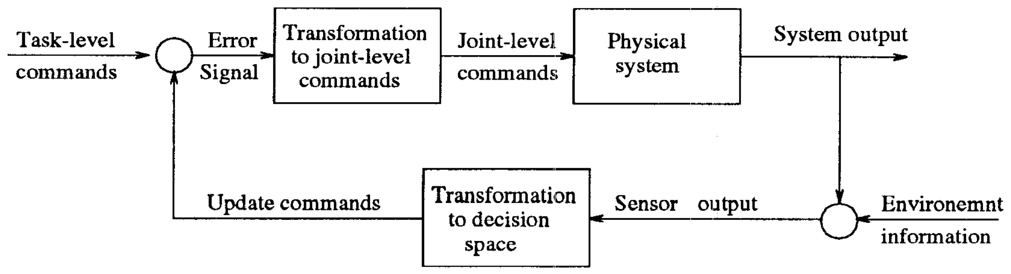

Figure 1: Closed loop control system.

based control for robotic applications can be found in [8].

Luo and Kay [9] conducted a survey on multisensor-based mobile robots. In their survey, the presented a number of control strategies that has been used in this area.

Brooks proposed a new architecture for controlling mobile robots [2, 3]. In this architecture, layers of control system are built to let the robot operate at increasing levels of competence. These layers are built as concurrent modules that communicate over low-bandwidth channels.

The idea of *smart sensing* was investigated by several researchers. Yakovleff et al. [15] represented a dual purpose interpretation for sensory information; one for collision avoidance (reactive control), and the other for path planning (navigation). The selection between the two interpretation is dynamic depending on the positions and velocities of the objects in the environment. Budenske and Gini [4] addressed the problem of navigating a robot through an unknown environment, and the need for multiple algorithms and multiple sensing strategies for different situations.

Discrete Event Systems (DES) is used as a platform for modeling the robot behaviors and tasks, and to represent the possible events and the actions to be taken for each event. A framework for modeling robotic behaviors and tasks using DES formalism was proposed by Košecká et al. [7]. In this framework, there are two kinds of scenarios. In the first one, reactive behaviors directly connects observations (sensor readings) with actions. In the second, observations are implicitly connected with actions through an observer.

In our proposed control scheme, the sensory system can be viewed as passive or *dumb* element which provides raw data. It can be viewed as an *intelligent* element which returns some "analyzed" information. Finally it can be viewed as a *commanding* element which sends commands to the physical system. Each of these views is used in different situations and for different tasks. A detailed description of the proposed control scheme is presented in the following section.

3 The Proposed Control Scheme

The robot behavior can be described as a function $\mathcal{F}$ that maps a set of events $\mathcal{E}$ to a set of actions $\mathcal{A}$. This can be expressed as:

$$\mathcal{F}\colon \mathcal{E} \longrightarrow \mathcal{A}$$

The task of the robot controller is to realize this behavior. In general we can define the controller as a set of pairs:

$$\{(e_1, a_1), (e_2, a_2), \ldots, (e_n, a_n)\}$$

where $e_i \in \mathcal{E}$, and $a_i \in \mathcal{A}$

The events can be defined as the interpretation of the raw data perceived by the sensors. Let's define the function $\mathcal{T}$ which maps raw data $\mathcal{R}$ to events $\mathcal{E}$:

$$\mathcal{T}\colon \mathcal{R} \longrightarrow \mathcal{E}$$

The functions $\mathcal{T}$ and $\mathcal{F}$ can be closed form equations, lookup tables, or inference engine of an expert system. This depends on the kind of application and the complexity of each transformation.

3.1 Abstract Sensor Model

We can view the sensory system using three different levels of abstractions (see Figure 2.)

1. **Dumb sensor:** which returns raw data without any interpretation. For example, a range sensor might return a real number representing the distance to an object in inches, and a camera may return an integer matrix representing the intensity levels of each pixel in the image.

2. **Intelligent sensor:** which interprets the raw data into an event using the function $\mathcal{T}$. For example, the sensor might return something like "will hit an object," or "a can of Coke is found."

3. **Controlling sensor:** which can issue commands based on the received events. for example, the sensor may issue the command "stop" or "turn left" when it finds an obstacle ahead. In this case, the functions $\mathcal{F}$ and $\mathcal{T}$ should be included in the abstract model of the sensor.

The dumb sensor can be used as a source for the feedback information required by the control system. It can be also used to gather measurements to construct a map for the surrounding environment. The process that uses a dumb sensor as a source of information needs to know the type of that sensor, the format of the data the sensor returns, and the location of the sensor, to be able to interpret the perceived data. The intelligent sensor may be used for monitoring activities. The process that uses an intelligent sensor. needs to know only

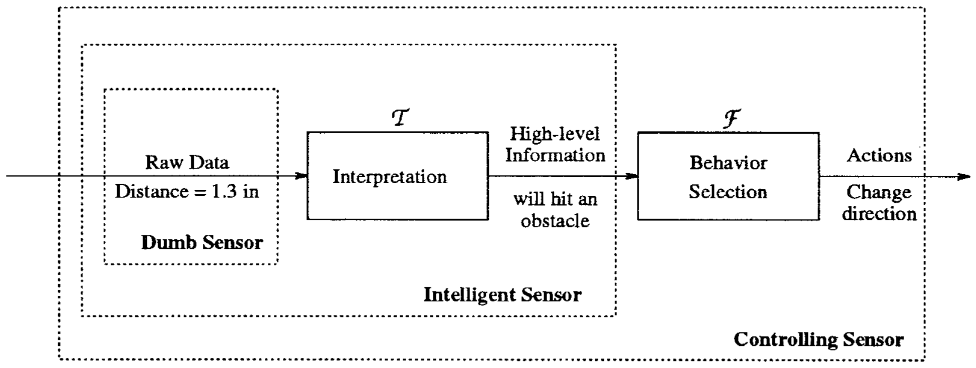

Figure 2: Three levels to view a sensor module.

the event domain and maybe the location of the sensor. On the other hand, the commanding sensor is considered to be a "client" process that issues commands to the system.

3.2 A Distributed Control Architecture

Several sensors can be grouped together representing a logical sensor [6, 12]. We will assume that each logical sensor is represented as a client process which sends commands through a chanel to a multiplexer (the server process) which decides the command to be executed first. Besides these logical sensors, we might have other processes (general controllers) that send commands to the server process to carry out some global goals. Figure 3 shows a schematic diagram for the proposed control scheme.

Let's call any process that issues commands to the server a *client process*. In this figure, there are three types of clients:

1. Commanding sensors, that are usually used for reaction control and collision avoidance.
2. General Controllers, that carry out a general goal to be achieved (e.g., navigating from one position to another.)
3. Emergency exits, which bypass the multiplexer in case of emergencies (e.g., emergency stop when hitting an obstacle.)

In most cases, the general controllers require feedback information to update their control parameters. This information is supplied by dumb sensors in form of raw data, or by intelligent sensors in form of events. On the other hand, a monitoring process might use only intelligent sensors as a source of "high-level" events instead of raw data. All clients (except for the emergency exists) send the commands to a multiplexer. The multiplexer selects the command to be executed based on a priority scheme which depends on the current state of the system and the type of operation the client is performing. Once a command is selected, all other commands can be ignored, since the state of the system will change after executing the selected command.

The low-level controller, shown in Figure 3, translates the high-level commands into low-level instructions which drive

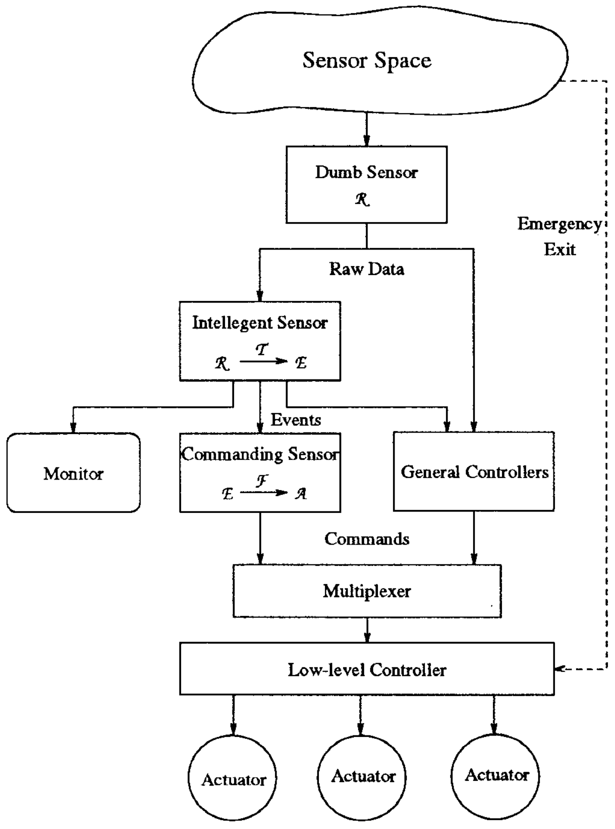

Figure 3: The proposed control scheme.

the system's actuators. The low-level controller receives its commands either form the multiplexer or from an emergency exit. After the command is executed, the system state is updated, and the sensor space is changed. New sensor readings are received and the cycle is repeated.

3.3 Communication Protocols

In the proposed control scheme, there are several clients sending commands asynchronously to the server. Therefore, we need to define a communication protocol to organize these commands, and to set a priority scheme for selecting the command to be executed first. In most cases, the clients need to know the current state of the system and the command history to update their control strategy. Therefore, the server has to broadcast the selected command and the current state of the system.

Each client may send commands to the server (through multiplexer) at any time. Each command is associated with the signature of the sender. This signature includes the name and type of the sender, and the priority value. In most cases, the reaction commands (usually from a commanding sensor to avoid collision) has a higher priority than any other client. The priority among the client may be specified by the user and/or by the current state of the system. Emergency exits should always bypass the multiplexer and send their commands directly to the low-level controller.

The message passing paradigm is used for process communication. This allows processes to be running on different platforms without the need for shared memory. In our implementation, MPI, Message-Passing Interface [14] was used because of its portability and to workstation clusters and heterogenous networks of workstations. It also provides an easy-to-use library functions to carry out the required communication protocols.

4 Experiments and Simulation Results

A simulator called *XSim* has been developed to examine the applicability of the proposed control scheme. This simulator is based on a mobile robot called "LABMATE" designed by Transitions Research Corporation [13]. This simulator displays the robot on the screen and accepts actual LABMATE commands like *go, turn, read-sonars,* etc. In this environment, moving from the simulation to the real robot is simply a matter of compiling the driver program with the LABMATE library rather than the simulation library.

The LABMATE was used for several experiments at the Department of Computer Science, University of Utah. It also entered the 1994 AAAI Robot Competition [11]. For that purpose, the LABMATE was equipped with 24 sonar sensors, eight infrared sensors, a camera and a speaker. [1] Figure 4 shows the LABMATE with its equipment.

[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.

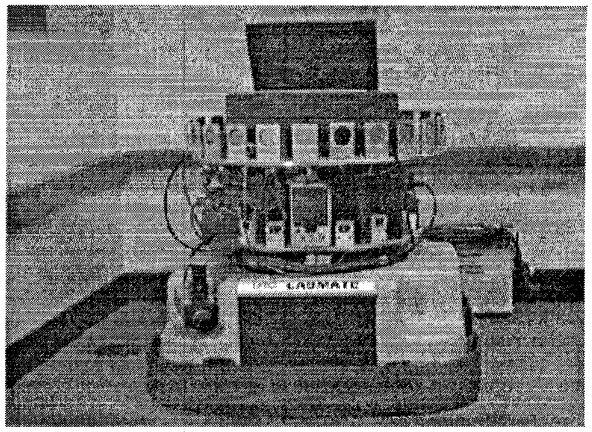
Figure 4: The LABMATE robot with its equipments.

4.1 Modeling the System

The sensors in the old scheme are used only as dumb sensors, while in the proposed scheme, sensors are used in three different levels. They are used as dumb sensors to provide feedback information for a general navigator. They are also used as intelligent sensors providing information to a monitoring process (e.g., a speaker as an output device.) Finally they are used as commanding sensors (clients) for collision avoidance. The emergency exits are hardware bumpers that command the robot to stop if it touch any object. There is also a general controller for navigation and map construction.

4.2 The Priority Scheme

In this system, there are several clients for the server. Beside these clients, there are two emergency exits represented by two bumpers, one on the front and one on the back. As mentioned before, emergency exits do not compete for the server, rather it sends its commands directly to the low-level controller.

The priority scheme in our application is set by each client as a number from 1 to 10, with 1 as the highest priority. Normally, 1 is reserved for the collision avoidance client. The server checks for the priority associated with each command, and executes the command with the highest priority while notifying the "losers" which command was executed. If two commands with the same priority arrive at the same time, the server arbitrarily selects one of them and ignores the other.

Commands that were not selected are cleared since the state of the robot has been changed after executing the command with the highest priority.

4.3 Simulation Results

Several experiments were performed on the simulator to check the applicability and validity of the proposed control scheme, and the results were very encouraging. The following is a description of three of these experiments along with the output of the simulation showing the portion of the commands that were selected and the trajectory of the robot during each experiment.

Experiment (1)

This was the first experiment performed to demonstrated the applicability of this control scheme. In this experiment, two clients were running simultaneously; the collision avoidance client, and a simple navigator which always commands the robot to move forward. The collision avoidance has priority 1, which is the highest priority, and the navigation client has priority 9. The following shows part of the output printed during this experiment which shows the commands that has been executed by the server.

```
Collision Avoidance: client #1.
Simple Navigation: client #2.
Server Starts as process #0.

* Accepted RESET from 1 *
- Rejected RESET from 1 *
* Accepted GO-FRWD from 2 *
* Accepted GO-FRWD from 2 *
* Accepted GO-FRWD from 2 *
* Accepted GO-FRWD from 2 *
* Accepted GO-FRWD from 2 *
* Accepted TURN-LEFT from 1 *
- Rejected GO-FRWD from 2 -
* Accepted TURN-LEFT from 1 *
- Rejected GO-FRWD from 2 -
* Accepted GO-FRWD from 2 *
* Accepted GO-FRWD from 2 *

          . . .
```

Figure 5 shows the trajectory of the robot in the lab environment.

Experiment (2)

In the second experiment, we added another goal-directed client which tries to move the robot to a certain goal location. This client has priority 5 which is higher than the simple navigator process. This new client sends commands to the server to update the direction of the robot such that it moves towards the goal location. In this experiment, the initial and the final points were chosen such that there are some obstacles between them. Figure 6 shows the robot trajectory for this experiment from the initial location to the goal location. Notice that at several points, the collision avoidance client took over and moved the robot away from the obstacles, then the new client updates the direction towards the goal point.

Experiment (3)

In the third experiment, we replaced the goal-directed client with a door-finding client. This new client tries to find

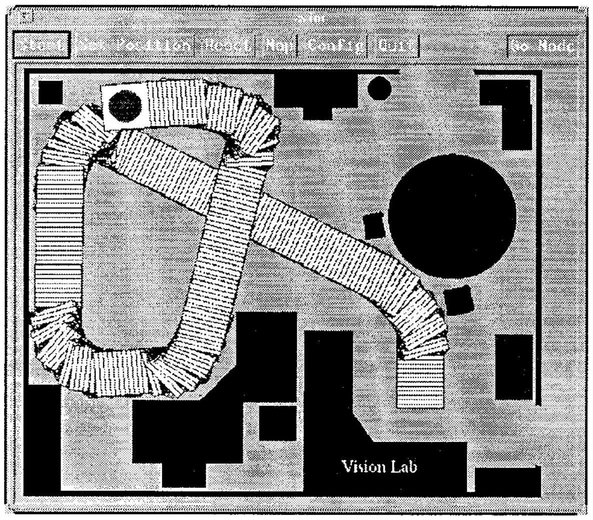

Figure 5: The trajectory of the robot for experiment (1).

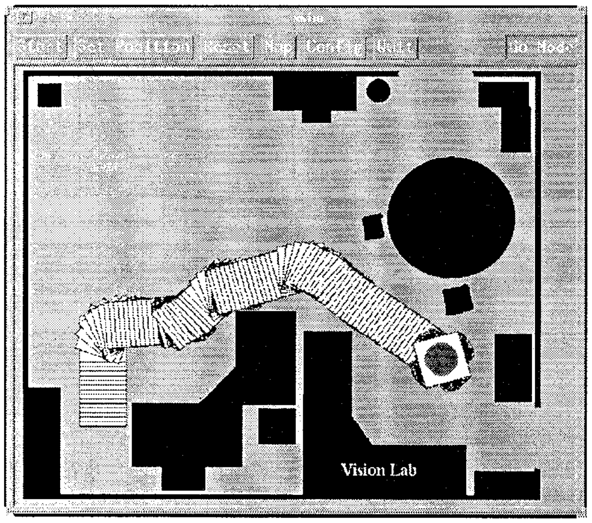

Figure 6: The trajectory of the robot from the initial to the goal point.

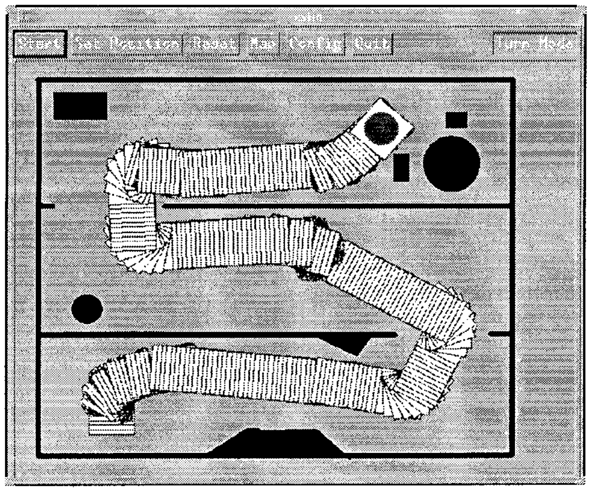
Figure 7: The trajectory of the robot while moving through open doors.

open doors and direct the robot to go through these doors. Finding doors using sonar sensor is very hard and problematic, and there is a lot of research in this area. For this experiment we used a very crude algorithm and a simple hallway structures just to demonstrate the capabilities of the proposed control scheme. Figure 7 shows the robot trajectory while moving in a hallway environment with two open doors at different places.

5 Conclusion and Future Work

In this paper, a distributed sensor-based control scheme was proposed. In this scheme, each sensor can be viewed with three different levels of abstraction; *dumb sensors* which provide raw data, *intelligent sensors* which provides high level information in a form of events, and finally, *commanding sensors* which can issue commands representing a reaction behavior for the system. Commands can be issued by different processes called *clients*. Each client may issue commands at any time, and a multiplexer (the server) selects the command to be executed. A priority scheme has to be defined as a bases for selection. Examples for applying this control scheme to a mobile robot were described along with the simulation results. We believe that this scheme provides for more flexible and robust control systems, and allows more modular design for the whole control system. It also provides fast response for reaction behavior which is an essential requirement in real-time systems.

The next step to this work is to implement a distributed controller for the "real" LABMATE using the proposed control scheme. A more detailed decision function for the logical sensors may be defined and the communication protocols among the sonar sensors needs to be explicitly defined. Higher level functions for increasing the accuracy of the measured point locations is to be defined.

References

[1] AHLUWALIA, R. S., AND HSU, E. Y. Sensor-based obstruction avoidance technique for a mobile robot. *Journal of Robotic Systems 1*, 4 (Winter 1984), pp. 331–350.

[2] BROOKS, R. A. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation RA-2*, 1 (March 1986), pp. 14–23.

[3] BROOKS, R. A. A hardware retargetable distributed layered architecture for mobile robot control. In *IEEE Int. Conf. Robotics and Automation* (1987), pp. 106–110.

[4] BUDENSKE, J., AND GINI, M. Why is it difficult for a robot to pass through a doorway using altrasonic sensors? In *IEEE Int. Conf. Robotics and Automation* (May 1994), pp. 3124–3129.

[5] DEKHIL, M., GOPALAKRISHNAN, G., AND HENDERSON, T. C. Modeling and verification of distributed control scheme for mobile robots. Tech. Rep. UUCS-95-004, University of Utah, April 1995.

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

[7] KOŠECKÁ, J., AND BOGONI, L. Application of discrete event systems for modeling and controlling robotic agents. In *IEEE Int. Conf. Robotics and Automation* (May 1994), pp. 2557–2562.

[8] LEE, C. S. G. *Sensor-based robots: algorithms and architecture.* Springer-Verlag, 1991.

[9] LUO, R. C., AND KAY, M. G. *Multisensor integration and fusion for intelligent machines and systems.* Ablex Publishing Corporation, 1995.

[10] MILLER, W. T. Sensor-based control of robotic manipulators using a general learing algorithm. *IEEE Journal of Robotics and Automation* (Nov. 1987), pp. 157–165.

[11] 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.

[12] SHILCRAT, E. D. Logical sensor systems. Master's thesis, University of Utah, August 1984.

[13] TRC TRANSITION RESEARCH CORPORATION. *LABMATE user manual, version 5.21L-f.*, 1991.

[14] UNIVERSITY OF TENNESSEE, KNOXVILLE. *MPI: a message-passing interface standard.*, May 1994.

[15] YAKOVLEFF, A., NGUYEN, X. T., BOUZERDOUM, A., MOINI, A., BOGNER, R. E., AND ESHRAGHIAN, K. Dual-purpose interpretation of sensory information. In *IEEE Int. Conf. Robotics and Automation* (1994).