Sensing Under Uncertainty For Mobile Robots

Tarek M. Sobh, Mohamed Dekhil, and Alyosha A. Efros

Department of Computer Science and Engineering
School of Science, Engineering and Technology
University of Bridgeport
Bridgeport, CT 06601, USA

Abstract

In this work we present a control strategy under uncertainty for mobile robot navigation. In particular, we implement a server-client model, where the server executes the commands and the clients run in parallel, each performing its tasks. Tolerance analysis is performed to incorporate sensing uncertainties into the proposed model. The sensory system is depicted with a framework that allows different levels of data representation, based on the robust modeling of the sensing uncertainties.

Keywords: Mobile Robots, Uncertainty Modeling, Distributed Control, Sensing.

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. 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 described control scheme 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. By response time we mean the time between two consecutive commands. In some applications the required response time may vary according to the required task and the environment status. For example, in 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. This is analogous to human reactions to some events. In the normal cases, the sensory systems in humans (e.g., eye, ear, nerves, etc.) sends perceived data to the brain (the central controller) which analyze this data and decides the next action to be taken based on the result of the analysis and the required task to be done. However, humans have a very fast contracting reaction when touching hot surfaces for example. In such cases, this reaction behavior is due to commands sent directly from the nerves at the skin spot where the touch occurred to the muscles, bypassing the brain.

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 needs 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.

Another aspect of this work is incorporating tolerance analysis and measures into the used sensory system. This provides quantitative measures for the accuracy of the location of measured points. It also serves as the basis for devising sensing strategies to enhance the measured data for localization and map construction.

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 [1]. This approach is used to build high-level requests which may be used by the application program. These requests include measuring data points within a specific tolerance or within a certain time limit. This idea is demonstrated in the results in Section 3.

2 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.

2.1 Abstract Sensor Model

We can view the sensory system using three different levels of abstractions.

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.

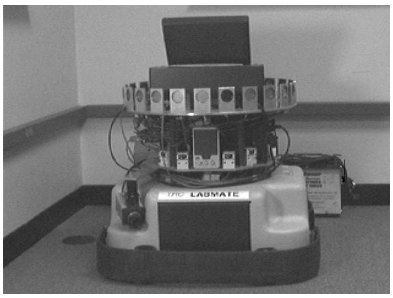

Figure 1: The LABMATE robot with its equipments.

2.2 A Distributed Control Architecture

Several sensors can be grouped together representing a logical sensor [2, 4]. 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.

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.)

3 Experiment 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 [5]. 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 [3]. For that purpose, the LABMATE was equipped with 24 sonar sensors, eight infrared sensors, a camera and a speaker. [1] Figure 1 shows the LABMATE with its equipment.

3.1 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 one of these experiments.

[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.

In this experiment, We demonstrate the use of the tolerance measures. This experiment also illustrates the use of the logical sensors concept to implement high-level requests which incorporate tolerance measures and time calculations.

The request which was implemented for this experiment is *measure* which has the following syntax:

$$measure(tolerance, time, preference)$$

where $tolerance$ is the required tolerance with 0 meaning get the best tolerance, and -1 means tolerance is not important. $time$ is the required response time, and again 0 means as fast as possible, and -1 means time is not important. When both, $time$ and $tolerance$ are specified, the logical sensor may not be able to satisfy both criteria, and this is when $preference$ is used to specify which criteria should be preferred. This request returns the resulting tolerance and the time consumed into the same parameters that were sent.

The following is the output of a program which uses this request to measure a point in front of the robot. First it sends a request to get the measure as fast as possible ignoring the tolerance.

```
Fast response required ...
!!! minimum time ...
!!!  current reading is 2071 mm, with tolerance 402.4 in time 0.9 sec.

Result: distance = 2071 mm, tolerance = 402.4, and time = 0.9 sec.
```

Second, the program sends a request to get the best accuracy (minimum tolerance), and the time is irrelevant.

```
Best tolerance required ...
!!! minimize the tolerance ...
!!!  current reading is 1536 mm, with tolerance 298.4 mm. in time 0.6 msec.
!!!  current reading is 1412 mm, with tolerance 274.3 mm. in time 2.5 msec.
!!!  current reading is 1383 mm, with tolerance 268.7 mm. in time 3.4 msec.
!!!  current reading is 1350 mm, with tolerance 262.3 mm. in time 4.4 msec.
!!!  current reading is 1291 mm, with tolerance 250.8 mm. in time 5.6 msec.
!!!  current reading is 1259 mm, with tolerance 244.6 mm. in time 6.5 msec.
!!!  current reading is 1215 mm, with tolerance 236.0 mm. in time 7.6 msec.
!!!  current reading is 1168 mm, with tolerance 226.9 mm. in time 8.7 msec.
!!!  current reading is 1139 mm, with tolerance 221.3 mm. in time 9.6 msec.
!!!  current reading is 1091 mm, with tolerance 212.0 mm. in time 10.4 msec.
!!!  current reading is 1062 mm, with tolerance 206.3 mm. in time 11.0 msec.
!!!  current reading is 1015 mm, with tolerance 197.2 mm. in time 11.8 msec.
!!!  current reading is 971 mm, with tolerance 188.6 mm. in time 12.5 msec.
!!!  current reading is 938 mm, with tolerance 182.2 mm. in time 13.2 msec.
!!!  current reading is 894 mm, with tolerance 173.7 mm. in time 13.9 msec.
!!!  current reading is 847 mm, with tolerance 164.6 mm. in time 14.7 msec.
!!!  current reading is 818 mm, with tolerance 158.9 mm. in time 15.3 msec.
!!!  current reading is 756 mm, with tolerance 146.9 mm. in time 16.3 msec.
!!!  current reading is 724 mm, with tolerance 140.7 mm. in time 16.9 msec.
!!!  current reading is 694 mm, with tolerance 134.8 mm. in time 17.5 msec.
!!!  current reading is 633 mm, with tolerance 123.0 mm. in time 18.4 msec.
!!!  current reading is 603 mm, with tolerance 117.2 mm. in time 19.0 msec.

  distance = 1536 mm,  tolerance = 117.2, and time = 19.0 msec.
```

Finally, the program specifies both time and tolerance to be met, preferring the time.

```
Tolerance required = 150.0, time required = 6.0 msec.
!!! both criteria are specified ...
!!!  current reading is 2190 mm, with tolerance 425.5 mm. and time 0.9 msec.
!!!  current reading is 2101 mm, with tolerance 408.2 mm. and time 2.7 msec.
!!!  current reading is 2068 mm, with tolerance 401.8 mm. and time 4.1 msec.
```

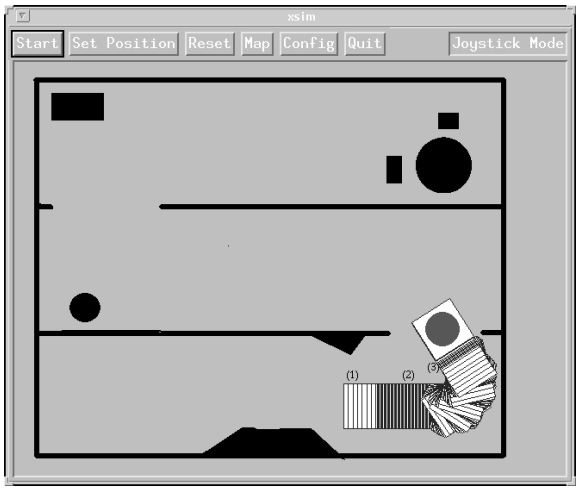

Figure 2: The trajectory of the robot while performing the requests.

```
!!!  current reading is 2026 mm, with tolerance 393.6 mm. and time 5.6 msec.
Result: distance = 2190 mm, tolerance = 393.6, and time = 5.6 msec.
```

Figure 2 shows the movement of the robot while taking these measurements. The first request did not cause any movement since it required minimum time. The second request cased the robot to move forward to minimize the tolerance region. During this movement, the speed of the robot decreases to get better accuracy. Finally, the last request also caused the robot to move forward, but it stopped before reaching the required tolerance since the time was preferred.

In this experiment we used only the translation in the y direction to minimize the tolerance.

4 Conclusions

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. Tolerance measures for sonar sensors were proposed and different strategies to increase position accuracy were investigated. An example for applying this control scheme to a mobile robot was described along with the simulation results. We believe that this control scheme provides 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.

References

[1] 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.

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

[3] 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.

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

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