(Accepted by the 1994 International Conference on Robotics and Automation)

Why Is It So Difficult For A Robot To Pass Through A Doorway Using UltraSonic Sensors?*

John Budenske and Maria Gini

Department of Computer Science
University of Minnesota
Minneapolis, MN 55455

Abstract

Complex tasks are usually described as high-level goals, leaving out the details on how to achieve them. However, to control a robot, details must be provided. Having the robot move itself to and through an unknown, and possibly narrow, doorway is an example of such a task. We illustrate the difficulty of such a task using actual data from a real robot. We show how the transformation from high-level goals to primitive commands can be performed at execution time and we propose an architecture based on reconfigurable objects that contain domain knowledge and knowledge about the sensors and actuators available. We then show how our approach is used in solving the illustrated task.

1 Introduction

A large part of our civilized world is unstructured and dynamic. This fact has been one of the leading obstacles to mass-quantity introduction of robots into everyday life. The variability of the world makes it impractical to develop detailed plans of actions before execution. The world might change and thus invalidate the plan. It makes more sense to develop, before execution, a detail-less plan of what to do and make the necessary details explicit during execution.

Thus, we are faced with a problem: how can a robot achieve a high-level goal and still remain reactive to its environment? Many approaches have been proposed to make robots reactive [3], to combine planning with reactivity [2, 6, 10], and to plan for sensors use [1, 9]. In the approach we present here the selection of the sensors, actuators, and processing algorithms is made at execution time. The flow of data from sensors into actuators is continuously reconfigured, depending on information sensed from the environment. Reactivity is maintained through constantly managing the sensors, collecting data, and using the data to further guide the robot through execution [5].

One example of a difficult robotic task which we have encountered in our endeavors is that of having the robot navigate itself to and through an unknown, and possibly narrow, doorway. Most people who have never attempted to program a robot to progress through an arbitrary doorway do not believe there is a problem. Small robots and large doorways are an obvious way to simplify the problem, but the problem becomes extremely difficult if the clearance is small, if the position of the doorway is not perfectly known, or if there are unexpected obstacles in the way. Obviously, going through a doorway is just an example of a task that we can solve with our approach. We will use it through the paper to help us in describing the issues and our solution.

2 Observed Problems

The problem of passing through a doorway can be decomposed into the sub-tasks of finding the doorway, tracking it during approach, and passing through it.

Finding the doorway is more than just looking for a "sonic hole" in the sensor stream. The pattern of a short distance, long distance, short distance (or variation thereof) can be attributed to many natural indoor structures as well as to the general occurrence of noise. The real problem comes when the robot finds something that looks like a doorway. How can the robot verify that it is indeed a doorway? To illustrate this point, let us first play a game of "Find the Doorway" (please don't look at the figures yet!).

Figures 1, 3, and 5 show the result of collecting one

*This work was funded in part by the NSF under grants NSF/CCR-8715220 and NSF/CDA-9022509, and by the AT&T Foundation.

snapshot of ultra-sonic readings from a ring of 24 sensors. The data are collected as an array of 24 readings *(x, y, θ, distance)*, where x and y are the position of the center of the sensor ring on the robot, θ is the angle from the front of the robot to the direction of the sensor, and *distance* is the returned range calculated from the time-of-flight to the object the sonar beam is bouncing off. Sonar data are illustrated by drawing the robot (the black square, where the notch indicates the robot's front), and plotting for each sensor an arc at the corresponding angle and range. An arc is used since any object in the sensor's cone (30 degrees) can return the sound signal[1]. To further illustrate the data, a line is drawn from the midpoint of each arc to the next. The resulting outline is a visualization of the "open" space around the robot.

Now, examine this visualization and "guess" the location of the doorway. Once you have decided, look at Figures 2, 4, and 6, which have a layout of the room superimposed on the original visualization. The game for Figure 1 is easy, but Figures 3 and 5 show how the doorway can be missed. Unfortunately, the sensed data are not always as clean as in Figure 1, where only intermittent noise occurred. The last four figures show the effect of physical interaction between the sensor beam and surrounding structures. This "noise" (that we call a *ghost door*) was consistently present until the robot moved, changing its physical relationship with the environment. Of course moving the robot increases the positional error.

Detecting the doorway is just the beginning. When tracking the doorway, it is easy to lose it and find it again later. If this occurs, is the robot sensing a ghost door or a real one? Even if the robot can sense exactly where the doorway is, moving to and through can cause other errors, since often the precision in positioning decreases with the distance traveled. Other problems occur when the robot goes through the doorway. Suppose the bumpers make contact with something. Is this conclusive evidence that there is no doorway? Or does it only mean that there is something between the robot and the doorway[2]? What does it mean to have the bumper contact something when the robot should be passing through the doorway? Is the robot misaligned to the opening? Is it farther through the opening than previously thought? Is the doorway too small for passage? Is the door in the way? Or is there an object obstructing the pathway?

[1] Actually, the sensor beam's footprint is far more complex than a cone, but for this illustration, the simplicity of a cone allows for better understanding.

[2] One would be surprised what effects a harmless ruffle in the carpeting could have.

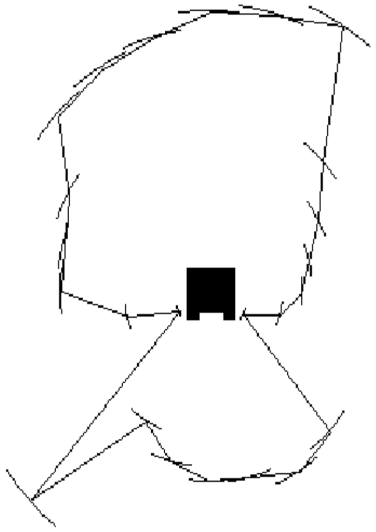

Figure 1: Sonar data

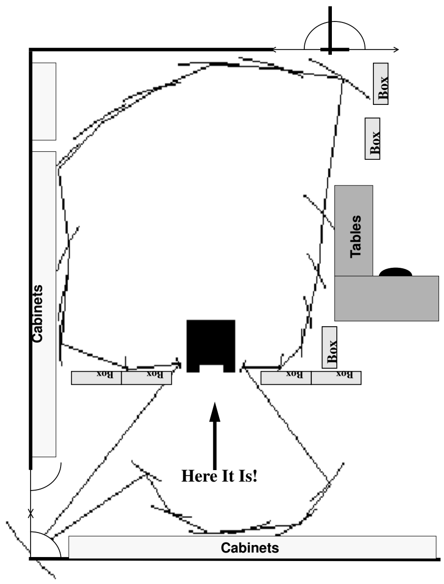

Figure 2: Find the Doorway

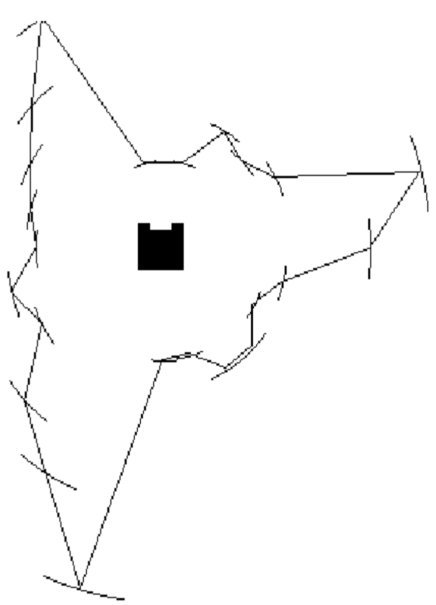

Figure 3: Sonar data

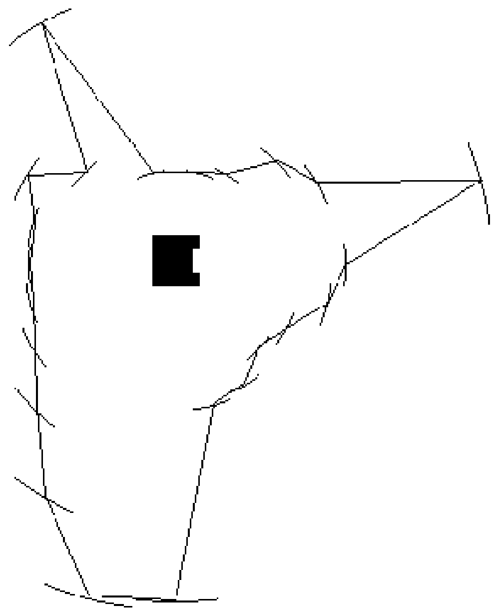

Figure 5: Sonar data

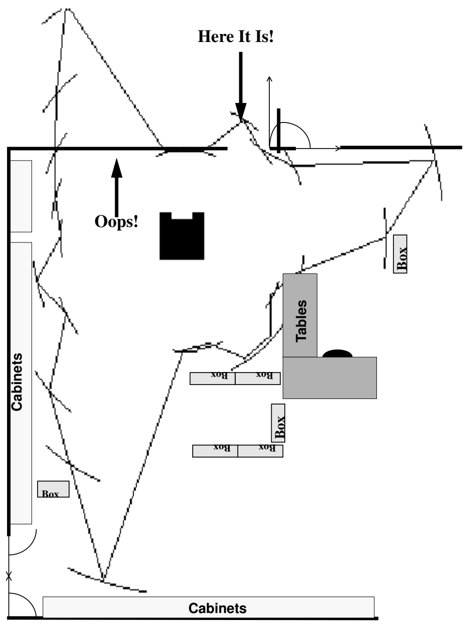

Figure 4: Find the Doorway

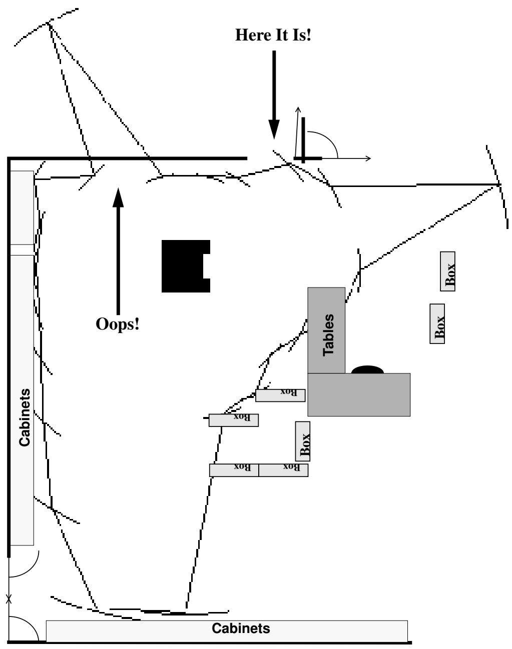

Figure 6: Find the Doorway

Overall, there are a number of things that could happen or go wrong as the robot searches for, tracks, and maneuvers through the doorway. Each situation calls for a different response, and the real problem is not so much what the response strategy should be, but what the situation is. Often extensive interaction with the environment is necessary just to collect information for assessing the situation.

It is also important to realize that expecting the execution to always go right is futile. Errors will occur, and unexpected events will get in the way of success. Gat [7] uses the concept of "cognizant failure", or failures which can be monitored for, detected, and thus recovered from. Monitoring for errors is actually monitoring for deviations from the normal expected processing. Thus, design for cognizant failure must lead to design for "Anticipatory Deviation". Knowledge on what and when to monitor, and on how to resolve is key to our approach.

Possibly, the above account of the problem may seem to trivialize it. That is not our intent. Our experiences have led us to the conclusion that as we configure the robot to tackle each task and subtask, the situations encountered cannot simply be addressed by a single algorithm. Multiple situations, multiple algorithms, multiple responses, and multiple sensing appears to be the course of action.

3 Logical Sensors and Actuators

As we mentioned before, the problem of going through a doorway is only representative of the sensor intensive tasks we are investigating. The central problem is in transforming the given high-level goal into an explicit representation of what is necessary to achieve it. We call this transformation process *Explication* and have discussed its implementation in [4].

At the center of Explication is domain knowledge. This includes knowledge on the use of sensors and actuators in different environment situations; knowledge on the characteristics (e.g. noise) of the sensor data and actuator commands involved; and knowledge on what deviations to anticipate and monitor for.

Two concepts are crucial. First is the notion of *Relevant Information Need.* Explication must deal with the question of "what information is relevant and thus is needed to execute this goal?". The goal "achieve X" is replaced by two things: explicit knowledge on how to achieve X using the sensors and actuators available, and an explicit representation of the needed information. Informational needs are then met by information providers. We have utilized the concept of Logical Sensor [8] to be such an information provider, and we have expanded it to allow other logical entities, that we call collectively *Logical Sensors/Actuators (or LSA).* Equally important is the concept of *Utilization-Detail.* Explication must deal with the question of "what details of using sensors and actuators are necessary in order to execute this goal?"

When given a goal, Explication uses the domain knowledge to find the relevant information and the details needed. These needs are then met by LSAs which either map directly to the corresponding information/details or apply Explication hierarchically until all needs are satisfied.

This framework has been implemented in an object oriented programming environment. It results in a flexible robotic sensor/control system which we call the *Logical Sensor Actuator Testbed (LSAT).* The LSAT is hosted on a Sun SPARC 4/330 computer which interacts with a TRC Labmate mobile robot (nicknamed Eric the Red) over two separate serial lines. The system is written in C, Lucid Common Lisp with the Common Lisp Object System (CLOS) and LispView windowing system. The framework is, basically, a collection of reconfigurable objects which are automatically combined. The objects contain methods for describing sensors, actuators, and processes. The knowledge is represented as rule-like entities within each LSA object. Knowledge is used within each LSA object to coordinate with the other LSAs necessary to achieve the goal.

We have experimented with doorways of different size and type, with different placement of the doorway (opening in the middle of a room or into a corner, etc), and with various amounts of obstacles in the room and near the door. An initial version of knowledge has been built into the current system. This includes knowledge to perform the standard maneuvers for seeking, tracking, verifying, and passing through a standard doorway, as well as identifying when the doorway is too narrow for passage. The knowledge spans across over 50 different LSA objects during the course of execution. Preliminary experiment runs show a success rate of 78 percent (39 success out of 50 attempts) at doorway situations varying in size, location, doorway type, and obstacles. We observed that as more knowledge was acquired, system robustness improved. Current efforts are concentrating on building more knowledge to deal with additional situations.

4 Example of Explication

To illustrate our approach, we will use an example from our lab experiments. The goal given to the robot is "MoveThrough (DOOR1)", where DOOR1 is the designated door to move through. The LSA *MoveThrough*, is matched to the goal. The overall strategy of *MoveThrough* is to coordinate between three phases:

1. moving and searching/detecting the doorway;
2. verifying the detected doorway and obtaining more precisely its location;
3. moving through the doorway.

For brevity, we will assume the robot has completed the first phase and it is near what looks like a doorway. It is important to note that the switch from the first phase to the second is coordinated by the *MoveThrough* LSA and that many of the LSAs used in the first phase are deactivated while others are reconfigured with newly activated LSAs to achieve the next phase. For example, a LSA called *DetectDoor* is used during the first phase to identify potential doorways. It is retained for the second phase to continue monitoring that doorway. If the doorway ever disappears (i.e. its tracking is lost), *MoveThrough* will shift again back into the first phase and attempt to re-attain tracking[3].

In the second phase, the *VerifyDoor* LSA is activated to make sure that the doorway truly is a doorway, that it is open, and that the robot could fit through it. It initially sets up a *DoorSize* LSA to monitor the size calculations from other LSAs and to determine how much clearance there is for the robot to pass through. If there is a large clearance, the robot would engage simple, yet imprecise, LSAs for propelling itself through the doorway. If it is a close fit, additional investigation must occur to verify the size and to better align the robot to the opening. The process of verifying the doorway involves *VerifyDoor* coordinating a number of LSAs through three sub-phases:

1. moving the robot next to the doorway, within the range of the proximity sensors
2. move the robot back and forth in front of the opening to verify its existence and size
3. center the robot in front of the doorway.

Once the door frame is verified (assuming the passage is a close fit), the *VerifyDoor* LSA is deactivated (thus deactivating all of its sub-LSAs) and two new LSAs are activated. The first is the *MonitorDoor* LSA which uses the sonar sensors to monitor the robot's placement and the robot's movement through the doorway. It also detects any new obstacles which may appear on the other side of the passage. This serves as a check on the progress of the second new LSA, *CloseQuartersMove*. Upon activation, the *CloseQuartersMove* LSA activates and coordinates a number of parallel sub-LSAs, which accomplish these sub-goals:

1. propel the robot through the doorway to a predicted "other side";
2. keep the robot in line with the door opening as the robot moves through the doorway, and monitor for obstacles in the forward path;
3. monitor the doorway-frame as it passes each of the side-mounted proximity sensors;
4. resolve various types of mis-alignments of the robot to the doorway-frame via shifting left/right;
5. monitor the movement of the robot, differentiating between pauses (used during the shifting and propelling) and low velocity stagnation[4].

Once the propelling and/or the *MonitorDoor* LSAs determine that the robot has reached the other side of the passage, *CloseQuartersMove* declares success, and thus *MoveThrough* declares success. If an error is encountered, additional strategies are applied to determine what is wrong and what to do about it.

5 Variations, Knowledge, Strategies

The above example is merely a template for what could happen and shows the anticipated sequence of actions for a typical execution. In real life, each attempted execution could yield a different sequence of actions due to the variations of the environment. These variations can come in many forms: sensor noise, actuator imprecision, imprecise knowledge of object locations and other world relationships. Overall, there are many little variations which can impact the situations the robot is in and the strategies selected for achieving the goal. Examples are: the doorway opening being of an unexpected size; the placement of the door within the opening making detection and passage difficult; whether the door opens in or out; if it is located in a corner of the room as opposed to the middle of a wall; ghost doors caused by sensor

[3] Of course knowledge is also present in MoveThrough to identify when a multiply-disappearing doorway is really not a doorway.

[4] Often during low velocities, the wheel motor drives will freeze up and require a reset of the velocity register to resolve.

noise and peculiarities in the data collection process; and unanticipated obstacles in the path of the robot.

There are a number of strategies which can be taken to reduce sensor noise. One strategy we use is to filter out the noise over a sequence of readings. If the robot is not moving, accuracy to within a few millimeters can be obtained with just a few readings. Filtering could be done via disregarding obvious outliers, averaging, or applying more sophisticated methods. Filtering works well when most of the values are close to the true value. If the robot happens to be sitting in a dead spot (i.e. a point where the sonar echo does not return directly back at the sensor's receiver and is thus undetectable), or an echo point (i.e. a point where multiple returns occur due to echoing) most to all of the readings are greatly off and thus the filtered value is unusable. Another problem is caused by moving objects passing near the robot that might produce a temporary disturbance in the sensed distances.

Knowing that these problems can exist is encoded as domain knowledge within the LSAs. If the robot detects (or simply suspects) that it is in a dead spot or echo point, it can apply the strategy of moving around from collection point to collection point and from there determine the best location for collection. Enough collection points need to be visited to guarantee that good points can be differentiated from bad points. Of course, the problem will still arise when half of the collection points all yield approximately the same filtered value, and the other half all yield a totally different filtered value. When that situation is detected, a different strategy needs to be applied.

Another strategy is to continuously move the robot around and incrementally fuse the sonar data across the movement. Though less precise than finding a single, echo-less collection point, this has a higher probability of producing "close-to" correct data. This is fine for tasks such as navigating across an empty room, but in tasks where there is little clearance, combinations of the above strategies may be needed.

The domain knowledge that helps in deciding the best strategy is encoded in multiple LSAs, and Explication uses that knowledge to coordinate the achievement of each sub-goal, and thus, the overall goal. These variations and strategies make this domain excellent for applying the LSA theory and its implemented LSAT.

6 Conclusion

We have described an architecture for accomplishing goals that uses large amounts of domain specific knowledge. This knowledge is used to decide, at execution time, what sensors, actuators, and processes are necessary for achieving the assigned goal. The mechanisms for reconfiguring the flow of data from sensors into actuators make this architecture appropriate for solving complex tasks in unstructured environments, as evidenced by the sample task shown in the paper.

Acknowledgements

We would like to thank John Fischer, Chris Smith, Karen Sutherland, and Eric the Red for their help.

References

[1] S. Abrams, P. K. Allen, and K. A. Tarabanis. Dynamic sensor planning. In *Proc. Intelligent Autonomous Systems*, pages 206–215, 1993.

[2] R. C. Arkin. Motor schema-based robot navigation. *International Journal of Robotics Research*, 8(4):92–112, Aug. 1989.

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

[4] J. Budenske. *Robotic Plan Execution in Dynamic and Unpredictable Environments.* PhD thesis, University of Minnesota, 1993.

[5] J. Budenske and M. Gini. Achieving goals through interaction with sensors and actuators. In *Proc. IEEE/RSJ Intern. Conf. on Intelligent Robotics and Systems*, pages 903–908, 1992.

[6] R. J. Firby. Building symbolic primitives with continuous control routines. In *Proc. Artificial Intelligence Planning Systems*, pages 62–69, 1992.

[7] E. Gat. *Reliable Goal-Directed Reactive Control of Autonomous Mobile Robots.* PhD thesis, Virginia Polytech. Inst. and State University, 1991.

[8] T. Henderson and E. Shilcrat. Logical sensor systems. *Journal of Robotics*, 1(2):169–193, 1984.

[9] S. A. Hutchinson and A. Kak. Planning sensing strategies in a robot work cell with multi-sensor capabilities. *IEEE Trans on Robotics and Automation*, RA–5(6):765–783, 1989.

[10] D. W. Payton, J. K. Rosenblatt, and D. M. Keirsey. Plan guided reaction. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(6):1370–1382, 1990.