Natural Communication with Robots

by

Mark C. Torrance

B.S., Symbolic Systems
Stanford University (1991)

Submitted to the Department of Electrical Engineering
and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1994

Signature of Author . . . .
Department of Electrical Engineering
and Computer Science
January 28, 1994

Certified by . . . .
Lynn Andrea Stein
Class of 1957 Assistant Professor of Computer Science
Thesis Supervisor

Accepted by . . . .
Frederic R. Morgenthaler
Chairman, Departmental Committee on Graduate Students

Natural Communication with Robots

by

Mark C. Torrance

Submitted to the Department of Electrical Engineering
and Computer Science
on January 28, 1994, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

We have developed a system which mediates between an unmodified reactive mobile robot architecture and human natural language I/O. We introduce reactive-odometric plans and demonstrate their use in plan execution, plan recognition, and learning to associate human terms with perceptually unremarkable locations in the environment. The communication component of our architecture supports typewritten natural language discourse with people. It lets users name places either immediately or in relation to other known places, ask questions about the robot's plans and the spatial relationships of known places, and give the robot short and long term goals. This thesis presents results obtained with our implementation of this architecture on a physical mobile robot system designed by[Connell, 1992a] and in simulation. These results reflect experiments performed by the author and by other users.

Thesis Supervisor: Lynn Andrea Stein
Title: Class of 1957 Assistant Professor of Computer Science

About the Author

Mark Torrance received the degree of Bachelor of Science with distinction in Symbolic Systems at Stanford University in March, 1991 before coming to M.I.T. He is a member of Phi Beta Kappa, a recipient of the Stanford Dean's Award for Academic Achievement, and was awarded a National Science Foundation Graduate Fellowship. He plans to continue working at M.I.T. in the Artificial Intelligence Laboratory toward a Ph.D. in Electrical Engineering and Computer Science.

Acknowledgements

This research would not have been possible without the wonderful guidance and assistance of a host of people. My adviser, Lynn Stein, deserves a great deal of credit for her enthusiasm, her confidence in me, and her terrific advice every step of the way. Ian Horswill, having just completed his dissertation in a related area of mobile robotics, provided immeasurable support and assistance on matters ranging from hardware implementation to content and style of this document. I offer my great thanks to IBM T. J. Watson Research Center and Jonathan Connell for loaning MIT the mobile robot used to implement and demonstrate this thesis work. Michael Frank was generous with his time in serving as a subject for both of my experiments involving other people, and in valuable discussions about this research. George John at Stanford copied and carried the AAAI version of this thesis the important last leg of its journey. Maja Mataric and Barney Pell were truly helpful and inspirational as I worked toward this goal.

This thesis is based upon work supported under a National Science Foundation Graduate Fellowship. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. The research described here was conducted at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology under the supervision of Professor Lynn Andrea Stein. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014–91–J–4038. Professor Stein's research is also supported by the General Electric Foundation Faculty for the Future Award and by the Class of 1957 Career Development Chair.

This thesis is dedicated to Leslie, my fabulously devoted and supportive wife, without whom I am quite sure I could not have accomplished this at all.

To Leslie

Contents

List of Figures

Chapter 1

Overview

Two important capabilities for a mobile robot are the ability to know and report where it is and the ability to go where people tell it to go. Several current mobile robots have these abilities in more or less limited forms. Unfortunately, the language in which people ask questions, receive answers, or give instructions is often unnatural, if these capabilities are supported at all. Frequently this makes the process of interacting with a mobile robot difficult or complicated.

One problem with making robots that can understand human descriptions for places is that to actually use these descriptions at face value requires arbitrarily sophisticated perception. For example, consider these terms people might use in descriptions of places: "A map of Russia on the wall," "a big room," and "office number 705." These are perfectly reasonable and natural ways to refer to places, but they potentially require very complicated perception.

A solution to this problem takes advantage of the fact that in most cases, the descriptions people use in giving directions or commands refer to stable parts of the environment. The poster isn't going anywhere any time soon, nor is the office. Thus, if we can make a robot associate human-language descriptions with places it recognizes through its own perceptual mechanisms and navigation strategy, our robot will be able both to take advantage of successful low-level navigation techniques and to interact with people in a natural way.

We would like people to be able to direct a robot to go to any location within a building that they can name and that it can physically reach. We would like to support descriptions which are convenient for people, because we can remember them and others can also use them.

Therefore, we have developed a system which mediates between an unmodified reactive mobile robot architecture and human natural language I/O. We introduce reactive-odometric plans and demonstrate their use in plan execution, plan recognition, and learning to associate human terms with perceptually unremarkable locations in the environment. The communication component of our architecture supports typewritten natural language discourse with people. It lets users name places either immediately or in relation to other known places, ask questions about the robot's plans and the spatial relationships of known places, and give the robot short and long term goals. This thesis presents results obtained with our implementation of this architecture on a physical mobile robot system designed by [Connell, 1992b] and in simulation. These results reflect experiments performed by the author and by other users.

This chapter begins by defining a task for mobile robots that requires them to communicate with a human tutor to learn human names for places in their shared environment, and explaining the motivation for this task. The chapter continues by stating the project of this thesis and describing evaluation criteria that apply to any solution to the described task. A brief sketch of our solution, a description of the contributions of this thesis, and a roadmap to the remainder of the thesis complete this chapter.

1.1 The Task

We define a task for mobile robots called the *communicating mobile robot task.* The task presumes a navigating mobile robot that has basic corridor following and obstacle avoidance capability. We address in Section 4.4 the specific requirements we make of this robot navigation system, and the suitability of some particular existing navigation systems and strategies to this task.

The communicating mobile robot task is defined by the following additional requirements of the robot:

- The robot should learn to associate human terms with places in its environment by receiving a running description of its immediate surroundings as it explores.

- The robot should learn about additional places from descriptions of them which relate them spatially to places it already knows.

- The robot should respond appropriately to instructions given in these learned terms, which includes navigating to previously named places.

- The robot should use these terms correctly to describe places it has visited and the spatial relationships among places it has learned about.

The remainder of this section describes each of these requirements in more detail. The motivation for this choice of task is described below in Section 1.2.

The robot should learn human terms for places in its environment by receiving a running description of its immediate surroundings as it explores. In our present formulation of the task, places must be points in the environment, such as "in front of Mark's office door", rather than areas with extent, such as whole corridors. In Chapter 10, which describes future work, we discuss the possibility of relaxing this requirement. While we do not make the use of natural language explicit in the task description, after considering a variety of communication strategies in Chapter 2 we decide to use natural language for our solution to this task. These descriptions are provided in typewritten English by a tutor who follows the robot around as it explores its environment.

The robot should learn about additional places from descriptions of them which relate them spatially to places it already knows. One of the uses for a robot which could solve this task would be to direct visitors to places they want to reach. To facilitate this, the robot should be able to learn about places that it can't actually reach, if they are important to people.

The robot should respond appropriately to instructions given in these learned terms. The robot is expected to correctly navigate to a place it has learned about when it is instructed in the appropriate communication modality to go to that place.

The robot should use these terms correctly to describe places it has visited and the spatial relationships among places it has learned about. In support of the ability to communicate about places, we require that the robot be able to correctly answer questions such as “What is to the north of Mark’s office?” and “How would I get from the AP lab to Andy’s office?”.

1.2 Motivation For This Task

The communicating mobile robot task is motivated by a number of goals. We seek to build robots that are easy to instruct. We want robots to be able to learn about environments other than the ones in which they were first tested. An interactive conversation with a tutor can help a robot solve localization tasks and deal with perceptual aliasing problems. Robots would need human tutors even if they could locally recognize any place, in order to develop a common language. Communication facilitates judgments of the rationality of the robot. This section addresses these motivations in turn.

Current navigating mobile robots suffer from the problem that the language in which you give them goals is unnatural. Many robots can only accept goals that are programmed by their designer in some internal robot representation. It is an important goal of this research that people be able to instruct robots in ways which are convenient for the people instead of convenient for the robots. This motivation supports the more widespread use of robots by people who can learn the restrictions on the communication language more readily than they can learn the details of an internal spatial representation.

Robot systems which require the robot programmer to train the robot to recognize landmarks don’t work well when the robot needs to work in a different environment. The robot designer, or other expert, must travel with the robot to the new environment to reprogram the robot's landmark recognition. A concrete goal of the research described here is to build a robot system that could be trained by someone who is not familiar with the internal structure of the robot program, in an environment other than the one in which the robot was originally designed.

Localization is a mobile robot task in which the robot must determine its location after being turned on at an unknown position within a known environment. Some authors [Basye, 1992a, Brown *et al.*, 1992] solve this localization problem by exploring the environment after seeing the perceptually ambiguous place until the actual place can be discriminated. While we do not address the localization problem directly, we believe that information provided in natural language by a tutor can in some cases obviate the need to explore for localization.

The perceptual aliasing problem occurs when the immediate local perceptual image of two different places appears the same [Chrisman, 1992, Whitehead and Ballard, 1991]. One of the things that makes the communicating mobile robot task hard is that there may be perceptual aliasing between places that the tutor wants the robot to discriminate. We expect that information provided by a tutor can help the robot to resolve problems of perceptual aliasing. In particular, in cases where the robot has become confused about its location, the tutor can offer clarifying advice that may disambiguate the robot's location.

Even if perceptual aliasing were not a problem, the ability to communicate with a tutor is important for a robot that will be able to accept goals expressed in human terms. The landmarks in the environment that are convenient for robot navigation don't necessarily correspond to places that are important to people for communication about the world. If we wish to give our robots instructions in ways that are convenient for us, we need to develop robots that have the ability to learn which places in the environment are important in our ontology. We believe that this can be done without resorting to powerful or unsolved techniques in perception to pick out the objects and features that people actually use, by taking advantage of a certain structure in the environment. In particular, the names people use for places pick out *places*, which the robot can come to recognize by other means so long as they are stationary.

People judge the rationality of an agent's behavior in part based on their understanding of its intentions. The intentions of a robot which cannot communicate effectively with people are far less apparent than those of a robot that can. By building a robot that can communicate with people, we expose its intentions to its observers and thus enable more sophisticated judgments of its rationality. We do not argue that communicating robots are necessarily more rational than noncommunicating robots, only that they engage in more activity on which assessments of their rationality can be based.

The communicating mobile robot task is motivated by each of these desiderata. The project of this thesis has been to develop a mechanism which can solve the task.

1.3 The Project of this Thesis

This thesis explores the communicating mobile robot task by considering possible solutions to the task, the extent to which existing work can be used to make progress on this task, and by proposing a solution and demonstrating it on a physical robot. We have not attempted to come up with the only mechanism that can perform the task. Instead, our contribution is in the representations for spatial plans we have developed, and our analysis of the way these representations mediate between communication and reactive navigation to solve the task.

1.4 Our Solution

This section describes our solution in brief. Our system is composed of three main components: natural language I/O, a memory and planning system, and a reactive robot capable of simple corridor following and obstacle avoidance. Our natural language system uses very simple techniques and the reactive robot system was designed and implemented by [Connell, 1992b]. The main contribution of this

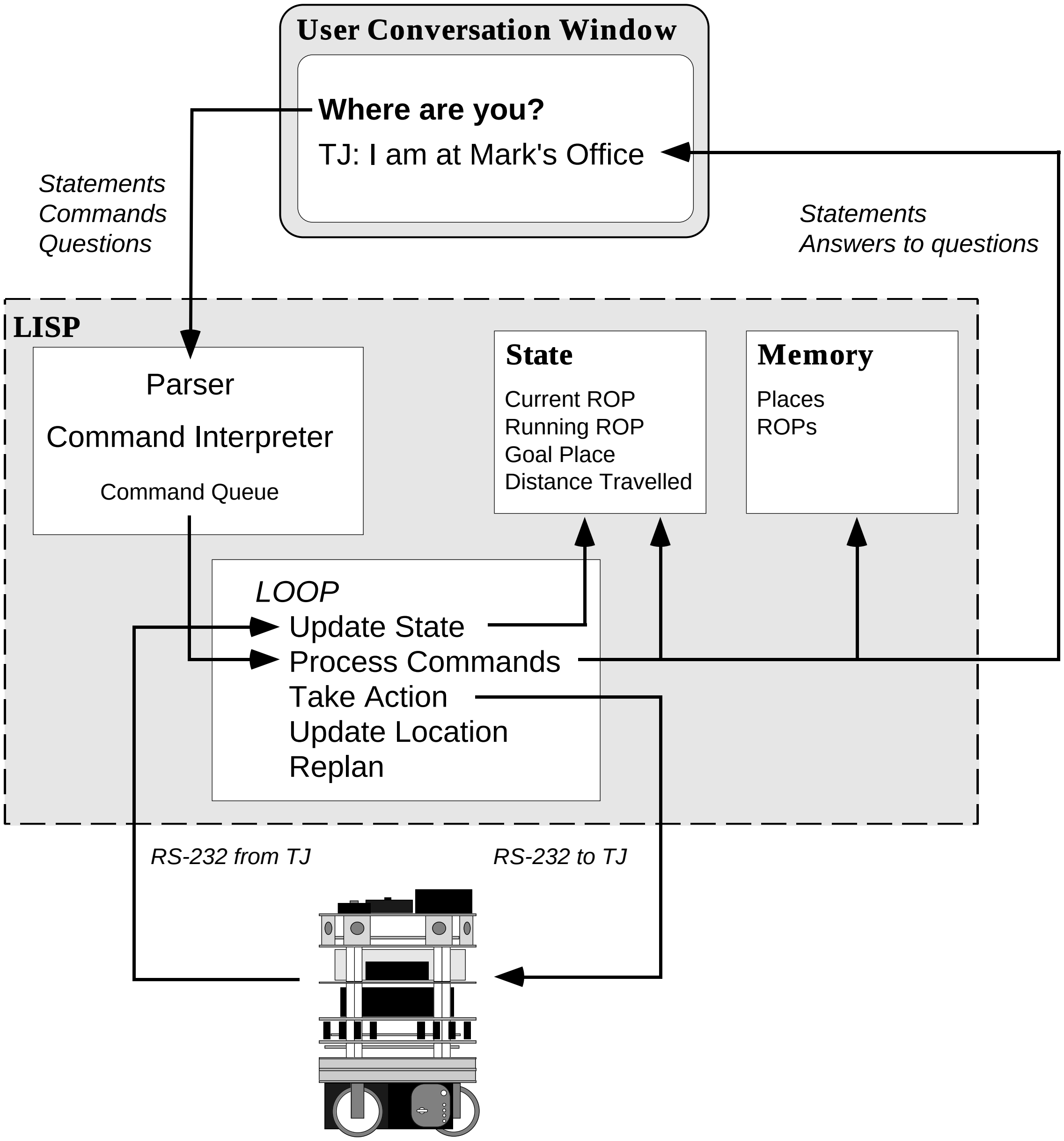


Figure 1-1: Overview of our system. This figure shows some of the information flow within the system we describe.

work lies in the design of the plan learning and execution system which mediates between these other components. Figure 1-1 shows the general structure of this system.

Our architecture mediates between the user and the robot by way of the state and memory stored in a Lisp system. Navigation and communication are supported by data structures for *places*, *plans*, and *reactive-odometric plans*, or ROPs, all of which are described in Chapter 5.

The robot system navigates by storing a directed graph that connect *places* by way of ROPs. ROPs are *automatically recorded* each time the robot is told by a user that it is at a new named place. These ROPs serve as the arcs in a simple graph which is searched by shortest-path algorithm that enables the robot to navigate again to a place it has learned about. Place recognition is performed by *plan recognition* on the relevant ROPs. We treat the underlying reactive robot system as a black box, and operate only on its control interface as described in Chapter 4. The details of the implementation of our higher level navigation are described in Chapter 6.

Natural language understanding is accomplished by parsing statements, commands and questions into simple Lisp function calls, which operate on the state and memory. Most language generation is performed by these same functions, which generate language in response to utterances made by the user. Language may also be produced to report state changes the system recognizes such as arrival at a known place. The implementation of the natural language component of our system is described in Chapter 7.

1.5 Sample Dialogue

By way of illustration, we here include a sample dialogue with our system. This is not taken from an actual interactive session with the robot, but was handcrafted for pedagogical reasons. All of the behavior exhibited in the dialogue is implemented and actually works on the robot.

We use a simple hypothetical map for this example, shown in Figure 1-2. Pat's

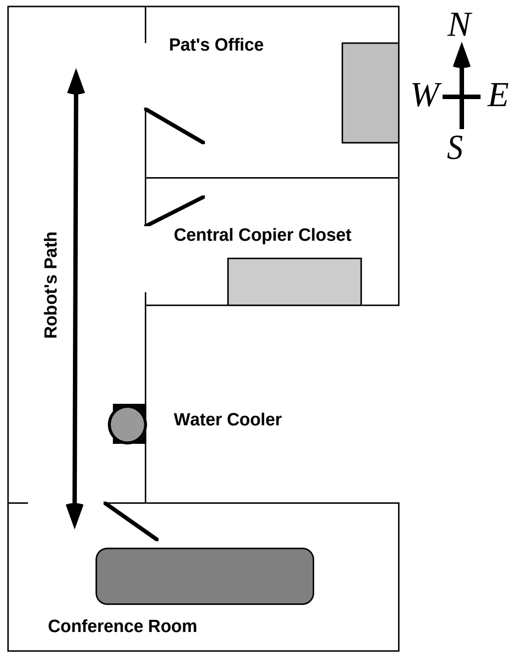


Figure 1-2: Map of hypothetical environment for sample dialogue

office is at the top, or north. The Central Copier Closet is south of it, the water cooler is further south, and the conference room is at the bottom. At the start, the robot is facing west and is in the hallway in front of Pat's office.

"You are at Pat's office" [1]

"I am learning about PAT'S OFFICE"

"Turn Left"

"I am turning left."

The robot turns left and faces south.

"Where are you?"

"I am AT PAT'S OFFICE."

The robot remembers the preposition *at* used by the tutor. It knows it is still at the same location since it has only rotated in place.

"Go to the end of the hallway."

"I am going to go until I see an obstacle in front."

[1] Boldface statements in quotations are text which the user types to the robot.

The robot drives to the end of the hallway.[2]

"This is the conference room."

"I am learning about THE CONFERENCE ROOM."

The robot stores a ROP to get from Pat's office to the conference room, and a reverse ROP to get back.

"The water cooler is to the north of the conference room."

"I am learning about THE WATER COOLER."

The robot doesn't yet know how to get there, but it has a limited sense of where the water cooler is.

"Go to Pat's office."

"Now I am going to PAT'S OFFICE."

"Would you please tell me if we pass THE WATER COOLER?"

The robot plans a route to Pat's office, and executes the ROPs that lie along this route. The second sentence expresses the robot's limited knowledge about the location of the water cooler.

"Where are you?"

"I'm on my way from THE CONFERENCE ROOM to PAT'S OFFICE."

"You are at the central copier closet."

"I'm learning about THE CENTRAL COPIER CLOSET."

"I'm going from THE CENTRAL COPIER CLOSET to PAT'S OFFICE."

"Go to the map of Russia"

"I don't know about THE MAP OF RUSSIA yet."

"Go to the water cooler"

"I don't know how to get to THE WATER COOLER."

"Go to the conference room"

"Now I'm going to THE CENTRAL COPIER CLOSET."

The robot replans to get south to the conference room via the copier closet. Later, it says

[2] Here our picture differs from what we intended when we wrote the dialogue, in that there is not an obvious obstacle to stop the robot. Please assume for purposes of this example that the conference room door is usually closed, and the robot stops short of it.

“This is THE CENTRAL COPIER CLOSET.”

“Now I’m going from THE CENTRAL COPIER CLOSET to THE CONFERENCE ROOM.”

And once it arrives,

“I have arrived AT THE CONFERENCE ROOM.”

1.6 Evaluation Criteria

This section explains evaluation criteria that apply to any solution to the CMRT. It points ahead to a place where we apply these criteria to our solution, and invite readers to do the same.

The most straightforward criteria to apply are the requirements stated in the task itself. That is, a proposed solution to the task should exhibit the performance characteristics described above in Section 1.1. In particular, it should be able to associate names provided by a tutor with places in the environment based on direct or indirect descriptions. It should be able to use those names in responding appropriately to navigation requests or user queries. Judgment of its ability to perform this task well is inherently subjective, however, in our analysis of our own solution we have tried to present a balanced account in Chapter 11.

One of the strongest evaluation criteria we may apply to solutions to the CMRT is the requirement that people other than its author be able to use the solution in new environments. We have also applied this criterion to our solution, and documented the results in Chapter 8.

1.7 Contributions of the Thesis

This section describes the contributions this thesis makes.

1. Defines a clear problem, the communicating mobile robot task, work on which is relevant to the more general problem of interfacing natural human communication with behaviorally effective task solutions.

2. Provides a solution to the CMRT which integrates existing work on reactive navigation architectures and natural language systems through a mediating representation we define.

3. Introduces ROPs, a representation which is natural for integrating odometric information with more robust sensed plan-step termination conditions.

4. Identifies the strengths and weaknesses of a variety of choices of communication modality for the CMRT.

1.8 Preview

This section outlines the remainder of the thesis. Chapter 1 has presented the Communicating Mobile Robot Task, motivated it, and provided a preview of our solution. Chapter 2 explores a variety of possibilities for the communication modality to use for this task, and explains why we chose natural language. Chapter 3 describes natural language systems in detail, explains the implementation of our natural language system, and documents the language subset understood and produced by our system and how we decided what to include.

Chapter 4 characterizes the performance of the reactive component of our system, including the physical robot TJ and its subsumption software. It also explains which aspects of this robot system are essential to the operation of the rest of the work we describe and why.

Chapter 5 contains a grammar of the knowledge representation we have designed for places and plans. Chapter 6 explains how these representations support the robot's navigation ability, and Chapter 7 explains how they support its ability to communicate in natural language.

Chapter 8 documents the results of our experiments with the system, with an emphasis on relating those results to the evaluation criteria we set out in Section 1.6. Chapter 9 places this work in relation to a variety of other work on robot navigation and embodied natural language systems. Chapter 10 describes work that might be performed as natural extensions to the work done here, and Chapter 11 offers our conclusions as a result of this thesis.

Appendix A concerns the use of simulators in this research and generally in mobile robot research. Appendix B contains the text of a small human study we performed in which we asked subjects for sample dialogues with a mobile robot.

Chapter 2

Choices for Communication

There are a variety of ways we could choose to communicate with a robot while it solves the communicating mobile robot task. This chapter surveys some of the strong contenders, and explains why we chose to use natural language. The choices for communication we consider here include maps drawn by robots, maps drawn by people, gestures, buttons, and spoken and typewritten natural language systems.

2.1 Maps Drawn By Robots

One natural choice is to have the *robot* draw the map it is constructing in a way that it will be familiar and recognizable to people. A tutor can then name places on the map by, for example, clicking on the map and then typing in a name or selecting an icon. Alternatively, users could just click on the map directly to indicate goals to the robot.

Several researchers describe systems that learn maps and draw them in human-recognizable form. Notably, [Connell, 1992b] demonstrated such a system using the same physical mobile robot and low-level reactive control system that we used in this research.

This choice has several advantages. The robot doesn't have to drive to places while they are being named, although it has to have driven there once before while

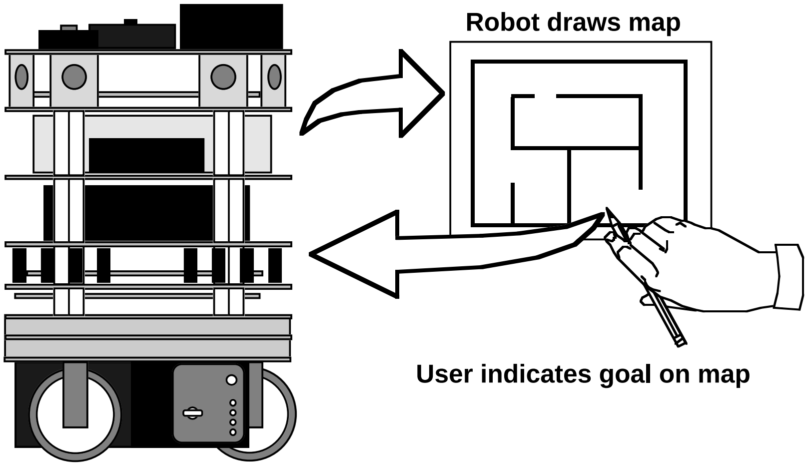


Figure 2-1: Maps drawn by robots

constructing the map. There is not much of a language problem to solve. Selecting a place on a map drawn by the robot is very precise with respect to location, and may already be in the robot's terms.

There are also disadvantages. Maps drawn in this way will lack details helpful to people as context for their place-naming, such as posters on the wall, wall-openings that were closed (e.g. by doors) when the robot toured the world to construct the map, and other features that are present in the world but would be missing from the map. In addition, this approach requires the robot to gather more detail, to construct an accurate map for people to use, than the robot may need to navigate in the environment. Finally, the robot must have explored the environment in advance, before users can give it goals.

2.2 Maps Drawn By People

Another approach is for *people* to draw maps or sketches to communicate ideas about where they want the robot to go. This approach involves a potentially greater recognition and understanding problem than even linguistic approaches, but there is a promising new idea which may ease this burden.

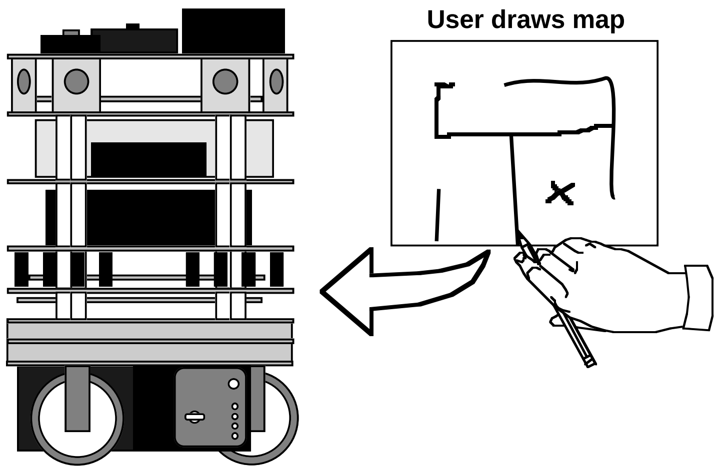


Figure 2-2: Maps drawn by people

Lynn Stein, in [Stein, 1991], proposes an imagination system in which people inform a robot about its environment in advance by drawing a map for it. The robot explores the map directly, using a simulator, rather than trying to interpret it in some abstract spatial terms unrelated to its navigation strategy. This has the advantage that the robot imagines, or predicts, the experience it will have in the real world, and thus constructs a representation for its subsequent navigation that should model the world well.

We might imagine extending this approach to handle naming of important places, say by indicating to the simulator that the robot should be told “this place is important” when it reaches a certain mark. In this way, the imagination system could be applied to the communicating mobile robot task.

This approach is only effective to the extent that the simulator models the world well enough for the robot to construct the same spatial representation when operating in the simulator as it does in the environment. As we have argued in [Torrance, 1992] and in Appendix A, the simulator does not need to model every aspect of the world; just those that are relevant to the distinctions the robot is making. For example, if the robot is thresholding a sensor value and then acting on its value as a predicate, the simulator needs only to predict which of the two possible values the virtual sensor will take on. However, in practice we expect that for the more complex sort of robot that could remember places as specified in the communicating mobile robot task, metric aspects of the environment will become important to a degree that renders it remarkably hard to construct the simulator the imagination method requires.

If there is a mismatch between the world imagined by the robot and the real world, either because of simulator idiosyncracies or because of discrepancies between the drawn map and the real world, the robot will need to be able to revise its internal representation in the face of new facts about the environment. Doing this revision while maintaining learned information about important places could become arbitrarily hard, especially given the modest sensing capabilities of most mobile robots. We would expect such discrepancies to arise frequently in practice.

2.3 Gestures

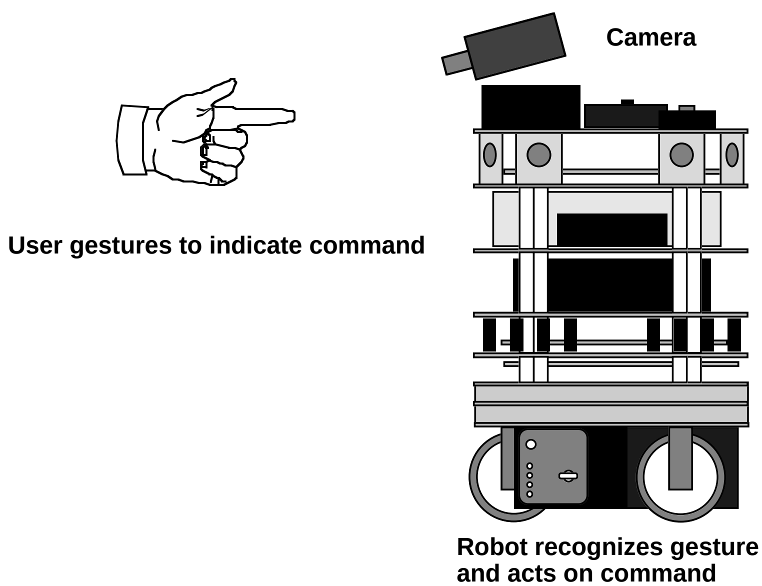


Figure 2-3: Using gestures to indicate commands

Gestures are another form of non-linguistic communication that we could use to communicate intentions and goals to a robot. These could include pointing to a place or standing at it to indicate its importance, pointing in a direction you want the robot to go, or moving or waving an arm or leg to convey some signal or ask a question.

As you can see from these examples, gestures are more appropriate for some types of communication than for others. [Horswill, 1993a, Horswill, 1993b] constructed a tour-giving robot in our lab that uses vision for navigation and gesture recognition. His robot Polly speaks with a voice synthesizer, and invites people to "wave your foot around if you would like a tour." This is an example of a non-intuitive gesture that happens to be one the robot can recognize, but it plays a reasonable role in communication since the robot explains its meaning.

The kinds of gesture recognition that would be most useful and effective for communication are those that already have a meaning to people, such as pointing, standing at a place, and holding a hand still to mean "stop." These particular gestures may be hard to recognize computationally. [Horswill, 1993b] also developed a cheap nod detector that could be used on a robot with a camera pointed up.

In any case, there are communicative aspects of this problem that do not lend themselves naturally to gestures. Among these are naming a place for use in a command issued later, asking the robot to go to a particular place that isn't within sight of the tutor or the robot, giving the robot information about a place that isn't nearby, and asking questions of the robot. So even if gesture recognition were employed in its natural roles, there would still be a need for other forms of communication.

2.4 Buttons

By now, the reader may be asking, "Why can't we just put some simple buttons on the robot?" Perhaps the robot could be equipped with a control panel full of buttons labelled "Important place 1," "Important place 2," and so on. A user presses "Learn" and then names a place with "Important place n." Then later the

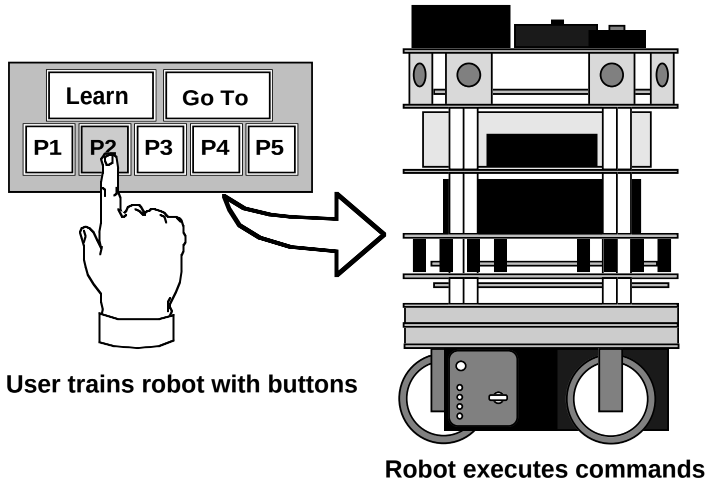


Figure 2-4: Using buttons to train and command the robot

user presses "Go-to" and selects a place-button. In fact, [Mataric, 1992] presents a mobile robot with just such an interface.

This does describe one aspect of the natural language interface we currently have implemented on our robot, with one important exception. Our system allows its users to use terms *they* think of for these places. This makes them memorable to humans, and allows the robot to benefit from the same common names that facilitate communication among people about places.

Furthermore, our move to a natural language system, albeit a simple system, reflects our conviction that the work we have done on spatial representations that support navigation and communication will be applicable to more sophisticated kinds of linguistic communication than we have explored here. We will have more to say on this in Chapter 3.

2.5 Typed Language

The modality of natural language, whether spoken or typewritten, supports the natural naming of places, explicit questions and answers which may refer to remote places and objects, and the statement of goals to go to previously named

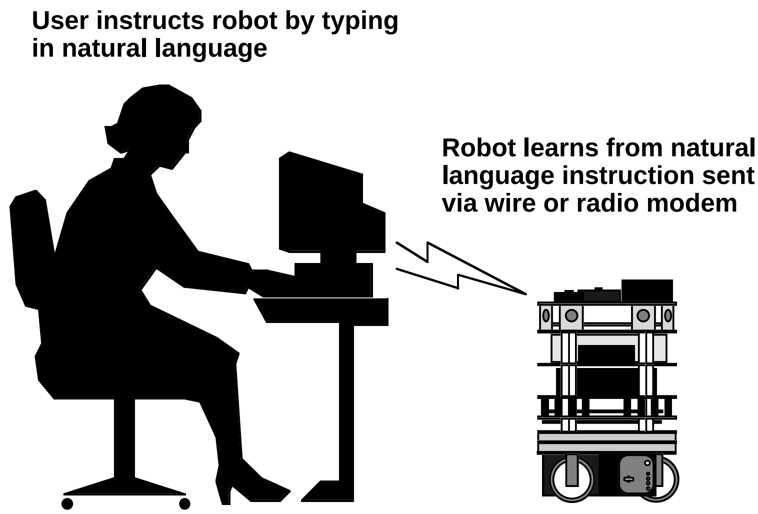
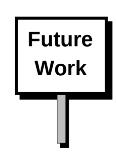


Figure 2-5: Using typewritten language to communicate with a robot

places. Natural language systems also offer opportunities to apply discourse theory that has been developed for natural language communication but less so for other modalities [Terveen, 1993].

The disadvantages of typed language interaction with the robot stem from the need for a keyboard, which would tend to preclude a free and open interchange while the robot is navigating through the environment. We have come up with a satisfactory solution to this problem by letting the user and the robot converse in either of two ways: on a fixed workstation connected to a radio modem, through which the system communicates with the untethered mobile robot, or on a portable computer that can be worn around the neck like a candy vendor's tray and tethered to the robot with a serial-communications connection.



We plan to explore applying the work developed in this thesis to another, vision based robot in development by Ian Horswill. The interface to this robot may include a video display on any workstation in our lab of what the robot is seeing at the moment, which would help somewhat the problem of remote place-naming.

In our system, the user interacts with the robot by having a typewritten conversation in English on a computer. The user can freely intersperse any of the following types of speech acts set out in the communicating mobile robot task:

- direct commands such as *go*, *stop*, *turn left*, and *face north*

- informative statements, such as *you are at John's office* or *you are lost*

- long-range goals, such as *go to Mary's office*

- questions, such as *Where are you?*, *Where is John's office?*, or *How would I get to John's office?*.

Spoken language output is easy and is supported on our robot; the user can choose whether to have the robot's utterances spoken or just displayed. More detail about the natural language system we have developed is found in the next chapter.

2.6 Spoken Language

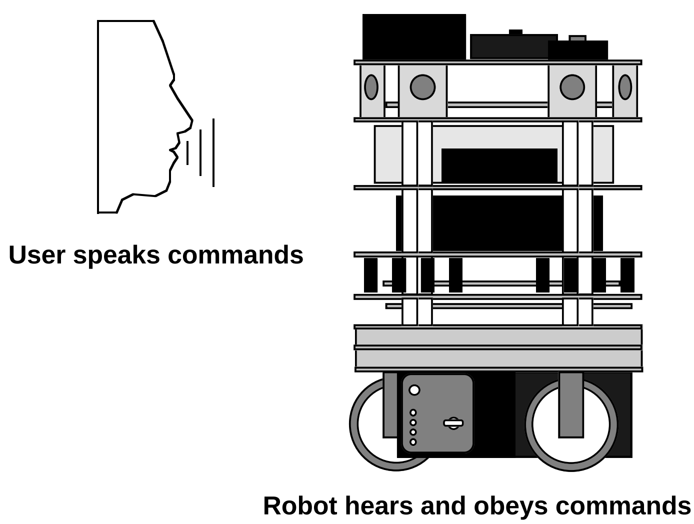
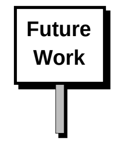


Figure 2-6: Using spoken language to communicate with a robot

An ideal communication system for a mobile robot would allow users to interact with the robot in any and all of the ways they find natural. Speech is a predominant form of human-to-human communication of instructions, explanations, and goals. Speech processing has the advantage that the robot system could potentially support the rich variety of natural language that affords people efficient communication, in a way not encumbered by the requirement of a keyboard and display.



State-of-the-art speech recognition has already reached the point that it would probably be practical for the application we have defined here. In supporting named places that use vocabulary not previously taught to the system, more sophisticated speech processing techniques would need to be used. In particular, representing the poorly-recognized words by their waveform for later matching, rather than fully interpreting each sentence, would enable a speech processing system to readily support this application.

In developing this thesis, we chose not to try to incorporate current speech recognition technology. As we assessed the problem, we found the introduction of speech recognition into this system would have served mainly orthogonal goals. There is a significant amount of interesting research progress regarding the tighter integration of language understanding, including speech recognition, and situated action. We look with anticipation to the results of these attempts, and believe that connecting present-day commercial speech recognition to the system we have built would serve the interests of demos, but not science.

Chapter 3

Natural Language

This chapter describes the use of natural language in our system. It begins with a discussion of the constraints the communicating mobile robot task places on the natural language system and the ways in which they have motivated our restrictions on the natural language system we employed. It goes on to describe the kinds of language our system currently supports.

3.1 Traditional Natural Language Processing

Traditional Natural Language Processing systems decompose the problem of language understanding into a particular series of steps. *Syntactic analysis*, or parsing, is concerned with determining the structure of an utterance in terms that are independent of meaning. It produces a parse tree that represents this structure. *Semantic analysis* operates on this parse tree and produces a semantic representation that should contain meaning by virtue of its connection to a system that connects it to some world the utterance concerns.

Our main complaint with this model is that it is primarily designed for one-way processing, from sound utterance to textual representation to parse tree to semantic interpretation to world. It makes little allowance for the impact of context or other facts about the world on the disambiguation of speech recognition, syntactic ambiguity, or context-sensitive semantics.

Many people have worked on this problem, most resolving it by changing what data is sent upstream. For example, modern speech recognition systems return several likely strings with their associated probabilities of being correct. Parsers reject some of these strings as ungrammatical, and pass on multiple possible parses where there is ambiguity. Semantic interpretation can often reject certain possibilities as nonsensical, and it in turn passes on the interpretations of the remaining possibilities. Ultimately, the system that will use the semantic representation must decide what was meant, or engage in discourse to ask for clarification.

This architecture works fine as far as it goes, but some are coming to question this decomposition of the problem. Parsing language into a syntactic representation depends on an idealized model of language; extending parsers and lexicons to cover truly natural language can be quite hard. Semantic interpretation that operates without the benefit of knowledge of the current state of the world and of the robot's mind will produce descriptions that are overgeneral. Knowledge about the environmental, mental, and discourse context for the utterance is crucial to proper interpretation.

New approaches offer hope for a way to break the grip of this traditional architecture. Charles Martin has done work on a natural language system that connects language understanding intimately to action [Martin, 1993].

3.2 Restricted Natural Language

This section describes ways in which constraints from the task we have chosen impact the design of the natural language system.

The communicating mobile robot task involves a particular kind of natural language problem. Following the methodology in Section 3.3, we determined that we only need to support a limited variety of English sentence constructions. Thus, we a very constrained grammar will suffice, but we would like to be able to use arbitrary names for places without having to make lexical entries for every word we might use.

One solution to this problem is automatic lexical type inference from context. This involves a natural language system which can infer the lexical type, or syntactic category, of words it does not know. It does this by using the known words in the sentence to constrain the possible categories of the unknown words. This works well when only one or two words per sentence are unknown (REWORK: need a reference).

Another solution, and the one we've adopted, is to constrain the grammar enough that we can tell for sure when we're parsing a noun phrase, and then allow those to contain arbitrary text. Our current implementation of the system works this way, with a pattern-matching input recognizer in place of a parser.

We are not wedded to this pattern matching approach. As lexical inference technology improves, it will become an attractive alternative to the system we developed. In practice, however, we find that relying on names stored as uninterpreted strings works fine for the kinds of names people use to describe places in our environment.

Occasionally people would like to use more than one description for the same landmark. One that came up in our experiments was “the map of Russia” vs. “the poster of Russia”. We have provided language support for synonym creation, as in “the map of Russia is the same as the poster of Russia.”

A simple extension could yield a hybrid solution, in which the user's sentence is first passed to a parser which breaks out complicated structure, and then to a pattern matcher. The pattern matcher could perform string matching against the components of the broken out parts of the sentence. Alternatively, it could try to match the whole input sentence if the parser did not recognize some parts of it. This hybrid approach would combine most of the power of a traditional parser with the flexibility of the pattern match system in which the full lexicon does not need to be known in advance.

3.3 Methodology of Language Support

This section describes the way in which we determined what set of speech acts to support. Most of the language support was developed according to our needs. It was influenced by a small human subject study we performed, by the comments of observers who watched the author use the system, and by other users of the system.

Our goal in including the set of things we did was not to exhaust all possibilities a user might expect on interacting with a natural language system in this context. We did, however, try to support the main types of instructive, informative, and interrogative speech acts that we believe are critical for solving the communicating mobile robot task.

The small study involved only two subjects. We asked them to write a sample dialogue they might expect to have with a robot, with speech acts in each of the categories we describe above. We further asked these subjects to write their sample sentences in increasing order of difficulty as they perceived it. We were hoping that sentences which might require arbitrarily hard perception or reasoning ability would be reserved for the end of the sample dialogues. The text of the questionnaire for the study, and the full dialogues written by the subjects, are included as Appendix B.

Our primary purpose in conducting the study was to look for examples of language use which were, in our opinion, fundamental to the task, but which we had failed to support to date. For this purpose the study was a success.

The nature of the responses varied considerably. It is dangerous to generalize from just two responses, but our observation is that the first subject, who was more familiar with the capabilities of existing mobile robots, was more restrained in his expectations of the natural language interaction. The real lesson we learned in conducting this study was that we *drastically* underspecified the problem, or the characteristics of the robot, so that both participants expected fully AI-complete problem solving behavior from the interaction.

The choice of what language to support was also influenced by the comments of observers who watched the author use the system, and by the other users of the system who used it in support of our evaluation criterion. Their use of our system is documented in Chapter 8.

3.4 Language Supported By Our System

This section describes the set of language that is supported by our system, including where needed a description of the behavior that language makes the system have.

We use the following conventions. *Direction* refers to any of the eight directions "north", "northeast", "east", "southeast", "south" "southwest", "west", or "northwest".[1] In addition, these may be abbreviated with one or two character versions, as in "n", "ne", and so on. *Sentence* refers to a whole sentence as accepted by our parser. *Place* refers to a place name or place description, which may consist of uninterpreted noun phrases.

Square braces delimit optional text. Curly braces delimit a set of words separated by vertical bars, one of which must be provided. If the behavior of the system varies depending on which word is present, that fact is documented following the sentence in question. Things the user types are set in boldface type; the robot's responses are in normal type.

3.4.1 Statements

"You are {at | in | on} ***place*****"**

This is the primary mechanism by which the user trains the robot about new places. It is presumed that the user knows where the robot actually is, and that the user is being honest. The robot will learn this place, and will subsequently be able to navigate back here from any other place that it has experienced as connected to

[1] Note that some other relative directions, including "left" and "right", are supported in addition to these directions.

this place.

“You are facing *direction***”**

There are times when the robot becomes disoriented. This speech act provides a way for a user who notices the robot is incorrect in its claims about its heading to correct it.

“*Place* **is [to the]** *direction* **of** {**here** | **you**}**”**

“You are [to the] *direction* **of** *place***”**

“*Place* **is [to the]** *direction* **of** *place***”**

This type of speech act gives the robot some information about a place, but not enough to allow it to navigate there without further assistance. This information can be used when explaining how a user could get to the learned place, however, since the user has access to a richer set of perceptions than the robot does. In addition, the robot will record the relationship between the two places described (or between its current place and the place described). Each time the robot is at the known place and headed in the appropriate direction, until it has learned how to get to the new place, it will remind the tutor to tell it when it gets to the new place.

“*Place* **is [the same as]** *place***”**

This speech act lets the robot know of another name for a known place. It is at present an error to tell the robot two places are the same when the robot had already learned different paths to get to each of them, as we did not implement the required place-unification and plan-unification to support this capability.

3.4.2 Commands

Except for “faster,” “slower,” and their synonyms, each of these commands interrupts any other activity or planning the robot was doing.

3.4.2.1 Velocity

“Go”

This is the basic way the user tells the robot to move. It requests the robot to start moving forward in the current direction.

“Stop”

This asks the robot to stop moving. This might be used in preparation for naming a new place, for example.

“[Go] {fast | slow}[er]”

This sentence causes the robot to adjust its speed to meet the user’s preference. This reflects modes of operation specific to TJ; other robot architectures might provide a different set of choices that would need to be supported through language. They do not have any permanent effect since our system doesn’t store the robot’s speed or speed changes.

3.4.2.2 Heading

“[Turn] {right | left} [*degrees*** [degrees]]”**

“[Turn] around”

“Face *direction***”**

These commands cause the robot to turn relative to its current heading, or to turn to a certain absolute heading. They also make the robot stop moving forward, as the turn radius of the robot when it is moving depends on its speed. We found it safest to require the robot to stop when it is making a turn. When *degrees* is not provided to a turn left or turn right command, the robot turns 90 degrees.

3.4.2.3 Short Term Plans

These commands are ways to get the robot to move forward with a certain prespecified stopping condition. The robot will announce that it has arrived at its goal and stop when the condition specified in the sentence is achieved. Sentences grouped together here are treated as synonymous.

"Go [until you get] to the end of the hall"
"Go as far as you can"
These commands ask the robot to go until it sees something in front of it.

"Go until you can turn {right | left}"
"Go until there is an opening on your {right | left}"
These commands ask the robot to go until there is no obstacle in the requested direction.

"Go until you {can't | cannot} turn {right | left}"
"Go until there is no[t an] opening on your {right | left}"
"Go until there is {a wall | something} on your {right | left}"
These commands ask the robot to go until there is some obstacle in the requested direction.

"Go [about] *number* **{inches | feet | yards}"**
These commands ask the robot to go the specified distance.[2]

3.4.2.4 Long Term Plans

Commands in this category initiate behavior that may take some time to complete. This behavior remains fully interruptible if the user decides on a different goal or

[2] We happened to choose English measurements instead of metric, because the distances used by the communication interface to the reactive robot we used are expressed in inches. This conversion would be easy.

wants to teach the robot about some new place.

“Go to ***place*****”**

This is the primary method used to direct the robot to navigate to a place it has previously learned about. If the robot knows how to get to this place from where it presently is, it will plan a route there and begin to navigate it. The planning is performed again at each place it passes through along the way, because it is so quick and it might allow the robot to take advantage of new information it has learned.

“Go back”

This command would be unlikely to be used in practice by a tutor other than the author. It causes the robot to return to the most recent place it was sure of. The robot does this by reversing the high-level navigation actions it has taken to get to this point. It must substitute distance-measured termination conditions for ones that depend on other conditions such as the presence or absence of walls, since those conditions are usually different when travelling the same passage in the opposite direction.

3.4.2.5 Requests for Information

“Describe ***place*****”**

“Tell [me] about ***place*****”**

These ask the robot to provide a simple description of the location of *place*. We considered supporting something much more complicated here, including the modelling of which places the user knows about already so the robot can cast its description of *place* in terms of those places. In the end, we decided that project wasn’t central to our thesis.

3.4.2.6 Utility

"Forget where you are"

This makes the robot stop and forget where it thought it was. This is useful if the robot has become confused about its location, and the user wants to command it using go, stop, and turn commands to get it back to a known location, without having the robot mistakenly learn this route as the way to get from where it erroneously thought it was to the place the user finally gets it to.

"Forget {all | everything}"

This completely refreshes the robot's memory when things are really messed up.

"Forget about *place*"

This makes the robot forget anything it has learned about the named place. This includes all plans that get it there from other places. This might have the effect of disconnecting parts of the robot's graph of places if the robot has learned lots of other places already; it is usually a good idea to teach it *place* again as soon as possible.

"{Load | Save} [*filename*]"

These speech acts allow the user to store and refresh the robot's memory of the places and plans it has learned. This is particularly useful for allowing the robot to explore different locations and not have any danger that they might conflict in its memory.

3.4.3 Questions and Responses

"Which way are you facing?"

This simple question asks the robot to name the direction it currently believes it is facing. Odometry error can cause the robot to become confused about its heading; the tutor can correct this confusion by telling it its correct heading.

“Where are you {going | headed}?”

This asks the robot to name the place it is headed for if it is presently on its way to a place.

“Where {is TJ | are you | am I}?”

These are ways of asking the robot to name its current location, or to name the places it is between when it is not at a particular place. It is mostly useful as a way of confirming that the robot is doing the right thing.

“Where is ***place*****?”**

This has the same effect as “Describe *place*” and “Tell about *place*.” See the description provided above in the Requests for Information section.

“What is [to the] ***direction*** **of {here | you}?”**

“What are you [to the] ***direction*** **of?”**

These sentences are trivial to implement given the rest of our architecture, but we did not have time to implement and test them. These are ways of asking the robot to describe nearby places that it knows about. Because the robot stores its plans in terms of directions in world-coordinates, it can determine this kind of information directly from those plans.

“How {do | would} {you | I} get from ***place*** **to** ***place*****?”**

“How {do | would} {you | I} get to ***place*** **[from** ***place*****]?”**

These questions ask the robot to describe routes between places or from the current place to another place. The robot uses a different strategy to describe these routes depending on whether the user is asking how TJ gets there or how a person would get there. If the former, TJ will explain that it can't get to places it hasn't learned yet by experiencing them.

We expect that in intended use, the person asking these questions doesn't already know the answer. Even if they are unfamiliar with the robot's environment, people following the plan that the robot provides should be able to find the place once the plan gets them close. For this reason, the system will include directions at the end of the plan such as "Then go north until you see John's office," which the robot can't follow itself, but which a person can.

3.4.4 Modals

We have begun to implement the described behavior of these sentences, but have not yet fully tested them.

"{**Then** | **Next** | **Finally**} *sentence***"**

Sentences that start with one of these words cause the robot to defer the command that follows until it has completed its other pending commands. This provides a way to tell the robot to

> **Go to the end of the hall.**
>
> **Then turn right.**
>
> **Finally go about 10 feet.**

This capability is an important precursor to a robot system that would be able to understand and store full plans without simultaneously executing them.

The author has worked on another system that can perform this task with a simulated robot and a natural language interface, described in [Torrance, 1994].

Chapter 4

The Reactive Layer

This chapter describes the reactive layer of our architecture, the basic navigation strategy of the robot TJ as implemented by Jonathan Connell. It also describes the robot TJ to the extent needed to understand how it accomplishes obstacle avoidance and corridor following.

4.1 Physical Architecture

The mobile robot TJ, illustrated in Figure 4-1, was designed and built by Jonathan Connell at IBM T.J. Watson Research Center [Connell, 1992b].

The robot is constructed on a Real World Interfaces (RWI) B-12 base, a 12 inch diameter commercial product used frequently in research on mobile robots. The remainder of TJ's hardware sits in separable sections of the same diameter above the base. Wiring between the sections runs through a series of holes down through the rear of the robot, much like a spinal cord. This architecture makes it relatively easy to interchange parts or to do repairs to one or more sections independently.

The cylindrical symmetry of the robot is important, because it makes it possible for this robot to rotate in place without danger of interacting with static obstacles. This isn't entirely accurate, since there are openings in the surface of the robot which can get caught on protrusions, for example, but in practice turning in place is quite safe.

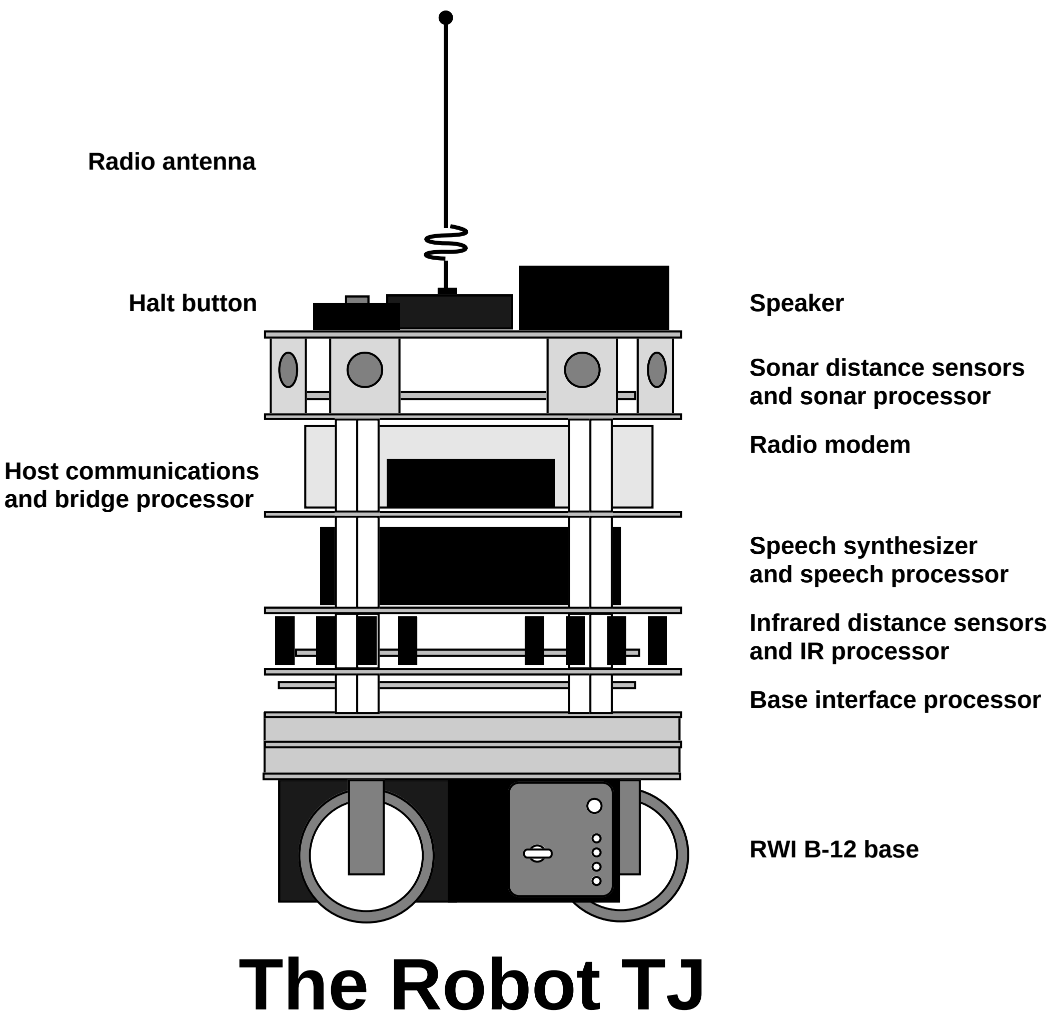


Figure 4-1: Components of the physical mobile robot TJ used in this research

4.1.1 Sensors

TJ's sensors include infrared and sonar distance sensors as well as the extensive support for odometry that is built into the RWI base and a radio modem.

Twelve infrared sensors are mounted in a ring, with heavier concentrations in the front and sides of the robot. These sensors are used primarily for corridor or wall alignment and obstacle avoidance. Three higher power infrared sensors, tuned to a longer range, are mounted facing directly forward and to the left and right of the robot's travel direction. These are used for obstacle avoidance and to detect wall openings, which may correspond to open doors or side passages, at times when the robot is aligned with a corridor. A dedicated 6811 microprocessor, the *IR Processor*, collects these distance measurements and reports them when requested to the bridge.

The infrared sensors are most effective on moderate to light colored surfaces that reflect infrared light well. They are also confined to a single layer of the robot, about 9 inches from the ground, so they can't detect obstacles at other heights.

Sonar sensors are employed to help overcome these problems. These are Polaroid ultrasonic ranging transducers, connected to custom analog signal processing hardware. This hardware, designed by Connell, does more analysis of the time domain signal from the sensor than is typically done in applications that employ sonars. As a result, TJ's sonars are often able to detect the distances to each of multiple echos from a single ping. This same board contains the *Sonar Processor*, a dedicated 6811 that collects this data and returns it to the bridge when so requested.

The version of the RWI base TJ uses computes fairly accurate odometry measurements of the total distance the base has translated and rotated since it was reset. (A newer version of this base has greatly improved odometry; unfortunately, we did not use the new version in this research.) The *Base Interface Processor* integrates this information over time to develop an estimate of the robot's current x, y, θ position relative to its starting position and orientation.

Data communications received over the radio modem or through an RS-232 cable that bypasses the modem, comes into TJ through the *Host Communications*

Processor, and provides input to the *Bridge Processor*. This allows the host computer, in our case a Macintosh Powerbook, to send commands to the robot as described in Section 4.2. Among these commands are strings to be said by the voice synthesizer when it is enabled, so this direction of communication serves in the output side of the natural language system.

4.1.2 Effectors

TJ's primary effectors are the wheels on its RWI base. In addition, it is equipped with a speech synthesizer and amplified speaker, and a radio modem.

The RWI base is a nearly holonomic drive system. This means translation and rotation can be controlled independently. The base is only nearly holonomic because the point of contact for each wheel is not precisely beneath the vertical rotation axle around which it rotates. This means the wheels must slip some on the ground when the base rotates. This effect is negligible, however, compared to the slip that can be introduced by uneven floor surfaces such as carpet. The Base Interface Processor supports drive commands to these processors.

The speech synthesizer provides spoken output from English text input with a quality no better or worse than typical commercial speech psynthesizers. We find it to be adequate for our needs, especially when it is used in a discourse setting where the other conversant has expectations about what the robot will say.

The radio modem (or its substitute RS-232 cable) provides sensory information and command feedback from the processors on TJ to the host computer, an external computer to which the robot is linked and which runs the other levels of our architecture.

4.2 Computational Architecture

Code written in Lisp communicates with the TJ system through the Host Communications Processor and the Bridge Processor. These processors provide pass-through commands to each of the processors in the style of remote procedure calls, and the

Bridge Processor itself implements subsumption architecture [Brooks, 1986] by providing for command inhibition.

The robot has a set of virtual effector resources that can be turned on or off by the Bridge Processor. These include fast and slow versions of turn left, turn right, translate forward, and translate backward. A higher level program sets up which of these resources should be available at any time; to restrict the robot to slow motion, for example, all of the “fast” resources are inhibited.

Programs that run on each of the processors, called “behaviors,” can provide “advice” about which resources should be inhibited. One behavior that runs on the IR Processor, for example, will inhibit the robot’s rotation toward an obstacle it percieves. Another behavior runs on the Sonar Processor, and slows or stops the robot when there is an obstacle in front. The higher level program in Lisp can, over the serial communications link, determine which of these behaviors will be active at any time.

4.3 Performance

The capabilities of TJ employed in this research include ballistic rotation, odometric measurement, obstacle avoidance, corridor following, and discrimination of openings from obstacles on the left, right, and in front. This section reviews TJ’s performance on each of these tasks.

TJ is nearly flawless at ballistic rotation. The wheels do slip on the carpet, so the angle turned through is not always the angle requested. The routines that implement this ballistic rotation are in the RWI base ROMs, and return a value that accurately reflects the change in odometry from before the turn to after. Still, slippage on the carpet can play a role. A mistake of three or four degrees at the start can translate to a big error in position after travelling a long distance. Fortunately, we operated the robot in environments where the relative scarcity of big open spaces helped us in this case. Hallways provide the robot a chance to realign and correct for errors in its odometry. To be fair, in many applications where wide open spaces are common, carpet is less so, and bare floor improves the rotational accuracy of bases like the RWI B-12 tremendously.

For translation, the TJ architecture provides a way to measure the robot's progress in inches *in the commanded direction*. This means that, to the extent the robot's rotational odometry is correct, it can get a good estimate of how far it has travelled down a corridor even in the presence of the wall following behavior. This is important, because the raw translational odometry could vary a lot from run to run down the same corridor, depending on where and how often the robot happens to adjust its heading.

At obstacle avoidance, TJ has some trouble. Its likelihood of stopping for a person who steps in front of it, for example, depends dramatically on the color of the person's pants. The IR sensors can see white and light colored pants, and miss the dark pants. We would have expected the sonars to do well here, but they don't. We haven't yet undertaken a thorough investigation of this problem, so we suspend judgment until we have. Still, the robot's poor performance at moving obstacle avoidance deserves serious consideration when thinking about how to integrate this robot into a complete system that can operate safely in an office environment. We did not tackle this problem beyond our analysis here.

TJ is wonderful at corridor following. It can achieve speeds of better than 1.5 meters per second safely (in the absence of moving obstacles). Front-mounted sensors reliably detect static obstacles as the robot approaches them and their associated behaviors, when active, cause the robot to slow for the approach. Connell has developed an ingenious wall following strategy based on three carefully distance-tuned IR sensors that works very well.

The opening discrimination routine works well, again for static obstacles. It does not discriminate closed doors from walls, a performance requirement of some systems [Myers and Konolige, 1992b]. Consequently, the rest of the system using this routine must accept it for what it is, and try to use it only to detect the openness of a door the robot is pretty sure is there, or the presence of an opening the robot is pretty sure stays open. If both the position of the robot and the state of the opening are unknown, this discriminant is useless for localization.

4.4 Requirements of the Base System

This section describes the requirements we believe the rest of our architecture makes of the base system, and how other systems might meet those requirements.

We originally expected to require the base robot system to be capable of landmark recognition and point to point navigation among these landmarks. We intended that our ROPs would be used only for local navigation with a recognized landmark as a starting point, and that distant places would be acquired by first planning globally in the space of recognized landmarks and then using ROPs to go the final distance to the odometrically recognized place. [Horswill, 1993b] has presented a vision-based system which has this performance, increasing our confidence that this is reasonable to expect.

In performing this research, however, we determined that TJ does not have the appropriate sensors for landmark recognition except in the context of good odometric information. Since we intended to push the use of odometry anyway, we emphasized this and removed our requirement of landmark recognition. In the end, we believe landmark recognition would be a substantial improvement to the system we have implemented, as it should allow more robust performance. However, we find that the odometry we have employed as provided by TJ is remarkably adequate for the task in the absence of any other landmark recognition.

Obviously, high quality odometry is assumed by the rest of this research. Places will only be recognizable to within the error introduced by the measurement of translational odometry in the desired direction. Moreover, without landmark recognition, serious errors in translational odometry could make ROPs that rely only on distance measurements unusable. Luckily, the high quality translational odometry provided on TJ and its RWI B-12 base are now becoming standard, so meeting this requirement should not be a problem for many robots.

Chapter 5

Plans and Representation

This chapter describes our representations of places and plans. These representations form the primary contribution of this thesis, and serve as the bridge between natural language and the reactive robot. The following two chapters explain how these representations are used to support navigation and communication respectively.

5.1 BNF

The following is a Backus Normal Form grammar that expresses the relationships between our representational data structures for plans, places, and ROPs. More information about each of these data structures, including the semantics of their various components, is found in the subsequent sections.

$$
\begin{array}{rcl}
\mathit{plan} & ::= & (\mathit{rop}*) \\
\mathit{place} & ::= & \langle\, (\mathit{name}*), (\mathit{rop}*), (\mathit{rop}*), \mathit{preposition}, (\mathit{place}*), (\mathit{place}*) \,\rangle \mid \texttt{nil} \\
\mathit{name} & ::= & \text{any string} \\
\mathit{preposition} & ::= & \text{“at”} \mid \text{“in”} \mid \text{“on”} \\
\mathit{rop} & ::= & \langle\, \mathit{place}, \mathit{place}, (\mathit{step}*) \,\rangle \\
\mathit{step} & ::= & \langle\, \mathit{heading}, \mathit{length}, \mathit{stop\text{-}condition} \,\rangle \\
 & & \\
\mathit{heading} & ::= & 0.0 \quad .. \quad 360.0 \mid \texttt{nil} \\
\mathit{length} & ::= & 0 \quad .. \quad \infty \mid \texttt{nil} \\
\mathit{stop\text{-}condition} & ::= & \texttt{shortfront} \mid \texttt{longfront} \mid \\
 & & \texttt{shortleft} \mid \texttt{longleft} \mid \\
 & & \texttt{shortright} \mid \texttt{longright}
\end{array}
$$

5.2 Places

Place

Names
ROPs to me
ROPs from me
Preposition
Nearby new places
Nearby known places

Figure 5-1: Data structure for storing a place

A *place* is a data structure in our program that is used to represent a point location in the physical world. Places are almost always points that the robot can navigationally reach, although we do store places we have learned something about but for which we do not have a procedure to get there. In typical use in our system, which only drives the robot around the corridors of our building and does not attempt to drive into offices, the spot in the hallway in front of an office is named with the number and / or occupant of that office. When the user subsequently asks the robot to "go to Mark's office," it navigates back to the point in front of the office at which it was previously told "this is Mark's office" or "you are at Mark's office."

A place includes storage for a variety of names that are used to describe the place. The names include each name that has been used by a tutor to describe the place; these are introduced through language such as "you are at *place*," "*place* is north of here," "*place* is the same as *place*," or "*place* is east of *place*."

Each place also stores a set of ROPs that lead from this place to other places and a set of ROPs that lead from other places to this place. These serve to connect the places together in a graph, where each place is a vertex and each ROP is an edge. A variety of mechanisms ensure that the growth of this graph is controlled; Chapter 6 has more information on this topic.

A place also indicates the preposition, if any, used by the tutor to describe presence at the place. This is either "at", "in", or "on". If no preposition is used when the place is taught, such as in the sentence "This is *place*," the system uses "at" by default until it is told otherwise in a subsequent sentence.

In addition, two other lists are kept with a place. One contains a list of places that the robot has learned are nearby this place, but whose precise location is not yet known. These are called *nearby new places*. The other list is used when the place itself is only known by its relationship to other nearby places; those other places are stored as *nearby known places*. As the robot is told about places by their relationship to other places, it manipulates these lists. For example, if the user says "John's office is north of the conference room," the system will store JOHN'S OFFICE on the nearby-new-places list of THE CONFERENCE ROOM, and will store THE CONFERENCE ROOM on the nearby-known-places list of JOHN'S OFFICE.

This information is used in a variety of circumstances. When the system is at the conference room and begins to head north, it will ask the user to tell it when it gets to John's office. Additionally, when describing how a person could get to John's office, the system can tell the person how to get to the conference room and then suggest that the user "go north to John's office." This plan is not precise enough to allow the robot to navigate to John's office on its own, but it is useful information that could allow a person to get there. More information about this topic is found in Chapter 7.

5.3 Reactive-Odometric Plans

Figure 5-2: Data structure for storing a ROP

A *Reactive-Odometric Plan*, or ROP, is a representation of a short-range plan to get the robot between two places. These ROPs provide a way for the robot to understand a place in terms of a way to get there in the world. A ROP includes references to the places it leads from and to, which we sometimes call the *starting place* and *goal*, respectively, of the ROP. It also includes a list of steps.

The steps of a ROP provide the language of interaction between the reactive layer and the higher layers of the planner and the natural language system. We have determined that this language needs to have these properties:

1. Taken together, the atoms of this language are sufficient to get the robot to go to anyplace we desire it to go, and that it can physically reach.

2. These plan steps can be automatically recognized by a plan-recognition module, and therefore regenerated by a memory-based planner.

3. They can be told to the robot by a person.

The language of ROP steps we have designed meets these criteria.

A step of a ROP corresponds to one straight-line leg of the journey represented by the ROP. Each step includes three components; a *heading*, a *distance*, and a *stopping condition*. Each of the components is optional.

The heading, when present, indicates the direction the robot should face when beginning to execute that step. It is expressed in absolute world coordinates rather than being relative to the previous direction the robot was facing; this allows fairly straightforward reversal of ROPs. If the heading is `nil`, the robot continues on at its current heading when beginning to execute this step.

The distance, when present, reflects an estimate of the travel distance of this step of the ROP. When executing the step, the robot will travel the required distance and then stop or move on to the next step. In ROPs that the robot has actually used to navigate, distance measurements are stored for every step. These measurements are recorded directly from the odometric distance measured by Connell's software for TJ. As described in Section 4.3, this estimate reflects progress in the commanded direction, rather than recording the total progress of the translation motors of the base. This turns out to be quite repeatable and reliable.

This distance measurement on each step is also used by the system to compute a distance estimate for a complete ROP. This latter estimate is used in the high-level planner as the cost of using this ROP, as it works to find the shortest path between two places.

The stopping condition, when present, provides an additional condition under which the robot will consider itself done with this step of the ROP. At present, this condition can be one of these six possibilities:

- `shortfront`
- `longfront`
- `shortleft`

- `longleft`
- `shortright`
- `longright`

These represent sensory conditions measured by the robot. These conditions are introduced when a person gives an immediate mode command that involves measuring one of them, such as "Go to the end of the hall," "Go until you can turn left," or "Go until there is something on your right."

An example of a ROP that would get the robot to go north down a hallway and stop at the end, 120 inches away, would be:

$$\langle \text{PLACE } 1, \text{PLACE } 2, (\langle 0.0, 120, \texttt{shortfront} \rangle) \rangle$$

In this ROP, PLACE 1 and PLACE 2 refer to the system's internal representations for those places, and the single step of the ROP contains the heading command 0.0, the distance command 120, and the stopping condition `shortfront`.

For purposes of efficiency, only nearest-neighbor ROPs are stored by the system. When the system learns about a new place B that is between two other places A and C, it replaces the ROPs A⇔C with new ROPs A⇔B and B⇔C. In addition, the structure of the environment means each place will have few nearest neighbors. These factors restrict the growth of the graph of places connected by ROPs. This property is explored further in Section 6.4.

ROPs can be easily reversed. This is important since when the robot learns about B after travelling there from the known place A, the system needs to install both the forward ROP A⇒B and a reverse ROP B⇒A. The same algorithm is also used to reverse the current ROP to get the robot back to its most recent known place when it is asked to plan a route while it is between known places.

ROP steps that involve explicit stopping conditions will not normally work correctly when reversed. Consider a step that says to go until there is a gap on the right. There may be no natural stopping condition to indicate the other end of that leg of the journey, and even if there is it wouldn't be the same gap. Since every ROP step the robot has experienced is annotated with its distance, we sacrifice the robustness of the explicit stopping condition and just store a GO step that makes the robot travel the right distance in the reverse direction. This makes every ROP reversible.

By way of example, consider the ROP with these steps:

$$(\langle 0, 55, \texttt{longright}\rangle \ \ \langle 90, 20, \texttt{nil}\rangle)$$

These steps, when reversed, would look like this:

$$(\langle 270, 20, \texttt{nil}\rangle \ \ \langle 180, 55, \texttt{nil}\rangle)$$

The first of the original steps becomes the last of the reversed steps. Explicit angles, where provided, are reversed.

If steps in the ROP have `nil` headings, the reversed ROP expresses them as though their headings had been copied through from the most recent step that had an explicit heading. This ensures that each step of the ROP will be traversed in the reverse direction in the reverse ROP.

5.4 Plans

A *plan* is a representation of a longer range plan that would get the robot between two places or between its current location and a place. It consists of a sequence of ROPs, with the goal of each ROP being the starting place of the next ROP in the plan, and the goal of the final ROP being the goal of the whole plan.

Plans are not stored in any permanent way, but are generated as needed by Dijkstra's well-known shortest path algorithm. Because in typical interactions with our system new places are frequently named during the execution of a plan, the planner replans at each intermediate place. That is, in executing a plan, the robot just executes the first ROP in the plan and then replans. This would allow the robot to take advantage of fortuitous knowledge added by the user during the execution of the first ROP, such as a fact about the connectedness of two places the robot did not know were connected.

Chapter 6

Navigation Support

This chapter describes the way in which the representations described in Chapter 5 support the robot's navigation.

6.1 Basic Execution Sequence

The basic execution sequence of the robot is shown in Figure 6-1. It consists of a loop, in which data is grabbed from the robot, natural language commands are processed, the current ROP step is executed, the robot's location is updated based on plan recognition, and the robot replans as needed to get to its goal. This loop is repeated continuously, as often as possible. Each of the steps in the loop is described in sequence in this chapter, save for the processing of natural language commands which is reserved for Chapter 7.

6.2 Grabbing Data from the Robot

Certain data is required from the reactive system that provides low-level control of the robot. This includes the current heading, the distance the robot has travelled along the commanded heading since it was last reset, whether the robot is currently moving, and the current values of the sensors.

Polled sensors used by this higher level code include the long-range IR sensors

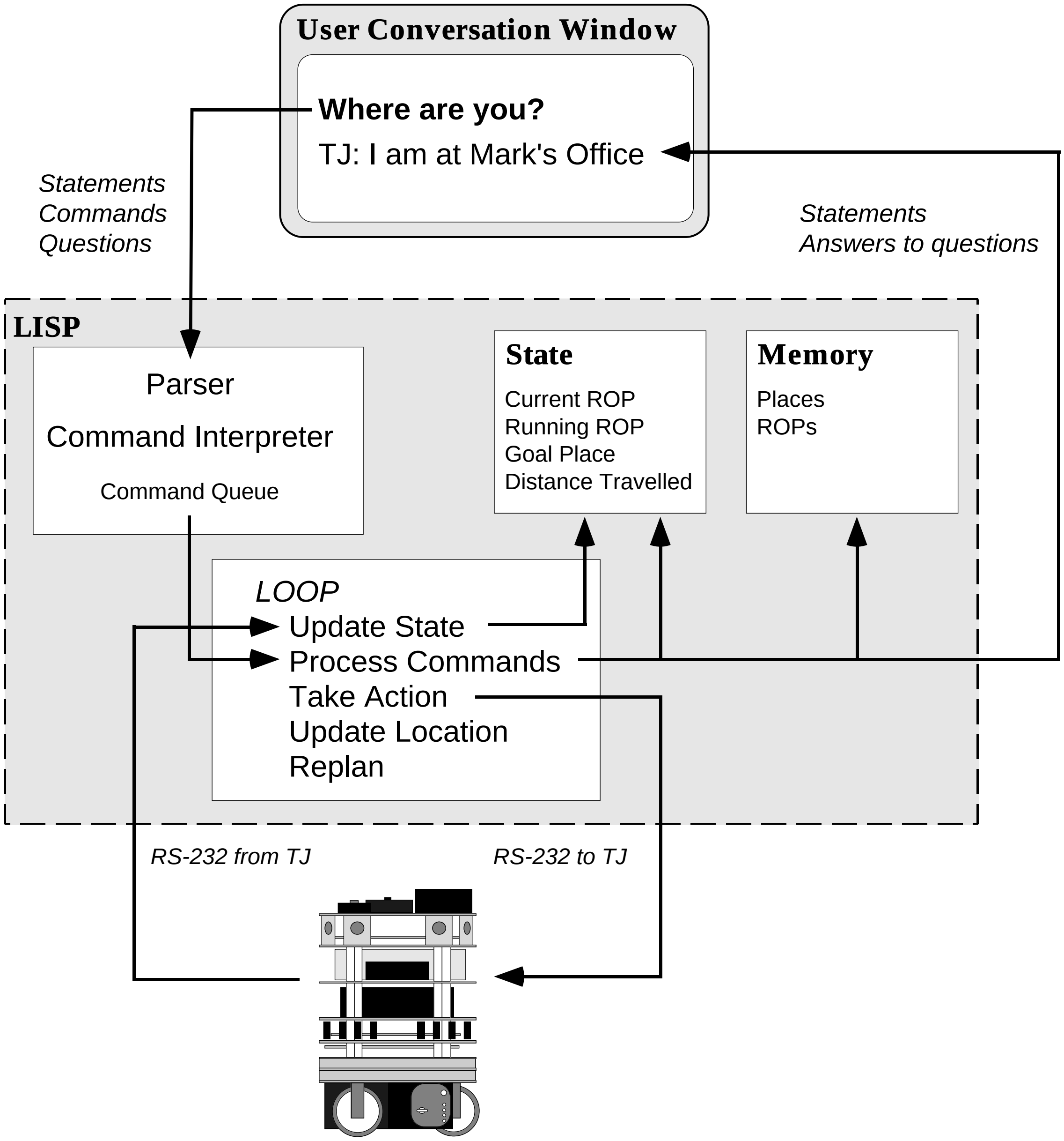


Figure 6-1: Overview of our system, showing *some* of the information flow. Processing commands and updating location can each produce natural language output; these may draw upon the memory and the state in so doing.

mounted in front and on the left and right of the robot, as well as the sonars. These sensors are merged to form three virtual sensors that reflect whether there is an obstacle near the robot to the left, to the right, and in front.

6.3 ROP Execution

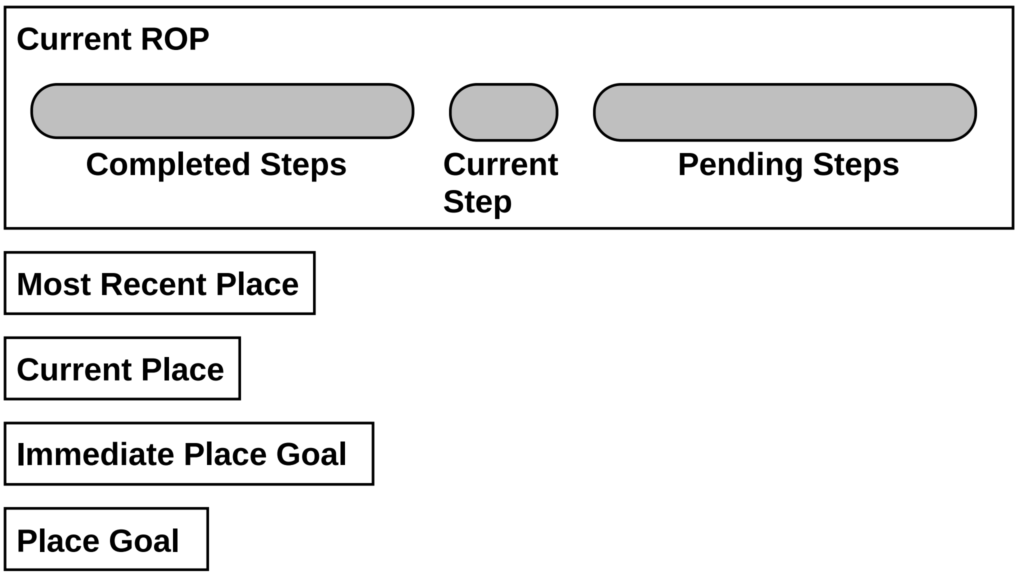
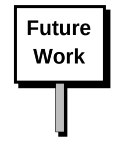


Figure 6-2: Components of the state of the robot

Figure 6-2 illustrates the variables that make up the robot's current state. At all times, `*current-rop*` reflects the robot's progress since its last known location, and contains steps that represent its current and pending navigation actions.

The current step may include a heading, a distance, and/or a stopping condition. The heading, if present, is used when execution of the step is begun, as a ballistic turn command to get the robot facing the proper direction. If the step contains an explicit distance, the ROP executor calls the step complete when the robot has travelled at least that distance. If the step contains a stopping condition, the executor monitors that condition and calls the step complete when it holds true. In the current implementation, if both conditions are present the robot will not stop until the stopping condition occurs; this increases its utility as a way of compensating for translational drift over time.



The algorithm leaves room for more sophisticated step termination conditions. For example, if both a distance and a condition are present, the robot could express some healthy skepticism if the condition does not occur at around the same place from execution to execution. Because the distance of each step is recorded in the completed-steps list, the robot would be able to return to a known place to try again or to ask for help when necessary.

6.4 Plan Recognition

As the robot leaves a place, it begins to track its progress along each of the ROPs it knows that start at that place. The robot recognizes that it has arrived at a place by noticing that the ROP it has executed so far is isomorphic to some ROP it is tracking.

This tracking problem does not become computationally complex because of a feature of the environment that works in our favor, and because of a pruning process we perform. The environmental feature is that places, viewed as graph vertices, have a relatively low branching factor. In our environment, as in many office environments, the robot's domain is largely composed of corridors and intersections. The possible travel directions from any intersection number at most four in our environment, and they happen to be in the cardinal directions in our building. Our system assumes the number is small, as this restricts the branching factor of the planning problem, but not necessarily that the paths are in cardinal directions.

The pruning process we perform removes a ROP that is superseded by a pair of ROPs which together achieve the path of the original ROP. These shorter ROPs are introduced automatically whenever a new place is named while the robot is on its way to a previously known place. For example, when the system learns about a new place B that is between two other places A and C, it replaces the ROPs A⇔C with new ROPs A⇔B and B⇔C. This pruning process means that the only ROPs stored from a place lead to adjacent known places, not to all known places. Thus, the low branching factor of the environment and the high density of named places with respect to this branching factor cause the branching factor of the place graph to remain low.

At present, ROPs only unify with each other if they were generated by the same source. When a ROP is recorded as the user commands the robot to drive around its environment, and the same ROP is later used as part of a navigation plan, the running execution of this ROP will unify with the stored version. That is, our current algorithm only detects the unity of identical ROPs. This is sufficient for recognizing places that the robot was expecting to get to because they were along the route of its high level plan (see Section 6.5, below), but does not allow sophisticated recognition of places the robot is surprised to have arrived at. For example, the system would not at present recognize that these two ROPs have the same effect:

$$\langle \text{PLACE 1}, \text{PLACE 2}, (\langle \texttt{nil}, 25, \texttt{nil} \rangle \ \langle \texttt{nil}, 25, \texttt{nil} \rangle) \rangle$$

$$\langle \text{PLACE 1}, \text{PLACE 2}, (\langle \texttt{nil}, 40, \texttt{nil} \rangle \ \langle \texttt{nil}, 10, \texttt{nil} \rangle) \rangle$$

These ROPs could have been constructed by two separate stop-and-go processes in each of which starting from place 1 the user said "Go," "Stop," "Go," then "You are at place 2."

We have designed and tested some preliminary improvements to this unification algorithm. We have augmented the ROP recording procedure so it merges together consecutive GO steps without explicit stopping conditions into a single step which says to go the sum of the old steps' distances; this eliminates the problem described in the previous paragraph. We have also fixed the implementation of the turn commands (such as "turn left" or "face north") so that multiple commands issued in sequence are represented as a single step that produces their net effect. We further merge a turn step without a translation command (i.e. with 0 distance) with a subsequent translation command that occurs, saving one combined step in the *completed steps* of the current ROP. These improvements are effective at making more ROPs correctly unify.

6.5 Planning

The robot can accept instructions that cause it to begin long-range navigation behavior to get to a particular place. These instructions come from natural language, and are discussed more in Section 7.2.4. When the user has asked the robot to go to a place the robot knows how to get to, it stores this place in the state variable `*place-goal*`.

When there is a place goal stored in `*place-goal*`, the robot will replan to get to it whenever it arrives at a known location. The robot replans because new information added while the robot executed the most recent leg of the plan could have changed which route is the shortest.

The planner uses the well-known Dijkstra shortest path algorithm to find the best route from its current location to `*place-goal*`. ROPs form the edges of a graph connecting the known places, so these are used as the steps of this plan.

A distance estimate is formed for each ROP, and used by the planner as it determines the shortest path. This estimate is formed by summing the distances of all the steps in the ROP.

Chapter 7

Communication Support

This chapter describes the way in which the representations described in Chapter 5 support natural language communication with people. It parallels Section 3.4, which details the intended behavior of each of the speech acts described in this chapter. Section 3.3 provides the methodology behind our choice of what language components to include. This chapter emphasizes the implementation of these speech acts.

We use the following conventions. *Direction* refers to any of the eight directions “north”, “northeast”, “east”, “southeast”, “south” “southwest”, “west”, or “northwest”. In addition, these may be abbreviated with one or two character versions, as in “n”, “ne”, and so on. *Sentence* refers to a whole sentence as accepted by our parser. *Place* refers to a place name or place description, which may consist of arbitrary text.

Square braces delimit optional text. Curly braces delimit a set of words separated by vertical bars, one of which must be provided. If the behavior of the system varies depending on which word is present, that fact is documented following the sentence in question. Things the user types are set in boldface type; the robot’s responses are in normal type.

7.1 Statements

"You are {**at** | **in** | **on**} *place***"**

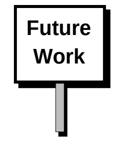
This is the primary mechanism by which the user trains the robot about new places. It is presumed that the user knows where the robot actually is, and that the user is being honest. The robot will learn this place, and will subsequently be able to navigate back here from any other place that it has experienced as connected to this place.

The robot does a number of things when the user tells it this statement. It terminates the current step, which sets its distance to the current odometric distance measurement of the robot and pushes the step onto the completed-steps list. If the named place was not already known, it instantiates a new place with the given name as its only name. It sets `*current-place*` to be this new place.

If the robot knew where it most recently was, that place will be stored in `*most-recent-place*`. In this case, it then stores the completed-steps as a ROP to get from `*most-recent-place*` to `*current-place*`, and also stores a reverse-rop to get from `*current-place*` back to `*most-recent-place*`. Section 5.3 describes the procedure for reversing a ROP. Likewise, if the robot was on its way to a named place (this would be stored in `*next-place-goal*`), it generates ROPs from the remainder of the terminated step and the still pending steps to get between `*current-place*` and `*next-place-goal*`. Last, it replans if it had a `*place-goal*` to take advantage of the newly installed ROPs.

Places may have multiple names; if the robot is already at a known place and it is told it is at *new place name*, it will just add *new place name* as a synonym for the place it knew it was at. Synonyms can also be introduced with "is" or "is the same as", as described below.

An easy extension to the work we have done here would keep track of the number of times users use each name for a place so the robot can use the most popular name for a place over time. A more sophisticated version of this extension would maintain a model of which user it was talking to, and store preferences for place names on a per user basis.



“You are facing *direction***”**

There are times when the robot becomes disoriented. This speech act provides a way for a tutor who notices the robot is incorrect in its claims about its heading to correct it. The robot interface provides a layer of abstraction between the heading reported by the base and the heading the robot uses. This abstraction is managed by an offset, which is added to the base-reported heading before it is reported, and subtracted from commanded headings before they are passed to the base control. The “You are facing” sentence changes the value of this offset so it makes the robot believe its current base heading is in fact the named direction.

“*Place* **is [to the]** *direction* **of** {**here** | **you**}**”**

“You are [to the] *direction* **of** *place***”**

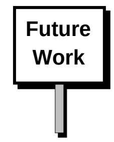
“*Place* **is [to the]** *direction* **of** *place***”**

This type of speech act gives the robot some information about a place, but not enough to allow it to navigate there without further assistance. This information can be used when explaining how a user could get to the learned place, however, since the user has access to a richer set of perceptions than the robot does. In addition, the robot will record the relationship between the two places described (or between its current place and the place described). It records the new place on the *nearby new places* list of the known place, and records the known place on the *nearby known places* list of the new place. Each time the robot is at the known place and headed in the appropriate direction, until it has learned how to get to the new place, it will remind the tutor to tell it when it gets to the new place. When it finally learns how to get to the new place, it removes the entries it made on their nearby-places lists.

"*Place* is [the same as] *place*"

This speech act lets the robot know of another name for a known place.

It is at present an error to tell the robot two places are the same when the robot had already learned different paths to get to each of them, as we have not implemented the required place-unification and plan-unification to support this capability.



7.2 Commands

All commands except speed changes share certain implementation features. They all cause the current step to be terminated, which sets its distance to the current odometric distance measurement of the robot and pushes the step onto the completed-steps list. In most cases, they kill all the pending steps, and install a new step as the only pending step. The contents of that step are described below after each command.

7.2.1 Velocity

"Go"

This is the basic way the user tells the robot to move. It requests the robot to start moving forward in the current direction. It is implemented by installing a new pending step with no heading, distance, or stopping condition.

"Stop"

This command stops the robot, causes the current step to be terminated, kills the pending steps, and has no other effects.

"[Go] {fast | slow}[er]"

These sentences operate by sending commands directly to the TJ interface to adjust the speed of the robot, but they are not recorded in the current ROP and otherwise have no effect on the operation of the robot. We found that speed changes do not affect the robot's odometry in any meaningful way, which makes it acceptable to have the speed changes independent of the execution of the ROPs.

7.2.2 Heading

"[Turn] {**right** | **left**} **[***degrees* **[degrees]]"**

"[Turn] around"

"Face *direction***"**

These commands cause the robot to turn relative to its current heading, or to turn to a certain absolute heading. They also make the robot stop moving forward, as the turn radius of the robot when it is moving depends on its speed. We found it safest to require the robot to stop when it is making a turn. When *degrees* is not provided to a turn left or turn right command, the robot turns 90 degrees. They are implemented by installing a new step with $heading{+}degrees$ as the heading, where *heading* is the robot's current commanded heading (that is, the most recent heading command found by searching through the pending and current ROP steps).

7.2.3 Short term plans

"Go [until you get] to the end of the hall"

"Go as far as you can"

These commands operate by installing a new step with no heading or distance commands but with `shortfront` as a condition.

"Go until you can turn {**right** | **left**}**"**

"Go until there is an opening on your {**right** | **left**}**"**

These commands operate by installing a new step with no heading or distance commands but with `longright` or `longleft` as a condition as appropriate.

"Go until you {can't | cannot} turn {right | left}"

"Go until there is no[t an] opening on your {right | left}"

"Go until there is {a wall | something} on your {right | left}"

These commands operate by installing a new step with no heading or distance commands but with `shortright` or `shortleft` as a condition as appropriate.

"Go [about] *number* **{inches | feet | yards}"**

These commands ask the robot to go the specified distance and then stop. They operate by installing a new step with no heading or condition commands but with the specified distance.

7.2.4 Long term plans

"Go to *place***"**

To execute this command, the robot first determines whether it knows the named place. If it does, and it can find a plan to get there from the current place or from its most recent known place, then the robot establishes the named place as its `*place-goal*`. If the robot is not presently at a known place, it will first return to its most recent known place using the algorithm described below under "Go back"; once it gets there, it will automatically replan a route to `*place-goal*` using the algorithm described in Section 6.5.

If the robot does not know how to get to the place, but does know where it is with respect to some other place, the robot will offer to go to the nearby known place, go in the appropriate direction, and let the user tell it when it arrives at the desired place. If the robot knows nothing about the place, it states this fact.

"Go back"

This command is implemented by stopping the robot, terminating the current step, deleting any pending steps, reversing the completed and current steps of the current ROP, and installing them as pending steps. Reversal of ROPs is described in Section 5.3. Notice that by using the existing mechanism for executing these reversed steps (the pending steps list), the whole ROP can be reversed again if the user later tells the robot to stop or to do something else while it is in the midst of going back to the most recent known place.

7.2.5 Requests for Information

"Describe *place***"**

"Tell [me] about *place***"**

At present, these commands just make the robot state whether it knows about the named place or not, and which places it is near. The robot checks to see whether *place* is known, and reports whether it is. If it is known, the robot reports the list of places that are reachable from *place* by a single ROP. It also reports the direction of the nearby place if the ROP to it contains only one step with an explicit heading.

7.2.6 Utility

"Forget where you are"

This command forces the robot to forget where it is. This is useful if the robot has become confused about its location. The command sets `*most-recent-place*`, `*current-place*`, `*next-place-goal*`, and `*place-goal*` to `nil`. This has the effect that the next time the robot is told where it is, it will not learn any new ROP to get to this place. In this way, the tutor can get the robot resynchronized with its map.

"Forget {all | everything}"

This more drastic command makes the robot erase all its state and its memory of the places in the world and the ROPs that connect them. It might be used when the robot is taken to a new environment to reset its software.

"Forget about ***place*****"**

This command is implemented by removing the place corresponding to *place* from the robot's memory, and all the ROPs that lead to and from it. This includes removing ROPs from adjacent places that tell how to get to this place. The robot will forget parts of where it is to the extent that this place is involved in that state. That is, any of the variables `*most-recent-place*`, `*current-place*`, `*next-place-goal*`, or `*place-goal*` will be set to `nil` if they are currently set to the place that is being forgotten.

A better implementation of this command might try to reconstruct ROPs that connect adjacent places, with the goal that these places remain connected in the agent's graph.

"{Load | Save} [*filename*]"

These utility commands make the system load a graph of places and their connecting ROPs from a file, or save the current graph out to a file. When loading, the robot additonally forgets its current location, as its state information might be corrupt with respect to the environment graph it has just loaded. The file save format is straightforward; it includes just the places with their associated names, prepositions, nearby-places lists, incoming and outgoing ROPs.

7.3 Questions and Responses

"Which way are you facing?"

This question is answered by reporting the robot's current heading, as converted to the nearest of the 8 principal directions, with the response "I am facing *direction*." This is useful for diagnosing problems of rotational odometry error.

“Where are you {going | headed}?”

This question is answered by reporting the robot’s `*place-goal*` with the response “I am on my way to *place*” if it has one, or “I am not on my way anywhere right now” if `*place-goal*` is `nil`.

“Where {is TJ | are you | am I}?”

This question is answered by reporting the robot’s `*current-place*`, if it is known, with the response “I am *prep place*,” where *prep* is the place’s associated preposition and *place* is the most recent name used by the user for the place. If the robot is between places, it will report that it is on its way from `*most-recent-place*` to `*next-place-goal*` with the response “I’m on my way from *place 1* to *place 2*.”

“Where is *place***?”**

This question reports information about *place* in the same way “Describe *place*” does.

“What is [to the] *direction* **of {here | you}?”**

“What are you [to the] *direction* **of?”**

These sentences are trivial to implement given the rest of our architecture, but we did not have time to implement and test them. They should report the nearest known place found by following a ROP in the appropriate direction from the current place. If the robot is between known places and the direction is consistent with the unique direction of the ROP it is following, the robot will report one of the places it is between as appropriate to answer the question.

“How {do | would} {you | I} get from *place* **to** *place***?”**

“How {do | would} {you | I} get to *place* **[from** *place***]?”**

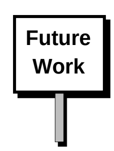
These questions ask the robot to describe routes between places or from the current place to another place.

If a path can be found between the requested places, the system describes this path in high level terms, using the intermediate place names. For example,

> To get from MARK'S OFFICE to IAN'S OFFICE,
> I go north to ANITA'S OFFICE.
> I go north to LYNNE'S OFFICE.
> I go north to ROBERT'S OFFICE.
> I go north to IAN'S OFFICE.

If the goal of the plan the system is asked to describe is a place the robot has no ROP to get to, the system uses a different strategy depending on whether the user is asking how TJ gets there or how a person would get there. If the former, TJ will explain that it can't get to places it hasn't yet experienced. If the latter, however, TJ will describe a path as above to a nearby place that is known, and then describe the direction the person should go to get to the desired place. For example,

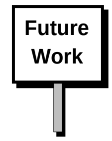
> To get from MARK'S OFFICE to THE YELLOW TRASHCAN,
> You go north to ANITA'S OFFICE.
> You go north to LYNNE'S OFFICE.
> You go north to ROBERT'S OFFICE.
> You go north to IAN'S OFFICE.
> Then you go west to THE YELLOW TRASHCAN.



This generation of language descriptions of plans could be improved in a number of respects. The repeated direction information is redundant; a multi-sentence model of discourse would help alleviate this problem. The presence of all the intermediate places in the path generation is not very useful; some post-processing of the plan could be used to simplify the description.

7.4 Modals

We have only begun to implement the described behavior of these sentences, but have not yet fully tested them.



"{**Then** | **Next** | **Finally**} *sentence***"**

Our intention is to implement them by not terminating the current step, but just appending the step described by *sentence* onto the end of the pending-steps list. This will allow the buffering of commands in the current-step and pending-steps in advance, supporting such sequences of commands as

Go to the end of the hall.
Then turn right.
Finally go about 10 feet.

This capability forms the beginnings of a capacity to support the description of full plans without the concurrent execution of them by the robot.

Chapter 8

Performance

This chapter describes the performance of our system. It provides data to support the conclusions we draw in Chapter 11.

8.1 Experimental Methodology

It is challenging to report the results of mobile robot research. A variety of approaches have been tried, from recording the path the robot took in detail by hand and on videotape [Mataric, 1992] to relying primarily on simulation [Kuipers and Byun, 1991].

In this research, we began with a reactive system that was known to be capable of basic obstacle avoidance and corridor following [Connell, 1992b]. We believe the most important aspect of the system's performance we can report is the juxtaposition of its navigation performance with its communication behavior. This is most easily appreciated in a live demonstration, but since this medium does not allow for that, videotape and descriptive dialogues will have to suffice.

We have not yet prepared a videotape of the robot in action; we expect to do this early in February, 1994. For the moment, then, we can offer only annotated dialogues which illustrate the actual behavior of the system.

In preparing these dialogues, actual transcripts of the natural language interaction were taken from the Macintosh Powerbook and pasted into this document.

These were then annotated with comments from the author concerning the robot's navigation behavior during the runs.

8.2 Experiments With TJ

We include here some examples of actual dialogue between the system and a user who was in this case the author. Text typed by the user is in a roman courier font; text said by the robot appears in italic courier.

8.2.1 Experiment 1

This transcript is taken from a successful section of a transcript from an early experiment. Parts of the transcript reflected bugs or behavior that have been fixed since this experiment was run. The part shown here was contiguous in the transcript.

`you are at mark's office`

I'm learning about MARK'S OFFICE.

`go`

I'm going.

`stop`

I'm stopping.

`you are at anita's office`

I'm learning about ANITA'S OFFICE.

`go`

I'm going.

`stop`

I'm stopping.

`you are at lynne's office`

I'm learning about LYNNE'S OFFICE.

go to mark’s office

Now I’m going from LYNNE’S OFFICE to ANITA’S OFFICE.

This is ANITA’S OFFICE.

Now I’m going from ANITA’S OFFICE to MARK’S OFFICE.

This is MARK’S OFFICE.

I have arrived at MARK’S OFFICE.

go to lynne’s office.

Now I’m going from MARK’S OFFICE to ANITA’S OFFICE.

This is ANITA’S OFFICE.

Now I’m going from ANITA’S OFFICE to LYNNE’S OFFICE.

This is LYNNE’S OFFICE.

I have arrived at LYNNE’S OFFICE.

8.2.2 Experiment 2

This reflects a more recent experiment, in which the system is stable and working consistently. The complete, unedited transcript is provided.

you are at Mark’s office.

I’m learning about MARK’S OFFICE.

you are facing south.

I am facing SOUTH. Thanks.

go

I’m going.

where are you?

I am AT MARK’S OFFICE.

stop

I’m stopping.

you are at karen’s office.

I’m learning about KAREN’S OFFICE.

where is Karen’s office?

I know how to get to KAREN’S OFFICE.

KAREN’S OFFICE is SOUTH of MARK’S OFFICE.

where is Mark’s office?

I know how to get to MARK’S OFFICE.

MARK’S OFFICE is NORTH of KAREN’S OFFICE.

where is the poster of russia?

I don’t know about THE POSTER OF RUSSIA yet.

the poster of russia is north of karen’s office.

I’m learning about THE POSTER OF RUSSIA.

where is the poster of russia?

THE POSTER OF RUSSIA is to the NORTH of KAREN’S OFFICE.

go to the poster of russia.

I don’t know how to get to THE POSTER OF RUSSIA from here.

go to mark’s office.

Now I’m going from KAREN’S OFFICE to MARK’S OFFICE.

Would you please tell me if I pass THE POSTER OF RUSSIA.

you are at the poster of russia.

I’m learning more about THE POSTER OF RUSSIA.

Now I’m going from THE POSTER OF RUSSIA to MARK’S OFFICE.

This is MARK’S OFFICE.

I have arrived at MARK’S OFFICE.

go to karen’s office.

Now I’m going from MARK’S OFFICE to KAREN’S OFFICE.

This is THE POSTER OF RUSSIA.

Now I’m going from THE POSTER OF RUSSIA to KAREN’S OFFICE.

This is KAREN'S OFFICE.

I have arrived at KAREN'S OFFICE.

which way are you facing?

I am facing roughly SOUTH.

go

I'm going.

stop

I'm stopping.

you are at roger's office.

I'm learning about ROGER'S OFFICE.

roger's office is 702.

I'm learning that ROGER'S OFFICE is also called 702.

where is roger's office?

I know how to get to 702.

702 is SOUTH of KAREN'S OFFICE.

where are you?

I am AT 702.

go

I'm going.

stop

I'm stopping.

you are at the northeast tee.

I'm learning about THE NORTHEAST TEE.

go

I'm going.

stop

I'm stopping.

go to the poster of russia.

Now I'm going from THE NORTHEAST TEE to 702.

stop

This is 702.

Now I'm going from 702 to KAREN'S OFFICE.

stop

I'm stopping.

forget where you are.

I'm forgetting where I am.

you are at roger's office.

I'm learning more about ROGER'S OFFICE.

go to the map of russia.

I don't know about THE MAP OF RUSSIA yet.

what do you know about?

I know about THE NORTHEAST TEE, 702, THE POSTER OF RUSSIA, KAREN'S OFFICE and MARK'S OFFICE.

go to the poster of russia.

Now I'm going from 702 to KAREN'S OFFICE.

This is KAREN'S OFFICE.

Now I'm going from KAREN'S OFFICE to THE POSTER OF RUSSIA.

This is THE POSTER OF RUSSIA.

I have arrived at THE POSTER OF RUSSIA.

8.3 Experiments by Other Users of the System

Michael Frank, another researcher in our lab, worked with this system to teach it about the seventh floor of our building. Unfortunately, the transcript from this experiment was lost when the portable computer containing it crashed. The experiment was very successful, however, as his report will attest.

We asked Michael to prepare a one or two paragraph summary of his experience in using the system. We advised him that we would include his report in our thesis whether it was positive or negative. This report follows.

> I used Mark Torrance's robot for a period of more than an hour. In that time, I was able to easily teach it about a large portion of the 7th floor of the MIT AI Lab. Commands such as "go as far as you can" and "go until you can turn left" provided a natural and convenient way to tell the robot how to get to new places. A number of times I tested the robot's ability to return to a place previously visited. The combination of odometric measurements with reactive wall-following proved to be very successful at achieving this task. Except for a few times when minor program bugs cropped up (and were fixed by Mark or temporarily worked around), or I accidentally pulled out the robot's cable, or the lisp environment crashed, the robot demonstrated a remarkable ability to return exactly to previously-named locations without bumping into obstacles, even when this involved going around several corners and past open doorways and around irreglar objects scattered in the halls. Even when odometric drift degraded the robot's rotational orientation, it seemed to be able to accomodate and correct for this drift through its reactive wall-following. Similarly, when the positional odometry drifted, the robot was able to continuously correct for that through the execution of reactive ROP steps such as "go as far as you can". Overall, my impression was that Mark's techniques are effective, allowing his system to break important new ground, in terms of performing a task that humans do well that no similarly-equipped robot has done before (to my knowledge), namely, reliably navigate an irregular office environment, despite the robot's having very limited and inaccurate sensors and effectors. This work points the way towards even more powerful

and flexible future mobile robot designs. I would not be surprised if commercial mobile robots of the future depend partially on some of these same techniques for navigating through offices and corridors to their assigned destinations, and for being taught about how to get to these destinations by their human co-workers.

Chapter 9

Related Work

The work described in this thesis emphasizes the construction of an integrated system that supports natural language communication as well as action in the world. Research done in several specific areas of Artificial Intelligence is related, as are some other projects that also attempt to bring together complete systems.

Related work on mobile robots falls under the topics of robot navigation, landmark recognition and self-localization. We also point to related survey articles on conventional planning. Related recent natural language work concerns active natural language processing, or *Active NLP*. Finally we describe work on other complete architectures for integrating communication with action in a robot system.

9.1 Robot Navigation

Robot navigation is an extensively studied area of AI. While there is still valuable work to be done in this area, the consensus of the community is that the problem of point-to-point indoor office navigation is to a great extent solved [Gat *et al.*, 1992].

A wide variety of methods have been employed in addressing this problem. These include potential field methods, geometric path planning, and point-to-point navigation on a graph with edges defined by reactive procedures and nodes representing landmarks.

Potential field methods were first introduced to the robotics community by

[Khatib, 1986], and have been explored further by [Arkin, 1989, Koditschek, 1987, Latombe, 1990]. These methods work by representing space as a vector field, where the vector at each point describes which way the robot should go. In typical use, there is a point in space that is the robot's goal, and a number of known obstacles. One field is introduced in which the goal a point attractor, and other fields are added to it to make each obstacle repulsive. Gradient descent is used to drive the robot toward the goal while it avoids obstacles. Local minima in the space are a problem, and these may be overcome by random motion (need a reference) or by iteratively increasing the repulsiveness of the robot's present location by introducing a new vector field (need a reference). We observe that the behavior of this latter method is similar in practice to the behavior of *who's* simulated robot experiments with the TD-lambda procedure, in which places become less desirable as they are explored.

Potential field methods are sensitive to the accurate measurement of the location of the robot and the obstacles around it. More recent methods based on realistic sensors recompute the fields as the robot runs. In these solutions, however, the problem of overcoming local minima becomes more pronounced. [Koditschek, 1987] solves these problems by computing *navigation* functions, which have a single global minimum, so that he can employ simple gradient descent. However, as noted by [Dean and Wellman, 1991], the cost of generating navigation functions can be high in complex environments.

Geometric path planning has been explored extensively for its value as a mathematical problem. Path planning in complicated configuration spaces is now well understood. Algorithms based on multiresolution cellular decompositions of free space, vertex-based shortest-path graph traversal, and smooth spline path generation have all been developed and used. Lozano-Pérez, Mason and Taylor introduce the *preimage* process by which goals are back-chained to find source positions from which the robot can move to them; they apply this process to the problem of peg insertion [Lozano-Pérez *et al.*, 1984]. They begin to explore the effects of position uncertainty in this environment; Latombe explores uncertainty in this problem in more detail [Latombe *et al.*, 1991].

Recently, very successful robot systems have been built which use combinations of simple behaviors to achieve robust navigation performance[Brooks, 1986, Brooks, 1987, Mataric, 1992]. Further research has explored the use of these systems as the basic level in multilevel control architectures, with other kinds of control at the top.

Jim Firby introduced Reactive Action Packages, or RAPs, as a way to provide conditional sequence control above primitive reactive behaviors[Firby, 1989]. RAP execution mediates between a planner which generates RAPs and the reactive behavior capabilities of the controlled system. An important emphasis of this architecture is error detection and recovery. Our work on ROPs, which also represent sequences of action, may be seen as a restriction of RAPs to the office-and-corridor navigation domain.

Erann Gat developed the ATLANTIS architecture and used it to control a variety of physical and simulated robots performing navigation tasks [Gat, 1991]. This architecture, like ours, implements planning at its highest level and uses a reactive system to control the robot during execution of the plans. He adds a third layer between these that is similar to Firby's RAP execution system, in that it is able to select an appropriate procedure for achieving a goal or performing a task from a library of procedures. ATLANTIS differs from Firby's system in that it allows the top layer of the architecture to do more than develop RAPs for the RAP library. In Gat's system, this top layer can "perform all manner of time-consuming computations, including sensor processing as well as planning."

Gat performed experiments on robot navigation in another part of the same environment we used for the experiments in this thesis. His system emphasized procedures that rely very little on odometry, since the odometry on the robot he used was poor. In our research, our success relies a great deal on the good odometry provided by Connell's robot TJ that we used.

Jonathan Connell [Connell, 1992b] developed the SSS architecture and used it to control the robot TJ, the same robot we used in this research. His architecture consists of three layers; servo, subsumption, and symbolic, which work in concert to control the robot. The robot TJ I am using implements his servo and subsumption layers, and the work I have done could fit into his architecture as a replacement for his symbolic layer.

9.2 Landmark Recognition

Landmark recognition involves recognizing known locations in the world. This process is important because it allows robots to know their locations for purposes of self-localization, and to correct for odometric error. Recent mobile robot research has explored the use of infrared, sonar, and vision for landmark recognition. Other work supports behaviorally defined landmarks. We conclude by describing how these methods can be used with odometry in a mutually reinforcing and redundant way.

Distance-measuring sensors that use infrared light or sonar can be used to capture “signatures” of particular locations. A signature is a record of the values of those distance sensors at a particular place. If the environment is stable with respect to those sensors in that place, then the signature can be used to recognize the place the next time the robot passes through it. This won’t work if, for example, there are doors that may be either open or closed that are part of the boundary of that location. Another problem can come if the robot is not always at the same lateral position when it passes through the given landmark.

Kuipers and Byun present a strategy that uses hill climbing in perceptual space to define perceptually salient landmarks [Kuipers and Byun, 1991]. Thus, a robot might navigate to a place it believes is close to its landmark, and then perform hill climbing to, for example, equalize as much as possible the values of all its distance sensors. This approach has the advantage that, exclusive of sensor noise or error, this uniquely defines each landmark. One disadvantage is the type of fine navigation required to perform the hill climbing. This work was also performed on a simulated robot; work using this strategy on a physical robot has not yet been published.

Mataric describes work in which landmarks are recognized by a high level process that observes the behavior of the robot [Mataric, 1992]. Certain behavior modules in her robot, Toto, are activated at the onset of corridor recognition and at other repeatable times. These onsets, as marked by the activation of the corresponding behavior modules, serve as landmarks. Toto is able to navigate to these landmarks by maintaining a connected graph of them in a distributed network representation, and spreading activation through this network to find the shortest route. We explored this type of landmark recognition in a simulator of Toto, but found the landmarks were not repeatable enough to serve as anchors for the odometrically guided ROPs we use to remember important places that are not perceptually distinctive.

Horswill describes a place recognition algorithm that combines vision and odometry [Horswill, 1993b]. The algorithm matches its current low-resolution image against stored images of all the landmarks the system knows about to find the best match. If its quality is high enough, the module declares it has recognized a landmark. In the presence of better odometry, we might imagine extending the algorithm to use expectations about the next landmark it will encounter to constrain the possible templates it tries to match.

Our system at present relies heavily on odometry, but can use more robust distance-based landmarks when it is instructed to do so. We originally planned to use our ROPs only in conjunction with some more reliable landmark recognition method, as the ROPs were intended for relatively local navigation. In practice, we found the translational odometry worked surprisingly well, and alleviated the need for a separate landmark recognition scheme. We believe landmark recognition techniques such as those described here could be effectively used in conjunction with the odometric techniques we have explored.

The problem of self-localization occurs when a robot does not know where it is within an environment. It is presumed that the robot knows something about the environment in advance, and that it is trying to localize itself with respect to this previously known map. The robot could be unsure of its location because its sensory expectations about the environment are not met or because it has just been turned on at an unknown location.

Kenneth Basye introduces an automata-based approach to map learning that provides a robot with the capability to infer the structure of the part of the world it can explore [Basye, 1992a, Basye, 1992b]. It does this by treating the world / robot pair as a finite automaton, using reactive strategies to robustly traverse links in the world. This procedure, or any other automatic map learning procedure, could be used to relearn a map of the environment until the map thus learned can be uniquely unified with the previously known information, thus constraining the robot's location.

Another approach to the self-localization problem is to make use of information that can be provided by a tutor or other user of the robot. We argue that robots will often operate in environments where people are available to answer questions or provide advice in cases where the robot has become confused about its location. Our system is designed to take advantage of this type of linguistic aid provided by a user.

9.3 Planning

A vast amount of research has been done on the problem of planning. We do not presume to provide a survey here; excellent overviews may be found in [Dean and Wellman, 1991, Georgeff, 1987, Hendler *et al.*, 1990].

9.4 Active NLP and Communicating Robot Systems

This section describes work on active natural language processing, which concerns systems that intimately connect the use of language to action. A few researchers have developed robot systems that learn in response to interactive natural language instructions; these are also described here.

SHAKEY the robot [Nilsson, 1984] was perhaps the paradigmatic system to connect natural language instruction to action in a mobile robot. Statements entered in English, such as

> "Use box 2 to block door DPDPCLK from room RCLK,"

were converted by the language system ENGROB [Coles, 1969] to a goal expressed as a first order predicate calculus formula:

> Blocked(DPDPCLK,RCLK,BOX2).

The planner STRIPS was then called to compute a sequence of operators that would achieve the goal. This plan was general where possible, and during plan execution it could be instantiated in different ways depending on unexpected circumstances encountered in the world.

The computational power available to mobile robots today is immense compared to that of SHAKEY. Real-time interactive conversation with a user, not practical then, is now possible on inexpensive portable computers. By emphasizing the interaction with the tutor, we have explored this new capability in a robot that solves navigation problems similar to those of SHAKEY.

Huffman and Laird present Instructo-Soar, a system which learns new procedures from sequences of instruction and also learns how to extend its knowledge of previously known procedures to new situations [Huffman *et al.*, 1993, Huffman and Laird, 1993]. Like our system, Instructo-Soar is capable of learning new procedures to accomplish a task, and then to use those procedures in a compound way under the control of a planner. Unlike our system, Instructo-Soar handles generalization, specialization, and extending knowledge about the task domain; we merely support knowledge acquisition of a particular sort within the task domain. This system is built within the Soar framework; its authors claim it is applicable to any problem representable within the Soar problem space framework. In their published work, Huffman and Laird apply Instructo-Soar to a simulated block-stacking robot problem as their example. The system has not, to our knowledge, been applied to the robot navigation domain or to any physical robot. In our research we have developed techniques that support actual robot navigation; these might be used as the basis for a performance domain theory that would allow Instructo-Soar to operate in the physical navigation domain.

Myers and Konolige describe a reasoning system that uses a combination of analogical representations and logical formalism [Myers and Konolige, 1992a, Myers and Konolige, 1992b]. The analogical representation is an annotated map of a robot's environment, which contains implicit in its structure many frame facts about the world that would be expensive to represent in logic. They also incorporate the use of a human advisor in what they call a semi-autonomous framework. In this work, Myers and Konolige recognize the potential of interaction with a human tutor for easing the perceptual burden on a robot, and explore the kinds of reasoning that can be done using their annotated map representation. To our knowledge, this system has not yet been implemented on a physical mobile robot.

Our work addresses some of the same issues. While they emphasize reasoning and problem solving ability, we have focussed on building the robot's capacity to recognize and remember places and to navigate to them on command. Some of our work on odometric place memory might be applicable as Myers and Konolige begin to implement their system on a physical robot.

Chapter 10

Future Work

This chapter describes directions in which this work might be naturally extended, and questions it has left unanswered.

10.1 Moderate Extensions

Chapter 2 mentions our plans to apply the work described in this thesis to another, vision based robot currently being developed by Ian Horswill. Linguistic communication with robots that have additional high-bandwidth human communication modalities available such as vision will provide interesting opportunities, as we may interleave Ian's work on gesture recognition with this work on natural language interaction.

The same chapter also mentions the exciting potential of state-of-the-art speech recognition systems; finding a reliable system that can run on a portable computer and interfacing it to a robot running our navigation system would be a good implementation project.

Chapter 6 leaves open the question of how to appropriately execute a step that contains both a distance and an explicit termination condition. Presumably the distance was measured when the ROP was first recorded, and so the distance should be similar the next time the ROP is executed. If it is slightly different, this might be chalked up to sensor or odometry error and so ignored. If the distance is dramatically different, the robot might presume it has encountered some unusual type of error, such as failing to see the sensory condition because of a change in the world or a bad sensor. Fortunately, since we record the steps of the current ROP reversibly, if the robot detects a serious error it has a recovery strategy that involves returning to the last place where it knew where it was.

There are several simple extensions to the current system described in Chapter 7. Keeping statistics on the number of times users use each of the multiple names for a place would help the system pick a well-used name from the assortment it might store. A simple user model could store preferences for place names on a per-user basis. This model could also record places the robot knows the user knows about, which would allow the robot to give routes to unknown places in terms of places the user knows, or at least to make descriptions of user-known places less elaborate than those of user-unknown places.

We do not yet uniformly support the commands that let the user state that two places are in fact the same. If the robot already knows about these two places from separate experiences, and there are no conflicts between the stored ROPs for the separately learned places, it should be able to merge them on this command. This, too, is problem left open for future work.

Likewise, the command for forgetting about a place does not leave behind a usable graph. It would be better for this command to also create ROPs to connect places on either side of the forgotten place, so that the graph would remain connected.

The hybrid approach to natural language processing we describe in Chapter 3 would be a natural step to take as soon as the complexity of the language desired in the system warrants it. Input text would be passed first to a parser, which might not recognize the text at all if it contains words that aren't in the lexicon. If it is not recognized, the pattern matcher could make an attempt, and if it is still not recognized, only then would the system report that it did not understand.

We did not complete the implementation of the description of nearby places; the answers to questions such as "What is to the north of Mark's office?" This implementation is straightforward given the rest of our architecture. As we described earlier, because the robot stores its ROPs in terms of directions in world coordinates, it can determine the answers to these questions directly from these ROPs.

The implementation of the modal sentences that begin with "then", "next", or "finally" should be completed and tested. We believe these sentences provide a solid basis for understanding and storing described plans without simultaneously executing them, but we can't yet know for sure.

Language generation in the system could be improved a lot. This is especially true for the multi-sentence descriptions of plans.

10.2 Bigger Questions

This section covers some bigger questions that may require a significant amount of further research to answer.

10.2.1 Error Recovery

We have left open the question of error recovery. Many anomalous situations can be resolved with the help of the tutor or user, who may notice that the robot has become confused and can help to correct it through language. In practice, we would hope the robot would not need to be attended by a user all of the time. Under these circumstances error detection and recovery strategies are called for.

Error detection requires knowledge of the robot's plans and enough sensory capacity to tell that something is interfering with them. People or other obstacles might block a corridor that was once free, for example. The robot should recognize that it is not making forward progress and initiate an error recovery strategy.

The fact that ROPs can be reversed makes for a good error recovery strategy if nothing better can be found. It is likely that the robot can successfully reacquire its most recent known place if it tries. This should work, based on our experience, even when the robot has been interrupted by an obstacle during the execution of a plan. The robot's translational odometry will not increase when the robot is forced to stop by an obstacle in its path. This condition could be used to detect this type of error; the fact that it stays the same means the ROP reversal should be successful as well.

10.2.2 Integration with Landmark-Based Navigation

It was our original goal to begin with a landmark-based navigating robot, treat those landmarks as highly reliable, and use our primarily odometry-based ROPs only for recording path offsets from the landmarks. We still believe this is a good approach, especially as landmark recognition techniques improve (see Section 9.2). This integration task poses some definite challenges in interleaving planning and navigating in the existing navigation system with ROP execution once the nearest landmark has been acquired.

10.2.3 Places with Extent

How could this system handle places with extent? So far, our system treats all place descriptions as referring to single points. There are some descriptions in plans people use, however, which refer to places that definitely have extent, such as “the east corridor” or “the playroom”. We would like a system that could understand that some places have extent. It should learn their extent by being told directly, or by being told at multiple different locations that they are all part of the same place.

We built a simple natural language system, which we called LOCO, that was able to learn about places with extent. This system was not connected to a physical robot, and its model of the world included no error in position sensing or effectors; it was primarily an exercise to explore the kinds of language we wanted to include in this thesis research. LOCO represented the world as a cellular grid, and assumed that places were defined by the minimal aligned rectangular area that encloses all the cells which LOCO was told were part of the place. LOCO could represent places that had overlapping extent and find an appropriate description of any location in terms of the places that contain it.

In implementing places with extent in the physical robot system, there are a few problems. One arises each time the robot is told that it is at a place it thought it already knew about. Should it assume the place has extent, or that it had experienced odometry error and that it is really now again at the same spot it had learned previously? The answer probably depends on the amount of the error the robot can detect between its present location and the place it had previously learned. If this error is small, it might assume it had become slightly off and that the place has no extent. If the error is large, perhaps it should assume that the place has extent. In intermediate cases the best thing to do might be to ask the user for clarification.

10.2.4 ROP Improvement

How can plans be relearned or improved over time? What level of control should the user have in language over when the robot replaces an old ROP with a new one? How can the system reason about space to optimize a ROP, or to correctly unify more ROPs? How can the system fold knowledge about multiple ways to reach a place into each other, and how can it use this knowledge to help invent better paths to other places that are along the way? We recognize these interesting problems as important areas for future research, but have not worked on them to date.

Chapter 11

Conclusions

This chapter describes our findings after performing this research. We apply our evaluation criteria as laid out in Section 1.6.

The solution we have developed exhibits the performance described in Section 1.1. In particular, it is able to associate names provided by a tutor with places in the environment based on direct or indirect descriptions. It is able to use those names in responding appropriately to navigation requests or user queries.

Another user has worked with the system for more than an hour. His experiences are described in Section 8.3.

Appendix A

Simulation

This chapter describes the use of simulators in research on mobile robots. It draws from material previously published in [Torrance, 1992].

Simulators used in research on autonomous mobile robots have been criticized for their tendency to change the nature of the problems the robot control architecture has to solve. In this chapter we address those arguments, and find that under certain conditions simulators can be a valuable tool to supplement research with physical robots. We conclude with guidelines for the successful design and use of simulators in research on mobile robots.

A.1 Introduction

There is much active work in the field of robot control architectures for mobile robots. Some have chosen to supplement or replace work on physical robots with research using simulators–software programs designed to model the interaction of a robot with its environment.

In some cases simulators are motivated strongly by the physical characteristics of a particular robot and environment. More often, simulators idealize and abstract certain parts of the problem.

The use of simulators as a substitute for experiments with physical robots has been roundly criticized [Brooks, 1987, Brooks, 1991b]. In this chapter, we consider these criticisms, as well as positive reasons for using simulators. We go on to discuss some existing simulators and the ways in which they have been used. We conclude with some principles for the appropriate design and use of simulators in mobile robot research.

A.2 Simulators Can Be Misused

Mobile robot simulators provide many features convenient for research, some of which will be discussed below. Despite their advantages simulators have been criticized for making it easy to solve some difficult problems, making it difficult to solve other easy problems, and for creating false decompositions of robot control problems. We shall consider these criticisms in turn.

In [Brooks, 1987], Brooks argues that simulation "requires a constant feedback from real experiments to ensure that it is not being abused." He goes on to say that simulators create a temptation to simulate the perceptual system, creating false decompositions which lead researchers to work on problems they claim will be integratable into a general framework. In [Brooks, 1991a], Brooks adds that cellular representations of space are problematic in simulations of physical robots.

We agree that real robots and environments must be used during the design of simulators which attempt to model them. It is plausible, however, that once a simulator has been demonstrated to model a certain robot and domain in many respects, that simulator could then be used for extended periods without reference to the physical robot. It is important that research which claims to be applicable to physical robots be tested on such robots. However, the advantages of working in simulation, discussed below, may make a substantial reliance on simulators a worthwhile alternative.

As for the criticism that simulators tempt researchers to create false decompositions into which their research will fit, this is a problem not just with simulators but with work in all fields on "part of a complete AI system," including work on vision, robotics, planning, reasoning, and learning. Our sense, however, is that scientists in these fields are coming to understand better and better the need to make their claims accurate and not overgeneral. In some cases, such as Brooks' work, this leads them to design different decompositions of the problem. In other cases, such as work on planning, this leads researchers to back off from their claims of generality to offer their work instead as solutions to more restricted problems.

Brooks' argument that cellular representations of space are unsatisfactory is compelling. Certain computational domains derive much of their interest from just the interactions and constraints that make cellular representations unsuitable for mobile robot simulators. The use of these representations in such domains is fine, but some have tried to extend the use of cellular spaces to robot control problems, particularly in the field of Distributed AI (see, for example[Durfee and Montgomery, 1989]). We agree with Brooks that such simulations are unlikely to model physical robots well enough to be useful research tools, and we recommend continuous space simulation.

We feel that it is important to credit scientists with the responsibility to use their tools wisely. A simulator is a powerful tool that clearly has the potential for misuse, but with increased familiarity, we believe researchers will come to understand the new responsibilities that come with the use of a simulator. In fact, we argue below that one of the advantages of simulators is that they create increased accountability by encouraging more independent verification and subsequent reuse of results. Note also that the use of simulators comes with much higher responsibility for clear separation between reporting of theoretical results, experimental results, and conjecture.

Another criticism of simulators is that it is very hard to write a good one. If we understood the domain well enough to build an accurate simulator, the argument goes, we could use that understanding directly to write programs to control robots in that domain.

This may actually be a good reason to build simulators, not to avoid them. If we can gain some understanding of the domain by working hard to simulate it accurately, that understanding can be valuable when it comes time to write control programs. This answer does not address the challenge of making good models. We argue that good programmers should be able to build good models, and that more important than simulating the world exactly is understanding the limitations of the simulation.

A.3 Simulators Can Be Useful

If the responsibilities mentioned above are heeded, simulators can be a powerful tool for the development of robot control architectures. Their use should increase the verification and incremental progress that will make work in this field science rather than engineering. Simulators will create an inexpensive, accessible development environment, and can be useful for experiments with new robot configurations before the robots are built. For all of these reasons, simulators can play an ever more important role in mobile robot research.

Scientists using simulators can share code, robot designs, and error models with one another. Research results can be more easily verified by other researchers with access to the same simulator software. Results can be reported in the literature with reference to the shared simulated environment with which other researchers are familiar. In this way, a well-designed simulator may provide some “benchmark” tests for mobile robot control architectures, useful for comparisons between different architectures.

One criticism we have of current work on the design of autonomous mobile robot architectures is the large quantity of completely novel architectures which are proposed each year. In most other sciences, great value is placed both on repeating experiments to verify the results of others, and on building incrementally on the results of others. One feature of work in mobile robots which may contribute to this problem is that results with physical robots are very expensive and difficult to reproduce. Another feature is that the standards for reporting are often not high enough for readers of an article to actually implement or perform the experiments suggested.

We believe that widespread use of simulators to supplement work with physical robots will help the field by making more experiments repeatable, facilitating dialogue about the details of the experiment and the assumptions it makes. It will also enable researchers to build on the implemented ideas of their peers directly, actually reusing simulated robots, domains, and control architectures.

Simulators provide an inexpensive development environment. This can be especially important for laboratories in which access to physical mobile robots is a very limited resource. Furthermore, hardware has lots of problems which it is perfectly reasonable to simulate away. These include physical connector problems, short battery life, and the difficulty of reconfiguring the sensors and effectors on a physical mobile robot.

Following this line, simulators may also be useful for experimenting with proposed designs for robots to see which are useful for certain tasks. An example is placement of sonars so a robot can effectively pass through a doorway. If the environment, mobile base, and sonars were well modelled, a simulator would be a good testbed for trying different robot configurations.

We have seen that simulators have the potential to be valuable as research tools. Let's go on to look at some existing simulators, and see the extent to which each meets the goals just outlined.

A.4 Existing Simulators

We discuss here a few simulators for mobile robots. This list is not meant to be complete or even representative. Rather, it suggests the character of some of what has been done with simulators in mobile robot research.

A.4.1 Realistic Simulators

The simulators described below have in common the goal of realistic simulation of particular physical mobile robots and their environment.

Erann Gat [Gat, 1991] implemented and used a simulator for some experiments as part of his Ph.D. thesis on his ATLANTIS architecture. In this case, the simulator modelled an existing physical robot. Gat provides a good discussion of the appropriate use of simulators, and follows his own advice and our recommendations in the clarity and quality of his reports on experiments performed with the simulator.

Jonas Karlsson and Patrick Teo have implemented a simulator called Botworld [Nilsson, 1991] for Nils Nilsson at Stanford University to demonstrate his work on Action Networks and Teleo-Reactive Sequences. This program uses a client-server model, allowing robot control programs to be written in other languages and run on different machines. The Botworld simulator is a good model of some aspects of a frictionless 2-d navigation and manipulation problem being studied with physical robots in the Aeronautics and Astronautics department at Stanford, but a simplistic model of other aspects. In this case, the physical robot system is so carefully engineered that a simulation can accurately model many aspects of the problem. So far, this work has not addressed the issue of the effectiveness of these strategies on low-level navigation of physical robots.

EDDIE is a testbed simulator written at Carnegie Mellon University for experiments on outdoor road-following and navigation. This simulator provides primitive sensor functions which approximate the mid-level output from vision and laser range-finding sensors. It has been advocated by its authors as a general purpose testbed which can be used for other work on mobile robots. While it is well-designed for its purpose, at least one researcher (Lynne Parker at MIT) has had difficulty adapting it to work in a different domain (multiple communicating mobile robots in an indoor office environment). This may teach us something about what is needed for a simulator to be truly reusable and general.

Lynn Stein has done some work on a program, MetaToto [Stein, 1991], which models to a very rough approximation the environment of the robot Toto built by Maja Mataric [Mataric, 1992]. In her paper, Stein maintains that when coupled with a working robot system, rough simulation can help MetaToto to build an approximate landmark map which can be useful while the robot learns about a new physical topology. The simulation in MetaToto is not intended to accurately model the physical domain of Toto, or to be used in the initial design and testing of a mobile robot architecture. It is interesting, however, in that it tests the hypothesis that simulators can be designed well enough that the same robot control program runs on both the simulator and the physical robot being simulated.

Early in the development of this Master's research, we planned to extend the work of Maja Mataric [Mataric, 1992, Mataric, 1990] on Toto to handle higher-level linguistically described cognitive features. Toward this end, we built a simulator for the sonar-based robot Toto in its indoor office environment. we succeeded in making the simulator correct enough that the same robot control programs will drive both the simulator and the robot. The problem is made easier by the fact that the drive system, a holonomic Real World Interfaces B-12 base, is well designed. Sonar modelling appears to be the hardest aspect of this simulator.

A.4.2 Idealized Simulators

Some simulators are not meant to simulate a real physical world at all, but merely test the kinds of decision making and problem solving that robots might need to go through. In some cases, these simulators abstract parts of the navigation problem by assuming a cellular space.

Rich Sutton's work on the DYNA learning architecture [Sutton, 1991a, Sutton, 1991b] uses a simulation which models some aspects of a dynamically changing environment with spatial locality. As a simulator, this program suffers from the criticism that it may allow a false decomposition of the problem of intelligent action by allowing Sutton to focus on a reasoning problem in isolation. We argue that the jury is still out on the question of how this problem of intelligent action may be decomposed. For this reason, it is still worthwhile to pursue strategies for isolated subproblems such as reasoning and perception. In this particular case, Sutton is very clear about the limits of his work. Mark Drummond and Martha Pollack have a simulator called Tileworld which, though different in some details, shares the dynamic and unpredictable characteristics of Sutton's simulator.

The video game domains Pengo and Amazon used by Phil Agre and David

Chapman in their work on Pengi [Agre and Chapman, 1987] and Sonja [Agre, 1988] are another example of a less than realistic simulator. Agre argues that the domain shares many of the problems of situated action found in the physical world, while finessing certain hard problems such as vision. The perception this research assumes is not as farfetched as a "magic recognition box," but it is well beyond the capabilities of current artificial vision systems.

MICE is an experimental testbed offered by Durfee and Montgomery for distributed artificial intelligence research [Durfee and Montgomery, 1989]. It contains many features which make it attractive for experiments with multiple communicating robots with heterogenous skills. It uses multiple asynchronous processes to simulate each robot, so the simulation of multiple robots may potentially start many processes. Its cellular representation of space may make it unsuitable for detailed physical simulation. Its separation of the simulation from the implementation of the agent control programs is a positive contribution.

These idealized simulators are less interesting from the perspective of mobile robot research, as they make no attempt to accurately model the problems of navigation physical robots encounter. Still, it is important to evaluate them with respect to their author's claims about the realistic nature of the problems they model.

So far no simulator, realistic or otherwise, is a competitive substitute for experiments on physical robots. What would it take to build such a simulator?

A.5 Principles for Design

We conclude by presenting a set of principles which may be followed in the design and use of mobile robot simulators. The powerful simulator we call for here has its origins in the discussion of better simulators as a future direction Erann Gat's dissertation [Gat, 1991]. Our hope is that by making some features of this proposed simulator concrete, we may encourage others to join our efforts to develop such a simulator.

First and foremost, a simulator must come with a clear sense of the assumptions made about the domain it models, and of the capabilities and limits of the simulation. Claims about the generality, scalability and usefulness of results demonstrated on a simulator must always be made with respect to the relationship of the simulator to the real world.

Another important characteristic of an effective simulator is modularity and extensibility. As Gat describes, an ideal simulator will "allow the user to construct customized robots by mixing and matching sensors and actuators which are actually software objects." Models of error used in these devices should also be customizable by the user. These components should be described in an easy to understand language, either declarative, procedural, or some combination, so that they can be shared between different implementations of the simulator running on different platforms.

The simulator should provide models of the types of sensors commonly used on mobile robots, including sonars, pyroelectric and infrared sensors, microphones and photocells, all preprocessed in various ways. For effectors, holonomic bases such as the RWI B-12 base and non-holonomic bases such as four and six-wheeled cars, treaded vehicles such as bulldozers and tanks, and legged robots should all be offered. In addition to "canned" sensors and actuators, submodules should also be provided at a variety of levels to facilitate implementation of new sensors and actuators. It should be possible to configure simulated robots in any reasonable way, including the mounting of sensors on articulated parts such as arms or actuated pivots.

The user should have as many choices as possible, both at design time—when constructing the robot and the environment—and at run time, in the characteristics and performance of the simulation. The simulator should provide both precise algorithms and faster approximation algorithms, and allow the user to choose between them to trade off precision against speed in the simulation. The user should be able to control the granularity of the simulation in both space and time. Continuous motion simulation should be used wherever possible, as unintended effects from discrete simulation have been documented [Brooks, 1991a].

Pseudorandom sequences used to generate data for simulated sensors should be repeatable exactly. If the simulator has control over the robot control architecture, and if that architecture is deterministic, this will allow a repeatability not possible in the physical world that can reveal elusive bugs in the control programs or in the simulator itself.

Surfaces and obstacles in the simulated environment should be able to have different response characteristics with respect to each sensor, to model things like different specular reflective response to sound or light. Different surfaces should be able to have different responses to the same sensor, to model the differences between walls of different colors, textured objects, and so on.

An interactive graphical interface is desirable for ease of use, development, and debugging user robot control programs, new robot models, and other new simulator modules. The simulator should provide for multiple possibly heterogenous robots, including "drones" or moving obstacles, controlled by the user at run time or by simple programs. Different robots should be able to be driven by different robot control architectures, potentially even implemented in different programming languages. A client-server model or other distributed systems approach may be helpful here. This can foster separation of the simulation from the robot control programs, as well as potentially distributing the computational load over several computers.

Finally, in using the simulator the scientific community should work toward generally accepted "toolkit parts," including error models, environments, sensors and effectors, parts of robots and whole robots which the community agrees are reasonable models of actual physical robots in particular domains.

A.6 Conclusion

Simulators can be a powerful tool for the development and testing of robot control architectures. We hope they will continue grow in number, power, flexibility and acceptance as the field working on autonomous mobile robots comes to understand their strengths and to use them wisely.

Appendix B

Text of Human-Robot Dialogue Study

This chapter contains the text of the questionnaire from a small study we performed on just two subjects, and the dialogues they wrote in response. Analysis of the results of this study are in Section 3.3.

B.1 Talking to Loco

Mark Torrance is working on a Master's Thesis which involves building a restricted Natural Language system for communication between a person and a robot about their shared physical environment as described by the person. To help me incorporate a useful subset of English, I'm asking you to contribute some sentences and / or dialogue fragments which you might expect to be supported by a robot that claims to speak your language.

Scenario: You have just purchased LOCO, a robot which can understand some English. You can tell LOCO about places in your office building by describing them or by showing them to the robot. Then LOCO can follow your directions, answer questions about where it is or where it's been, or about the spatial relationships of places you've taught it.

Please write some sentences and / or short dialogue fragments which illustrate the kind of interactions you would expect to have with LOCO. Precede sentences said by the user with "User:" or "U:", those said by LOCO with "Loco:" or "L:", and add any other comments or clarification on subsequent lines or in parentheses. If possible, put sentences you consider easy or basic first, and more challenging sentences later.

If you do this on paper, feel free to use more paper if you choose, and please return it to Mark Torrance, NE43-705, 545 Tech Square, Cambridge, MA 02139. If you do this online, please send your response in email to `torrance@ai.mit.edu` with the word `dialogue` in the subject of your message. Thank you very much for your participation!

1. Initial training–teaching LOCO about your environment.

2. Questions and answers about relationships among places LOCO has been taught (these may include things you didn't write down in 1.)

3. Examples of instructions you might give LOCO.

4. Questions and answers about where LOCO is and / or where it has been.

B.2 Dialogue by Michael Frank

This dialogue was written by Michael Frank on November 18, 1992.

B.2.1 Initial Training

Initial tranining–teaching Loco about your environment

(Loco is delivered to 1st floor lobby)

U: Hello, Loco. I am Michael.

L: Nice to meet you.

U: Come with me so I an show you around.

L: Okay.

(U walks into an elevator. Loco follows. It stops on the 4th floor. They get out.)

U: This is the 4th floor. All our offices are on this floor.

L: Good. I can't use elevators very well.

(U Opens door, enters hallway, walks down hallway. Loco follows.)

U: These are the FSF offices.

L: The doors on the right?

U: Yes.

(U walks a while longer. Stops at an office door.)

U: This is my office.

L: Okay.

(U steps down the hall a bit further.)

U: This is Bill's office.

L: Where exactly?

U: That door in the corner.

L: Okay. I'm unused to doors in corners.

(U walks on)

U: This is Jon Doyle's office.

L: Okay.

U: This is the MEDG Lounge.

L: The large open area on the left?

U: Yes.

(U walks through a door)

U: This is the main MEDG office. To your right is Scott's office. Annette's desk is in front of you. Peter Szolovits's office is over here. (walks toward it) The machine room is here. (stands in front of it)

L: Okay.

U: Let's go back outside

(L Leaves MEDG office, user following. They continue.)

U: This is Steve and Jennifer's office... This is Ira and Yeona's office... and this is

Carl and Kevin's office.

L: Okay.

U: That's it for now. Stand by in the lounge until needed.

L: Okay. Thank you for training me.

B.2.2 Relationships Among Places

Questions and responses about relationships among places Loco has been taught.

U: Hello. Can you tell me where Jon Doyle's office is?

L: Follow me.

(L leads U to the office.)

L: Here it is.

U: Thanks. How about Jennifer Wu's office?

L: It's across the lounge from here.

B.2.3 Statements of Goals

Statements of goals for Loco to execute.

U: Loco, take this letter to Annette and ask her to mail it.

(U puts letter on tray carried by Loco)

...

U: Loco, can you go around the MEDG offices and see who's in? Report back to me.

B.2.4 Questions About Location

Questions and responses about where Loco is and / or where it has been.

U: Loco, where have you been?

L: I was exploring the other end of the 4th floor.

U: Oh. Did you give Dave that donut?

L: Yes. He was in Carl's office.

...

U: Hello. What is this place?

L: This is the MEDG lounge. Can I help you?

B.3 Dialogue by Christopher Lefelhocz

This dialogue was written by Christopher Lefelhocz on November 1, 1992.

Anyway, here's something that I thought might help which may or may not be what you want. If you are giving the robot natural language then I would think that Joe average would give it directions on how to get places so I think that's where I'll start.

So...

Let's assume the Robot is names Jake and Joe is the "programmer". Starting in the Elevator Lobby...

Joe: We are in the Elevator Lobby.

Jake: Okay.

Joe: We go north and east.

Jake: Is there a door?

Joe: Yes, there is a door with a no smoking sign next to it.

Jake: Okay.

Joe: Going through the doorway we go down the hall to where it splits.

Jake: Okay, are we at the split?

Joe: No, (they walk), now we are.

Jake: Okay.

Joe: We go left and go past a bunch of doors.

Jake: Define bunch.

Joe: More than one.

Joe: Wait, we've now reached a "hole" in the wall.

Jake: What's this "hole" in the wall?

Joe: It's not a room, but there is more space than the size of the hallway.

Jake: What's here?

Joe: A computer, and also, Mark's office is off to the right?

Jake: Who's Mark and right of what is the office?

Joe: Mark is a friend of mine and to the right of the hallway is his office.

Jake: Could you be more specific?

Joe: Mark's office is the door that is 1 2 doors from the split in the hallway.

Jake: Okay.

Joe: We go down the rest of the hallway to the playroom.

Jake: This is the Playroom?

Joe: Yes.

Jake: What's here?

Joe: Posters, framed pictures, and some furniture.

Jake: How big is this area?

Joe: Bigger than the Hall, and Bigger than an office.

Jake: How big is an office?

Joe: 8X10 Feet.

Jake: Okay.

Joe: At the end of the playroom is another hallway.

Jake: Okay.

Joe: We can go down the hallway till we hit a wall.

Jake: How do we know we've hit a wall.

Joe: We can keep walking straight, because the hallway splits.

Jake: Is this the same split as before?

Joe: No this one is has a door to the left, and the hallway continues to the right.

Jake: What's behind the door?

Joe: The Elevator Lobby.

Jake: You mean the same elevator lobby where we started?

Joe: Yeah, the same one.

Jake: So we have gone in a circle?

Joe: Yes.

Jake: Okay.

Joe: Let's down the rest of the hallway.

Jake: Okay, tell me what's in this hallway.

Joe: More doors, we go right and then we have to go left.

Jake: Wait, what do you mean?

Joe: Oh sorry, we go right at the split then the hallway corners and we go left.

Jake: What do you mean by corner?

Joe: (annoyed) A corner to a hallway is where the hallway turns and doesn't split, into two hallways.

Jake: So a corner is a split with only one direction to go in the hallway?

Joe: Yes.

Jake: Okay.

Joe: We've just come on another split.

Jake: Which directions can we go?

Joe: We can go to the left, and straght.

Jake: So this is like the split after the elevator, but we are coming up on it from the continuing hallway, not the ending hallway.

Joe: Yes (impressed by the Jake's ability).

Jake: Where should we go?

Joe: To the left.

Jake: Okay.

Joe: We are now going down a wider hallway.

Jake: What's here?

Joe: A door to the right that says "stairs" on it.

Jake: What are stairs?

Joe: Don't worry about it, you don't have legs?

Jake: What are legs?

Joe: Forget it, it was a joke.

Jake: What's a joke?

Joe: Never mind.

Jake: Okay.

Joe: So we keep walking and reach another split.

Jake: Are we continuing along the hallway?

Joe: No, we going to go left.

Jake: Does our hallway end?

Joe: Yes.

Jake: Okay.

Joe: We can now walk past a bunch of doors to another split that has an exit.

Jake: Does this split continue?

Joe: Yes, but you might notice that we've been here before.

Jake: Where are we?

Joe: The split continues on to Mark's office, and goes to the left to the elevator lobby.

Jake: Oh, now I know where we are.

Joe: Good.

Well there you go, sorry for the humor, but I think this might or might not help. I assumed some type of interface where the machine asks questions since it needs more pertainent information. I'm trying to give you a flavor for what I would expect. It took me about 25 mins. to type in the commands and think about what was going on. The length of the sentences is probably about as long as you're going to get given that people don't like to type a lot.

Bibliography

[Agre and Chapman, 1987] P. Agre and D. Chapman. Pengi: An implementation of a theory of activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 268–272, Seattle, Washington, July 1987. Morgan Kaufmann Publishers, Inc.

[Agre, 1988] Philip E. Agre. The dynamic structure of everyday life. Technical Report 1085, Massachusetts Institute of Technology Artificial Intelligence Laboratory, October 1988.

[Arkin, 1989] Ronald C. Arkin. Motor schema-based mobile robot navigation. *The International Journal of Robotics Research*, 8(4):92 – 112, August 1989.

[Basye, 1992a] Kenneth Basye. An automata-based approach to robotic map learning. In Gat et al. [1992], pages 1–4.

[Basye, 1992b] Kenneth J. Basye. *A Framework for Map Construction*. PhD thesis, Brown University Department of Computer Science, Providence, Rhode Island, 1992.

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

[Brooks, 1987] Rodney A. Brooks. Planning is just a way of avoiding figuring out what to do next. MIT AI Lab Working Paper, September 1987.

[Brooks, 1991a] Rodney A. Brooks. Artificial life and real robots. In *Proceedings of European Conference on Artificial Life*, December 1991.

[Brooks, 1991b] Rodney A. Brooks. Intelligence without reason. MIT AI Lab Memo 1293, Massachusetts Institute of Technology Artificial Intelligence Laboratory, Cambridge, Massachusetts, April 1991.

[Brown *et al.*, 1992] Russel G. Brown, L. Paul Chew, and Bruce R. Donald. Map-making and localization for mobile robots using shape metrics. In Gat et al. [1992], pages 10–17.

[Chrisman, 1992] Lonnie Chrisman. Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 183–188, San Jose, California, July 1992. MIT Press.

[Coles, 1969] L. Stephen Coles. Talking with a robot in english. In Donald E. Walker and Lewis M. Norton, editors, *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 587–596, Washington, D.C., 1969.

[Connell, 1992a] Jonathan H. Connell. Extending the navigation metaphor to other domains. In Gat et al. [1992], pages 29–35.

[Connell, 1992b] Jonathan H. Connell. SSS: A hybrid architecture applied to robot navigation. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, pages 2719–2724, Nice, France, May 1992.

[Dean and Wellman, 1991] T. Dean and M. P. Wellman. *Planning and Control*. Morgan Kauffman, 1991.

[Durfee and Montgomery, 1989] Edmund H. Durfee and Thomas A. Montgomery. Using MICE to study intelligent dynamic coordination. In *Proceedings of the 1989 Distributed AI Workshop*, September 1989.

[Firby, 1989] R. James Firby. Adaptive execution in complex dynamic worlds. Technical Report YALEU / CSD / RR-672, Yale University, 1989.

[Gat *et al.*, 1992] Erann Gat, Marc Slack, Steve Chien, and Avi Kak, editors. *Working Notes: AAAI Fall Symposium Series: Applications of Artificial Intelligence to Real-World Autonomous Mobile Robots*, Cambridge, Massachusetts, October 1992. American Association for Artificial Intelligence.

[Gat, 1991] Erann Gat. *Reliable Goal-Directed Reactive Control of Autonomous Mobile Robots*. PhD thesis, Virginia Polytechnic Institute and State University, 1991.

[Georgeff, 1987] Michael P. Georgeff. Planning. In J.F. Traub, editor, *Annual Review of Computer Science, Volume 2*. Annual Review Inc, 1987.

[Hendler *et al.*, 1990] James Hendler, Austin Tate, and Mark Drummond. AI planning: Systems and techniques. *AI Magazine*, 11(2):61–77, 1990.

[Horswill, 1993a] Ian Horswill. Polly: A vision-based artificial agent. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 824–829, Washington, D.C., July 1993. MIT Press.

[Horswill, 1993b] Ian Horswill. *Specialization of Perceptual Processes*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, May 1993.

[Huffman and Laird, 1993] Scott B. Huffman and John E. Laird. Learning procedures from interactive natural language instructions. In *Proceedings of the Tenth International Workshop on Machine Learning*, pages 143–150, Los Altos, CA, 1993. Morgan Kaufmann.

[Huffman *et al.*, 1993] Scott B. Huffman, Craig S. Miller, and John E. Laird. Learning from instruction: A knowledge-level capability within a unified theory of cognition. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, pages 114–119. Lawrence Erlbaum Associates, 1993.

[Khatib, 1986] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5:90–99, 1986.

[Koditschek, 1987] D. Koditschek. Exact robot navigation by means of potential functions: Some topological considerations. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1–6, 1987.

[Kuipers and Byun, 1991] Benjamin Kuipers and Yung-Tai Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Robotics and Autonomous Systems*, 8:47–63, 1991.

[Latombe *et al.*, 1991] Jean-Claude Latombe, Anthony Lazanas, and Shashank Shekhar. Robot motion planning with uncertainty in control and sensing. *Artificial Intelligence*, 52(1):1–47, 1991.

[Latombe, 1990] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1990.

[Lozano-Pérez *et al.*, 1984] Tomás Lozano-Pérez, Matthew T. Mason, and R. H. Taylor. Automatic synthesis of fine-motion strategies for robots. *The International Journal of Robotics Research*, 3(1):3–24, 1984.

[Martin, 1993] Charles Martin. The situated language project. Presented at the Massachusetts Institute of Technology Artificial Intelligence Laboratory Revolving Seminar, December 1993.

[Mataric, 1990] Maja J. Mataric. A distributed model for mobile robot environment-learning and navigation. Master's thesis, Massachusetts Institute of Technology Artificial Intelligence Laboratory, May 1990.

[Mataric, 1992] Maja J. Mataric. Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation*, 8(3):304–312, June 1992. Journal article on Toto.

[Myers and Konolige, 1992a] Karen L. Myers and Kurt Konolige. Reasoning with analogical representations. In Bernhard Nebel, Charles Rich, and William Swartout, editors, *Proceedings of the Third International Conference on Principles* *of Knowledge Representation and Reasoning*, pages 189–200, Cambridge, Massachusetts, October 1992. Morgan Kaufmann Publishers, Inc.

[Myers and Konolige, 1992b] Karen L. Myers and Kurt Konolige. Semi-autonomous map-making and navigation. In Gat et al. [1992], pages 129–133.

[Nilsson, 1984] Nils J. Nilsson. Shakey the robot. Technical Note 323, SRI International, Menlo Park, CA, April 1984.

[Nilsson, 1991] Nils J. Nilsson. Botworld. Personal Communication, 1991.

[Stein, 1991] Lynn Andrea Stein. Imagination and situated cognition. AI Lab Memo 1277, Massachusetts Institute of Technology Artificial Intelligence Laboratory, February 1991.

[Sutton, 1991a] Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. In *Working Notes of the AAAI Spring Symposium on Integrated Intelligent Architectures*, March 1991.

[Sutton, 1991b] Richard S. Sutton. Reinforcement learning architectures for animats. In *Proceedings of the International Workshop on the Simulation of Adaptive Behavior: From Animals to Animats*, pages 288–296. GTE Laboratories, Incorporated, Waltham, MA 02254, MIT Press, 1991.

[Terveen, 1993] Loren Terveen. Human-computer collaboration: Reconciling theory, synthesizing practice, papers from the 1993 fall symposium series. Technical Report FS-93-05, AAAI, 1993.

[Torrance, 1992] Mark C. Torrance. The case for a realistic mobile robot simulator. In Gat et al. [1992], pages 181–184.

[Torrance, 1994] Mark C. Torrance. Two case studies in active language use. In Charles Martin, editor, *Working Notes: AAAI Spring Symposium Series: Active NLP: Natural Language Understanding in Integrated Systems*, Stanford, California, March 1994. American Association for Artificial Intelligence. Forthcoming.

[Whitehead and Ballard, 1991] Steven D. Whitehead and Dana H. Ballard. Learning to perceive and act by trial and error. In *Machine Learning 7*, pages 45–83, 1991.