Technical Report 1228

A Distributed Model for Mobile Robot Environment-Learning and Navigation

Maja J. Mataric

MIT Artificial Intelligence Laboratory

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AI-TR 1228	2. GOVT ACCESSION NO. AD-A225711	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Distributed Model for Mobile Robot Environment-Learning and Navigation		5. TYPE OF REPORT & PERIOD COVERED technical report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Maja J. Mataric		8. CONTRACT OR GRANT NUMBER(s) #SI-804475-D N00014-86-K-0685 N00014-85-K-0124
9. PERFORMING ORGANIZATION NAME AND ADDRESS Artificial Intelligence Laboratory 545 Technology Square Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Advanced Research Projects Agency 1400 Wilson Blvd. Arlington, VA 22209		12. REPORT DATE May 1990 13. NUMBER OF PAGES 129
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Office of Naval Research Information Systems Arlington, VA 22217		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution is unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

path planning	distributed representation
spatial learning	mobile robot navigation
emergent behavior	parallel algorithms

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis presents a method for robust mobile robot navigation, large space learning, and path planning, based on a totally distributed architecture. The described methods were implemented and tested on a physical robot. The robot, Toto, consists of an omnidirectional base supplied with a ring of twelve ultrasonic ranging sensors and a compass. It is fully autonomous with

(continued on back)

Block 20 continued:

all power and processing onboard. All experimental data were gathered in unaltered office environments with static and dynamic obstacles.

Toto is an example of incremental design methodology. The robot was programmed in the Behavior Language, based on the subsumption architecture. Its behavior consists of three real-time, reactive layers of competence: collision-free boundary tracing, landmark detection, and environment learning and path planning.

Low-level navigation consists of a collection of simple reflex-like rules which, when acting in parallel, result in an emergent boundary-tracing behavior. This behavior is used by the landmark detector which dynamically extracts features from the environment using the way the robot is moving as it is moving. The landmarks are used to construct a distributed map of the environment. The map is represented as a graph of landmarks. The links in the graph are used to indicate topological adjacency, and are assigned dynamically. The structure of the environment is used to bound the outdegree of the graph nodes resulting in linear graph connectivity.

The graph is distributed in that, like biological neurons, the nodes are concurrently acting behaviors: all receive sensor and landmark inputs and communicate by sending messages to their nearest neighbors. Using the parallel distributed implementation of the graph the robot can localize in constant time regardless of the size of the graph.

An adaptation of *spreading of activation* is used for path finding and optimization. It is equivalent to parallel graph search which computes both the topological and physical shortest path in time linear in the size of the graph. A simple algorithm for local motion decisions is introduced which utilizes a greedy strategy. The robot uses only local information to execute a globally optimal path to the goal. The need for replanning is minimized.

The main issues discussed in the thesis are: distributed v. global representation, qualitative v. quantitative computation qualitative v. quantitative representation, procedural v. declarative representation, design of emergent behaviors, dynamic v. static landmark matching, minimizing and simplifying communication.

A Distributed Model for Mobile Robot Environment-Learning and Navigation

by

Maja J Mataric

Massachusetts Institute of Technology

A revised version of the thesis submitted in partial fulfillment of the requirements of the degree of Master of Science in Electrical Engineering and Computer Science

Abstract

This thesis presents a method for robust mobile robot navigation, large space learning, and path planning, based on a totally distributed architecture. The described methods were implemented and tested on a physical robot. The robot, Toto, consists of an omnidirectional base supplied with a ring of twelve ultrasonic ranging sensors and a compass. It is fully autonomous with all power and processing onboard. All experimental data were gathered in unaltered office environments with static and dynamic obstacles.

Toto is an example of incremental design methodology. The robot was programmed in the Behavior Language, based on the subsumption architecture. Its behavior consists of three real-time, reactive layers of competence: collision-free boundary tracing, landmark detection, and environment learning and path planning.

Low-level navigation consists of a collection of simple reflex-like rules which, when acting in parallel, result in an emergent boundary-tracing behavior. This behavior is used by the landmark detector which dynamically extracts features from the environment using the way the robot is moving as it is moving. The landmarks are used to construct a distributed map of the environment. The map is represented as a graph of landmarks. The links in the graph are used to indicate topological adjacency, and are assigned dynamically. The structure of the environment is used to bound the outdegree of the graph nodes resulting in linear graph connectivity.

The graph is distributed in that, like biological neurons, the nodes are concurrently acting behaviors: all receive sensor and landmark inputs and communicate by sending messages to their nearest neighbors. Using the parallel distributed implementation of the graph the robot can localize in constant time regardless of the size of the graph.

An adaptation of *spreading of activation* is used for path finding and optimization. It is equivalent to parallel graph search which computes both the topological and physical shortest path in time linear in the size of the graph. A simple algorithm for local motion decisions is introduced which utilizes a greedy strategy. The robot uses only local information to execute a globally optimal path to the goal. The need for replanning is minimized.

The main issues discussed in the thesis are: distributed v. global representation, qualitative v. quantitative computation qualitative v. quantitative representation, procedural v. declarative representation, design of emergent

behaviors, dynamic v. static landmark matching, minimizing and simplifying communication.

Thesis supervisor: Dr. Rodney A. Brooks

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for this research was provided by Hughes Artificial Intelligence Center contract #SI-804475-D, the Office of Naval Research contract N00014-86-K-0685, and the Defense Advanced Research Projects Agency under Office of Naval Research contract N00014-85-K-0124.

Acknowledgements

The MIT AI Lab provided a very inspirational, helpful, and supportive environment throughout the entire thesis conception, implementation, and documentation process. Some individuals deserve special mention. I am very grateful to Rod Brooks for numerous helpful ideas, suggestions, and encouragement he provided when the task ahead of me appeared too monumental for my chisel. I am also very grateful to Anita Flynn for being an outstanding role model with enthusiastic advice ranging from tips on hardware debugging (yes, it's always a connector problem) to life philosophy (ok, let's change the world!). Thanks to Lee Tavrow for inspiring brainstorming and help with data gathering (in spite of the inadvertently incurred janitorial wrath). Much gratitude to Mike Erdmann for being a supportive officemate, a wonderful friend, and a source of great discussion and advice. Thanks to Peter Ning for great patience with my numerous hardware questions. I am grateful to Pattie Maes for our inspiring conversations. Many thanks to her and Anita Flynn for their extensive helpful comments on earlier drafts of this thesis. Thanks to the whole mobot group for late night company and support.

I am grateful to Eric Grimson and Rod Brooks, as well as to Jon Connell and Lee Tavrow for extensive comments on the thesis which helped in revising it for the Tech Report. Thanks also to Bruce Donald for many useful suggestions.

Special thanks to Rich Roberts for continuing support and calming influence throughout the years. Finally, endless gratitude to Mira Mataric for invaluable advice, friendship, wisdom, and promising genes.

Contents

1 Introduction **10**
- 1.1 The Challenge . . . 10
- 1.2 The Response . . . 11
- 1.3 Summary and Outline . . . 16

2 A Review of Related Work **18**
- 2.1 HILARE . . . 19
- 2.2 Elfes . . . 20
- 2.3 Moravec . . . 21
- 2.4 Drumheller . . . 22
- 2.5 Crowley . . . 22
- 2.6 Kuipers . . . 23
- 2.7 Payton . . . 24
- 2.8 Arkin . . . 24
- 2.9 Connell . . . 25
- 2.10 Summary . . . 25

3 Philosophy and Motivation **27**
- 3.1 The Subsumption Approach . . . 27
- 3.2 Choosing a Functional Representation . . . 30

4 The Robot Toto **32**
- 4.1 The Simulation Alternative . . . 32
- 4.2 Toto . . . 34
  - 4.2.1 Sensors . . . 34
  - 4.2.2 Sonar Hardware and Software Drivers . . . 34
  - 4.2.3 The Power Supply . . . 36

4.2.4 The Central Processor 36
4.3 The Programming Environment 38
4.4 Summary 39

5 Navigation **40**
5.1 Motivation 40
5.2 Sensor Characterization 40
5.3 The Navigation Rules 43
5.4 Emergent Properties 52
5.5 Summary 55

6 Landmark Detection **56**
6.1 About Landmarks 56
6.2 Dynamic Versus Static Landmark Matching 57
6.3 The Dynamic Landmark-Matching Algorithm 58
6.4 Performance 61

7 Environment Learning **64**
7.1 The Goals 64
7.2 Spatial Learning Through Graph Construction 67
7.2.1 Using Expectation 70
7.2.2 Using Position Estimation 72
7.3 Summary 77

8 Goal-Oriented Navigation **78**
8.1 Finding the Shortest Topological Path 80
8.2 Finding the Shortest Physical Path 81
8.3 Summary 82

9 Choosing the Right Network Topology **83**
9.1 A Review of Related Network Topologies 83
9.1.1 Network Theory 89
9.2 The Return to Linearity 91

10 The Dynamic Graph Topology **93**
10.1 The Graph Structure 93
10.2 The Switchboard Algorithm 96
10.3 Direction Preservation in Path Planning 98

10.4 An Example of the Switchboard Performance 101
10.5 Summary 102

11 Related Biology **104**
11.1 The Topological Versus Metric Dilemma in Biology 104
11.2 Bats, Bees, Birds, and Rats 106
11.2.1 Bats 106
11.2.2 Bees 106
11.2.3 Birds 107
11.2.4 Rats 107
11.3 Models of Spatial Memory 108
11.3.1 Hierarchical Models 108
11.3.2 Deutsch's Topological Model 109
11.3.3 Zipser's Model 109
11.3.4 Global Versus Distributed Neural Models 110
11.4 Summary 111

12 Conclusion and Future Work **112**
12.1 Directions for Future Work 112
12.1.1 Fine Tuning the Representation 112
12.1.2 Testing Generality 113
12.1.3 Adding Reasoning 113
12.1.4 Optimizations 113
12.1.5 Achieving True Parallelism 114
12.2 The Contribution 114

List of Figures

1.1 The Global Overview 12

3.1 Horizontal Task Decomposition 27
3.2 Vertical Task Decomposition 28

4.1 Toto 33
4.2 Computational Hardware 37
4.3 Toto's Three Competence Layers 38

5.1 Sonar Signatures 41
5.2 Sonar Consistency 42
5.3 Distance Radii 43
5.4 Avoiding Performance 47
5.5 Aligning Performance 48
5.6 Correcting Performance 50
5.7 Incremental Emergence 51
5.8 Implicit Arbitration 51
5.9 Room Boundary Tracing 53
5.10 Corridor Following 54

6.1 Room Landmarks 62
6.2 Corridor Landmarks 63

7.1 Node I/O 65
7.2 Network Communication 66
7.3 Test Environment 68
7.4 Linear Learning Trace 69
7.5 Linear Graph 69
7.6 Blocked Test Area 71

7.7 Using Expectation 71
7.8 A Rough Position Estimate 73
7.9 An Ambiguous Scenario 74
7.10 A Confusing Environment 74
7.11 An Incorrect Trace 75
7.12 Misleading Context Information 75
7.13 A Correct Trace Using Position 76
7.14 The Resulting Correct Graph 76

8.1 Turn Required 79
8.2 No Turn Necessary 79

9.1 A Difficult Environment 84
9.2 An Incorrect Qualitative Mapping 85
9.3 A Better Qualitative Grid 86
9.4 A Wasteful Grid 87
9.5 2.5-D Grid 88
9.6 Butterfly 89
9.7 Tree Mesh 90
9.8 The Dynamic Graph Topology 92

10.1 The Switchboard 94
10.2 Making a Jumper 95
10.3 Starting a New Path 97
10.4 The Resulting Jumper 97
10.5 Goal-Call Direction Arbitration 99
10.6 Goal-Call Propagation 100
10.7 A Path-Optimization Test Environment 101
10.8 The Topological Map 102
10.9 The Execution of Path Optimization 103

Chapter 1

Introduction

1.1 The Challenge

The work in this thesis was motivated by a classical problem in mobile robotics: goal-directed navigation. The research presented here provides an approach to the problem from a perspective different from that which is well entrenched in the philosophy of the field.

The MIT AI Lab Mobile Robot group has adopted a philosophy which was once considered radical. It was inspired by Brooks' introduction of the *subsumption architecture* as a method of building layered control systems for mobile robots. The method embodies a set of fundamental principles about the way the problem of robot control is handled. These principles address the issues of reactivity, real-time response, onboard processing, computational complexity, state maintenance, world modeling, and planning.

The subsumption approach presents an alternative to the classical planning paradigm in the way it decomposes the problem. Successful subsumption programs are effective combinations of heuristics and adjustments based on empirical data, rather than applications of a formalized method. Consequently, the subsumption approach has gained respect gradually, based on empirical support. It was, and continues to be, defended, tested, and debugged through the process of building physical robots.

The robots built in the group have demonstrated novel solutions to obstacle avoidance, wall following, object tracking **[Viola 90]** and object following **[Horswill and Brooks 88]**, room recognition and door-finding **[Sarachik 89]**, six-legged locomotion **[Angle 89]** **[Brooks 89]**, etc. The robots were

successfully endowed with abilities to perform a variety of apparently complex tasks such as soda can collection [**Connell 89**], and people and sound following [**Flynn, Brooks, Wells, and Barrett 89**].

The robots implemented by the group tested the feasibility of the subsumption method. Still, the question that remains asked by the critics is "How far will it scale?" Are there some fundamental limitations of the approach? What is the class of systems which can be implemented with the subsumption approach? What kinds of systems are outside that class?

Goal-directed navigation is commonly divided into a group of classical problems: obstacle avoidance, local navigation, and global path planning. Path planning is an example of what is considered to be a classical planning problem, which is possibly not solvable with a distributed, reactive approach such as that of the subsumption architecture.

Previous to this work, subsumption-based robots relied on random walk, directed by the stimuli in the environment. They wandered until their sensors alerted them to a specific condition which triggered task-related response behavior. The natural next step was to explore possibilities for deterministic navigation.

Goal-directed navigation necessarily involves the use of a world representation, which opens up a philosophical can of worms. Joined with the representation issue was the need to solve the path planning problem without the conventional notion of a path, or the use of a conventional planner.

1.2 The Response

This thesis describes a mobile robot which has the task of exploring the environment, learning its structure, storing it into an appropriate representation, and using that representation to find and follow shortest paths to arbitrary known locations.

The system was implemented on a physical robot which uses sonar sensors and a compass, and was designed to work in unaltered office environments with static and dynamic obstacles. All data shown in the thesis were obtained in real runs of the robot in such an environment. The implementation of the robot software and hardware was a process of mutual constraint satisfaction: the high-level task (goal-oriented navigation) imposed top-down constraints and influenced the choice of hardware and software for the robot. At the

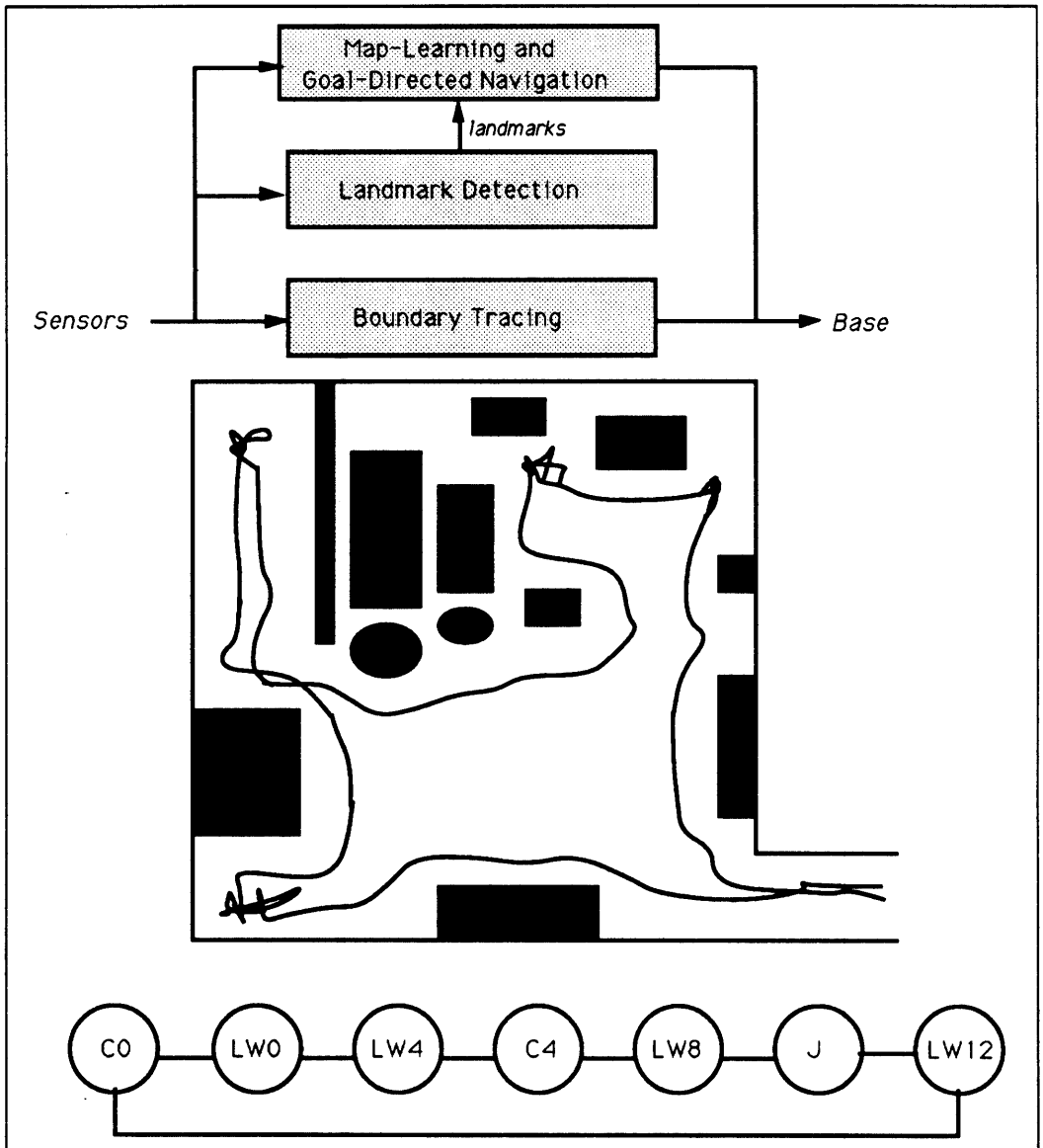


Figure 1.1: The three layers of the robot, a typical performance of the boundary tracing algorithm, and an example graph representing the explored environment.

same time, the bottom-up constraints imposed by the physical hardware, sensors, and actuators (described in chapter 4), influenced the design of the software and the approach taken in defining the entire task. Designing both bottom-up and top-down resulted in a fully integrated system in which the function of each part is well motivated. This helped simplify the debugging of the system, as well as increased its robustness and reliability. Although designed for a specific physical system, the methodology employed in the thesis generalizes to other sensor types and mobile robot architectures.

The work described in this thesis combines the often-segregated problems in goal-oriented navigation into a single system. The robot performs obstacle avoidance, local navigation, and path planning, but not through centralized control of separate modules. Instead, the system performs the task as a result of the combination of many simple, concurrently acting behaviors.

The robot, Toto, either explores the environment and builds and confirms its environmental map, or purses a goal. The system is designed as a hierarchy of competence layers. In the lowest layer, simple reflex-like rules combine into emergent collision-free navigation behavior with the property of tracing boundaries of objects. The middle layer uses the motion of the robot while tracing boundaries to dynamically extract landmarks in the environment. The top layer uses those landmarks to construct a distributed map of the world and use it to find paths. Figure 1.1 illustrates Toto's three layers of competence and chapter 3 describes the underlying philosophy and motivation.

Each of the competence layers was implemented with a novel approach in order to introduce and study alternative methods to solving the problems on all levels of robot control.

In the implementation of the low-level navigation layer the concept of emergent behavior was used for of developing a robust, collision-free object-tracing performance. The goal was to design a collection of intuitive, reflex-like rules which, when combined, result in the desired emergent behavior. A simple but sufficiently functional sensor characterization was developed in a form of a guiding heuristic for constructing navigation rules. The rules were designed to be as simple as possible. They are triggered by mutually exclusive environmental conditions so as to completely circumvent the need for explicit arbitration. They were added to the system incrementally, thus keeping its preformance and analysis tractable. The middle diagram in figure 1.1 shows a real run demonstrating the object tracing behavior and chapter 5 describes

it in detail.

The landmark detection layer uses a dynamic method for procedurally extracting features from the environment from the way the robot is moving while it is moving. Instead of selecting landmarks as sonar signatures corresponding to particular locations in the world, they were defined as large, permanent structures such as walls and corridors. These landmark types were are selected because they could by robustly detected with the available sensors. The advantage of this approach is in not having to rely on sensor precision and repeatability, or position control.

Landmark detection corresponds to continuous adjustment of confidence levels correlated with environmental features. A sufficiently high confidence level for a feature denotes a landmark. The landmarks are defined qualitatively, and their representation is implicit in that it results from the procedure a robot executes rather than form a static, declarative model. Another important advantage of dynamic landmarks is their generality. Like the low-level navigation behaviors, the landmark detection behavior generalizes to any sensor system providing proximity information, and is independent of its exact physical properties or configuration. Finally, the landmark detection layer utilizes the layer below it thus minimizing the amount of added reasoning. Chapter 6 describes this layer in detail.

Using purely qualitative descriptions of locations and a sparse landmark set results in ambiguities, so landmark descriptors are augmented with compass bearings and an estimated size. The latter is derived from using the notion of time as distance, which gives an implicit representation of time in the system. Assuming constant velocity of the robot, the compass bearing is integrated to provide a coarse cartesian position. This value is used for rough position comparisons in map localization. Details of landmark disambiguation are given in chapter 8.

The third layer of competence uses the landmarks provided by the layer below to construct a topological representation of the environment. A variety of graph topologies was explored (see chapter 9) before choosing the simplest yet powerful topology - a linear list. The list is augmented with dynamic links resulting in an undirected acyclic graph capable of embedding any 2D physically feasible topology (see chapter 10). The choice of topology is important since the graph structure is fixed at compile time. To avoid implementing a full crossbar between the graph nodes, a static switchboard mechanism is employed to simulate dynamic links. A method is presented

for a direct embedding of any world topology in the simple augmented list topology.

An analysis of the structure of the office environment, combined with the boundary tracing navigation behavior, yielded a simplifying definition of the topology of space (see chapter 10). This property allowed for bounding the outdegree of the nodes in the graph to an empirically determined constant. The resulting was a graph with linear connectivity.

The distributed nature of the spatial representation presented in this thesis comes from the implementation of the graph as a collection of concurrently acting behaviors. Each behavior corresponds to a unique landmark in the world. It receives inputs from the landmark detector, as well as from the sensors, and can communicate by sending and receiving messages from its neighbors in the graph. The robot's location in the world is indicated by a single active graph node corresponding to that location. The active node performs lateral inhibition by spreading deactivation. It also spreads expectation to its neighbor in the direction of travel. Expectation is a method of preserving minimal context to be used for graph verification and landmark disambiguation (see chapter 7 for details).

Localization within the graph consists of comparing the broadcast landmark to all of the graph nodes. The use of a parallel implementation allows for localization in constant time regardless of the size of the graph.

The process of environment learning consists of storing the landmarks in the graph for future use in path planning. The process of constructing a simple, linear graph, its use and its limitations are discussed in chapter 7. An augmented, more powerful graph representation is presented in chapter 10. An example of such a graph is shown in the third diagram in figure 1.1.

One of the challenges of the distributed approach was the implementation of path planning within a decentralized map, devoid of a global view of the relationship between the start and goal. The solution was implemented with a variation of *spreading of activation* **[Quillian 69]** approach. The goal node sends a *call* to its neighbors, which propagate it on through the graph. When a call reaches a node, its direction specifies the direction in which the robot should travel next from that landmark. Regardless of where the robot is located, it knows the optimal direction to pursue toward the goal (see chapters 8 and 10 for details). This eliminates the need for replanning if the robot strays from the desired path or becomes lost. The boundary-tracing algorithm simplifies the motion decision to a binary choice in most cases, at

times augmented with the compass bearing for decision points with a higher fanout (see chapter 10). Consequently, the edges in the graph need not carry any extra information. More importantly, there is no separate reasoning engine outside the graph; all spatial reasoning used for learning and path finding is in the graph itself.

Finding shortest paths consists of a parallel graph search. Finding the shortest topological path is equivalent to a search in a graph with edges of unit weight. This runs in worst case linear time in the size of the graph. Augmenting the nodes with an estimated length of each landmark weighs the edges properly in order to compute the physically shortest path in worst case linear time.

1.3 Summary and Outline

The main issues addressed in this thesis include:

- distributed versus global representation,
- qualitative versus quantitative computation
- qualitative versus quantitative representation,
- procedural versus declarative representation,
- design of emergent behaviors,
- dynamic versus static landmark matching,
- minimizing and simplifying communication.

The organization of the thesis proceeds chronologically through the research process, addressing each of the relevant issues as they are encountered.

Chapter 2 reviews related work in robot planning and goal-directed navigation. It describes the classes of existing approaches and gives examples of each.

Chapter 3 discusses the motivation behind the ideas implemented in this thesis, as well as the guiding philosophy.

Chapter 4 gives a detailed description of the physical robot which was used to test all the ideas.

Chapter 5 describes the navigation algorithms. Chapter 6 presents the landmark detection scheme. Both contain data from many test runs. Chapter 7 explains the learning algorithms, and the initial, simple graph structure. Chapter 8 gives the goal-oriented navigation scheme within the simple graph structure, and its performance.

Chapter 9 describes the limitations of the simple graph representations, and gives an overview of possible alternatives. Chapter 10 describes the improved, more general graph representation, and its performance.

Chapter 11 is an overview of related biological systems and neurophysiological data. It speculates possible correlations with biology, and presents a number of questions for future investigations.

Chapter 12 reviews the main results of the thesis, and suggests areas for future research.

Chapter 2

A Review of Related Work

The purpose of goal-oriented navigation is to enable a robot to reach some previously visited or otherwise known point in the world. If the goal is within the range of the robot's sensors, the task is usually trivial. Otherwise, the robot must know its position relative to the goal, in order to select its next action. Consequently, it needs a representation of the world.

In the classical literature, the task of goal-oriented navigation is usually presented as a path planning problem. The system is provided with a map of the environment, which it uses in conjunction with its sensors, to reach a goal location. Unless the robot's collision avoidance implementation involves a reactive scheme, all motions of the robot are preplanned.

Classical path planning has been used in both robot manipulators and mobile robots. The methods can be divided into two basic groups: local and global. Local methods, such as potential fields **[Khatib 86]**, are most commonly used for fine motion planning employed in manipulator control. Local methods compute a function using the parameters obtained from the external and proprioceptive sensors on the robot. The result is a direction vector for the robot's next move.

In contrast, global methods are based on searching through free space. These methods usually employ a cartesian world map containing information about the known obstacles. The map can be used to explicitly compute free space areas, such as through the use of configuration space obstacle growth **[Lozano-Pérez 81]**. Given some representation of free space, all classical planning algorithms are variations of search in that space. They rely on a representation of the world which, for each resolution point in the map,

determines if the robot is inside or outside an obstacle. The map is divided into regions based on this information, and a search for a path through the regions is performed. Such algorithms include Voronoi diagrams **[Canny and Donald 87]**, visibility graphs **[Lozano-Pérez and Wesley 79]**, quad trees **[Faverjon 84]**, and generalized cones **[Brooks 83]**.

Given that generating a path is usually reduced to search, the most difficult problem in goal-oriented navigation is the choice of representation that will optimize the search process. The following section gives a partial overview of some approaches to world modeling and path planning.

2.1 HILARE

The work of Raja Chatila, Georges Giralt, Marc Vaisset, and Jean-Paul Laumond on HILARE is an excellent example of a functioning application of the classical mobile robot control approach. Acknowledging that the environment of mobile robots is complex, they stress the importance of constructing and maintaining an accurate environment model and localizing within it **[Giralt, Chatila, and Vaisset 83]**.

HILARE is equipped with a number of different sensors: 14 ultrasonic range sensors, a camera with a laser range-finder, and an infrared beacon-based triangulation system. Recognizing the difficulty in maintaining different types of information within a single world model, their approach involves three levels of representation **[Chatila and Laumond 85]**.

- On the first level, the perceptual data are translated into a two-dimensional geometric world model employing polygonal approximations of the perceived obstacles. Gaussian error estimation is used in model matching.
- On the second level, the geometric model is used to deduce the topological properties of the environment. Space structuring is performed by constructing convex polygons corresponding to related obstacle edges and vertices from the geometric model. These polygons are called "cells" and their adjacency relationships reflect the topology of the explored space.
- On the third level, a semantic model can be constructed by object labeling in the topological graph. The performance of this level was not explored.

The advantage of the multi-level world model is that it offers different types of information which can be accessed by different parts of the robot's control system and checked for consistency. The disadvantage of the approach is the increased complexity of updating the models, as well as the decreased fault-tolerance.

Chatila and Laumond recognize the position problem as the key issue in world model maintenance. They suggest three methods for maximizing the accuracy of this process: absolute position referencing using known beacons, trajectory integration using odometry or inertial guidance, and relative position referencing using local features in the environment. Calibration is performed repeatedly in order to minimize cumulative position error and maintain the necessary accuracy of the models. Updating and correcting the world models is performed with a set of uncertainty functions applied to the sensor data.

2.2 Elfes

Another example of multiple levels of world modeling is offered by **[Elfes 86]**. The levels are viewed as separate mapping dimensions along the abstraction axis, the geographical axis, and the resolution axis.

- The abstraction axis maps the transition from data-intensive representations to higher levels of abstraction. This process is illustrated in the transition from the initial cartesian map to a geometrical representation. Sonar data from the sensor level map are used at the geometric level to group occupied cells into unique objects to be approximated polygonally. Finally, at the symbolic level, topological information is extracted from the geometric data, and represented explicitly.

- The geographical axis starts with views (locally visible areas), which are integrated into local maps, which, in turn, combine to form a global map.

- The resolution axis abstracts the data to a decreasing level of detail in order to speed up its processing. It allows for "zooming" in and out of regions of interest.

The work provides a clear formalism for grouping tasks involving world model construction and maintenance. The actual implementation, however, presupposes precise position and orientation data and claims that cumulative error only results in a topological distortion of the produced map. It claims that perfect position control is necessary and can be obtained through dead-reckoning combined with a stereo matcher for motion estimation.

2.3 Moravec

Moravec employed a simple yet functional representation of space in his path planning algorithm used on the Stanford Cart [**Moravec 83**]. The vision system on the robot modeled objects as clouds of features which were approximated as uncertainty ellipsoids, and eventually simplified to spheres. The spheres were projected to circles in a two-dimensional representation. A path consisted of a series of tangent segments between the circles. To generate the shortest path to the goal, the circles served as graph nodes in the path planning search.

In more recent work, Moravec addressed the specific problem of dealing with uncertainty in world model construction and maintenance. [**Moravec and Elfes 85**] introduced a grid-mapping method for representing the environment based on range sensor data. The entries in each grid cell correspond to the confidence in the cell's occupancy based on multiple sonar readings, as well as single transducer readings gathered from different locations over time. [**Moravec and Cho 89**] describes a less ad hoc analysis of the same approach using probabilistic occupancy maps based on Bayesian statistical analysis.

The certainty grid based cell occupancy representation is convenient because of its independence of the sensors used. This feature also makes it a good representation choice for sensor fusion into a single world model [**Moravec 88**]. The approach offers a simple but consistent formalism of uncertainty for world model maintenance. Assuming the path planner has some method for using the certainty levels, it can be viewed as a basic cartesian representation of free space.

2.4 Drumheller

Drumheller addressed the specific problem of map localization based on matching actual sonar data to an a priori world model [**Drumheller 87**]. His approach used an analytical characterization of the sonar sensor in order to estimate measurement error. After error correction and smoothing, he performed line fitting in order to obtain a line-segment representation of the robot's current position. The line segments were matched to a stored model library through the construction of an *interpretation tree* [**Grimson and Lozano-Pérez 87**].The search space of the matcher was pruned with the use of local geometric constraints.

This work shows that localization is feasible based on noisy sonar data, given a reliable world model and a sophisticated matcher. The disadvantage of the combinatorial blowup could be partially remedied if other sensors are used to further prune the search space (for instance compass data). The method does however rely on a static environment and was not tested on a physical robot.

2.5 Crowley

An approach similar to Drumheller's is described in [**Crowley 85**]. He proposes a method for updating a world representation by integrating local information into a global model as the robot is executing a path to the goal. The global model is previously learned or provided, while the local information is obtained with a rotating sonar and a touch sensor. Sonar data are recursively line-fitted to form line segments which are matched to a model base. The matcher updates the global model by subtracting the computed average error in the local position estimate from the global model.

As with Drumheller's approach, this method depends on a static environment and an efficient matching algorithm. An accurate position estimate is assumed based on wheel shaft encoders. Any errors in this estimate must be handled by the model matcher. The method was not tested on a physical robot.

2.6 Kuipers

[**Kuipers 79**] describes the TOUR model as a method of simulating some aspects of the human cognitive map. (The concept of a cognitive map is described in more detail in chapter 11 of this thesis). Kuipers' model of the spatial environment is constructed over time based on a number of egocentric inputs from the sensors. The model consists of sensory inputs called *views* and motions which cause a state change, called *actions*. The cognitive map is constructed from the sensorimotor input, which is modeled as a sequence of alternating views and actions.

Following a previously known route in the TOUR model corresponds to executing a *procedural map*. Such a map is a production-like schema consisting of a description of the goal, the current situation, the action to perform next, and the result to expect.

The model also uses a topological map. The map has two levels, one providing a topological network of places and paths between them, the other supplying the boundary and containment relations of places and paths.

A quantitative description of the environment can be introduced in the model, through adding metric components to each action, such as distance to traverse, and angle to turn.

The topological map ideas of the TOUR model were implemented in a simulation described in [**Kuipers and Byun 88**] and [**Kuipers 87**]. A qualitative graph representation of the environment is used in which nodes in the graph are locally distinctive features based on geometric criteria. The nodes are connected by directives such as "travel" and "turn." Traveling to a location in the graph consists of searching to localize the starting node, and following the arcs to the destination, with repeated hill-climbing whenever necessary.

The described topological model is appealing in its possible relation to the notion of the cognitive map. Additionally, it presents a simple method of representing the environment. The main disadvantage lies in the assumptions made in the simulation. The sensory system is modeled as a point-source range sensor with unrealistic characteristics. It would be interesting to test the method on a real system and evaluate its performance.

2.7 Payton

One common disadvantage of the path-generating methods described previously is that replanning is necessary if the robot strays from the planned path. Clearly, it is desirable that the robot know the proper direction to pursue regardless of its position within the map. **[Payton 88]** suggests an approach in which plans are represented as *action resources.* In his method, the world is represented with a cartesian grid of cells, and a goal is selected a priori. A full breadth-first search is performed from every possible start position to the goal based on a cost function utilizing the cell distance to the goal. The search generates the cost for each grid cell. Finding a path from any cell to the goal, then, is equivalent to gradient descent along the path of lowest cost.

The method was successfully tested on the Autonomous Land Vehicle. Its flexibility relies on the robot's ability to localize itself accurately. The limiting factor is the selection of a static goal; whenever the goal changes, the entire search must be repeated. In that respect the method has the same disadvantages as planning approaches.

2.8 Arkin

The work described in **[Arkin 87]** is an excellent example of a hybrid approach combining global and local classical path planning techniques. The system is divided into three hierarchical levels: the planner, navigator, and pilot. The mission planner interprets high-level commands and sets the criteria for the mission. The middle level, the navigator, performs the classical path planning task of producing a meadow-map-based free space representation and finding a piece-wise linear path within it. The path is based on an a priori map. The pilot is responsible for executing each segment of the path.

The behavior of the robot is determined by the interaction of *perceptual and motor schemas*, which are special-purpose computational elements. The pilot selects a set of appropriate schemas to be executed in a parallel, distributed manner. Each schema uses the appropriate input from the sensor and the world model to generate a generalized potential field. Arbitration between concurrent schemas is resolved through vector addition.

[**Arkin 89**] offers a detailed justification of such a hybrid approach to robot control. He cites neurophysiological evidence supporting this view in biological systems.

2.9 Connell

[**Connell 89**] introduced an entirely distributed, behavior-based control system for a mobile robot. Herbert navigates in office environments, finds and picks up soda cans, and returns them to a home location. The work stressed the issue of minimal representation and limited use of state. It also forced decentralization through the use of 24 loosely connected processors rather than a central processing unit.

Herbert does not construct any type of spatial representation, nor keep a history of the traversed path. In finding the way home, he employs a simple navigation scheme based on global orientation referencing. The robot is capable of recognizing doorways, which it uses as landmarks. Doorways are assumed to be the only choice points on the robot's path. Herbert relies on a simple heuristic: if on the way home, go south at each doorway. This scheme has an appealing simplicity, but depends on a convenient positioning of the home location, as well as a simple layout of the environment. In spite of its limitations, the approach is the first attempt at implementing global navigation in a fully distributed, reactive system.

2.10 Summary

There are many other relevant works, including [**Braunegg 90**], [**Stewart 88**], [**Durrant-Whyte and Leonard 89**], etc. This review selected a few representative examples of the types of approaches being explored. An overview of the field yields leads to the conclusion that the majority of goal-directed navigation methods are based on search through some representation of free space. The selection, construction, and update of this representation is the crux of the problem.

One of the goals of the work presented in this thesis is to bypass representing free space explicitly. As an alternative, the method to be presented deduces the structure of the environment from the robot's motion. It relies on the heuristic that motion must by definition be through free space. The

environment is described as a series of contiguous landmarks which are stored in a topological, distributed graph.

All actions of the robot are designed as real-time concurrent behaviors. The method is reactive but performs a classical planning task. The results of this research demonstrate that a hybrid approach which combines low-level reactivity with a high-level classical planner, may not be the only effective solution to goal-directed navigation.

Chapter 3

Philosophy and Motivation

3.1 The Subsumption Approach

The classical planning approach imposes what can be viewed as a horizontal or sequential decomposition of the task into processing modules [**Brooks 86**]. The typical sequence is as follows: a snapshot of the world is taken by the sensors; the sensor data are converted into a format understandable by the planner; a plan is generated; finally, the plan is translated into actuator commands. Figure 3.1 illustrates such a task decomposition. This organization is appealing from the point of view of the software designer as it appears natural and modular. It is often viewed as the "divide and conquer" approach to the problem. However, it suffers from several flaws resulting from its inherently sequential nature. Since the input of each module in the

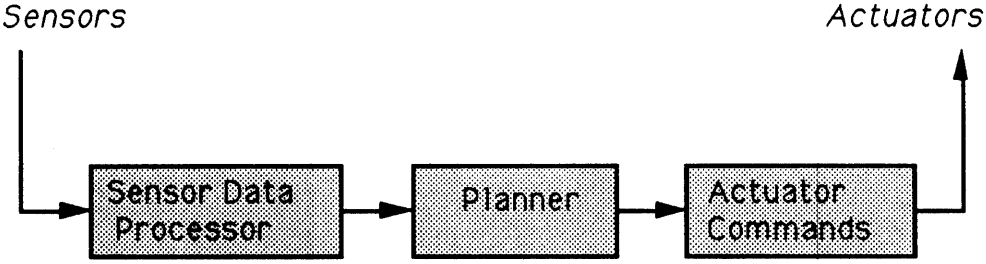


Figure 3.1: An example of a typical horizontal task decomposition into a set of serial computational modules.

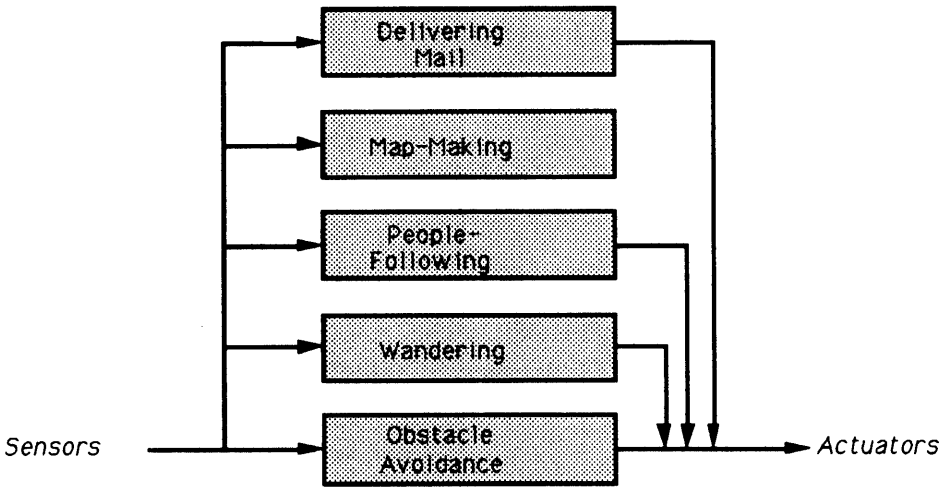


Figure 3.2: An example of a vertical task decomposition into a set of parallel, concurrently-executing computational modules.

process is the output of the previous step, the resulting process in necessarily serial. Thus it is susceptible to failure due to a malfunction in any module. Additionally, it imposes a time delay between sensing and action which may be too long for sufficiently reactive responses required by real world environments. Finally, the system complexity is built into the sequence and cannot be easily simplified. Computational complexity is a particular concern in autonomous robots where all processing must be performed on board.

[Brooks 86] suggests an alternative, vertical approach to decomposing the task in terms of task achieving behaviors. An example of such a subsumption-based vertical decomposition is shown in figure 3.2. The approach creates tight couplings between the sensors and actuators on the robot, separated only by very limited amounts of reasoning in the form of simple rules. The approach is embodied in the *subsumption architecture* which uses finite state machines augmented with timing elements (AFSMs) to construct simple rules **[Brooks and Connell 86]**. The AFSMs communicate through message passing, mutual suppression (one AFSM stops all inputs to another for a fixed time period), and inhibition (one AFSM stops all outputs of another for a fixed time period).

Combinations of AFSMs form *behaviors*, the building blocks of the *new subsumption architecture* **[Brooks 90]**. Behaviors are combined into lev-

els of competence corresponding to the robot's abilities. This approach is more fault-tolerant since failure of any layer does not affect the layers below. Additionally, this organization allows for modular addition and removal of behaviors and thus for incremental design and debugging. Most importantly, it allows for a tight loop between sensing and action which can be performed quickly and with much less computation.

The subsumption architecture approach was demonstrated on a number of robots at the MIT AI Lab. Allen used it for obstacle avoidance and wandering based on sonar data **[Brooks 86]**. Tom and Jerry were a demonstration of minimizing the subsumption code complexity and compiling it down to programmable array logic gates **[Connell 87]**. Herbert was an excellent illustration of an apparently complex system (one which navigates in an unknown environment, picks up soda cans, and takes them home) constructed with minimal state **[Connell 89]**. Genghis is an example of incremental behavior design applied to a six-legged walking robot **[Brooks 89]**. Squirt was a challenge in physical miniaturization in implementing intelligence with minimal hardware and sensor sophistication **[Flynn, Brooks, Wells, and Barrett 89]**.

A question often posed is: "How is subsumption better than other approaches?" or more simply: "What does subsumption buy us?" The subsumption approach does not offer any capabilities which cannot be implemented through one of the classical methods of robot control. Instead, it provides a different approach to the problem. Rather than a recipe for programming robots, it is a set of philosophical concepts about robot behavior design. It stresses the issues of reactivity, concurrency, and real-time control. The set of principles that subsumption condones can certainly be implemented with any other programming language. The behavior language is simply a programming tool which attempts to make the implementation of subsumption-based programs easier, as well as to force a careful consideration of the relevant issues.

The simplicity of the AFSM-based programming environment is not a limiting factor on the complexity of the programs it can generate. Consequently, classical planning could be implemented in subsumption, but that would completely violate the benefits of the concept. The objective is rather to obtain the functionality of a classical planning task, without the use of classical planning. Goal-directed navigation is such a task.

3.2 Choosing a Functional Representation

Goal-directed navigation demands some type of a world representation. A goal is by definition a special location specified relative to other known locations in the world. The robot must know its position relative to the goal in order to decide in which direction to move next, i.e. it must localize within the world model.

The complexity of the world representation influences the efficiency of localization. Any localization algorithm must take into account cumulative position error, sensor error and noise, as well as the incompleteness and inaccuracy of the representation itself. Additionally, the more detailed and analytical the representation, the more complicated it will be to keep it accurate and up to date.

The goal of this thesis is to use a *functional representation*, one that contains only the information necessary for the task. A common pitfall of classical planning approaches is that they rely on known algorithms applied to traditional representations. For example, they employ cartesian models for representing two-dimensional space. Although these models are well understood and are the default choice of most approaches, they are certainly not the most appropriate for all navigation tasks.

Instead of choosing a familiar representation method and building a navigation system around it, it is crucial to develop the world model after the task of the robot is well understood. The task should determine the model, rather than the model constraining the task.

One of the prime motivations behind using the subsumption architecture is the fact that its framework forces a careful consideration of the issue of representation. Using the reactive, behavior based model properly demands a different approach to world modeling. Subsumption-based robots have been shown to be capable of robustly performing various tasks which require little representation, such as collision-free navigation, wall-following, even soda-can collection. One of the main contributions of the subsumption approach was to show that such tasks required little representation.

The goal of this thesis was to explore a distributed, subsumption-based implementation for what is considered to be a representation-intensive task. Map building or spatial learning is a classical example of such a centralized task. This thesis explores a qualitative, distributed method for spatial modeling as a beginning of introducing representation-intensive tasks into the

subsumption repertoire.

Chapter 4

The Robot Toto

4.1 The Simulation Alternative

Simulations are not well suited for generating conclusive test data and results. While they are quite useful for the proof-of-concept stage of research, when the feasibility of algorithms needs to be tested, they do not suffice as proof of algorithm functionality in the real world. A simulation generating successful data tells us much less than a simulation that fails. If an algorithm fails in simulation it will certainly not work in the real world, but the opposite is not necessarily true. A trustworthy simulation requires accurate modeling of the physical processes involved. For mobile robots this means accurate modeling of the robot itself as well as its environment.

Modeling physical sensors has proven to be a difficult task. **[Kuc and Siegel 87]**,**[Kuc and Di 86]**, and **[Letovsky 84]** provide analytical methods for modeling and interpreting sonar data, with varying degrees of complexity. In general, the more physically sound the sensor characterization, the more complex and computationally intensive it is to simulate it. Not only do the simulations not run in real-time, but their failure to do so is due to reasons unrelated to the algorithm they are testing. Very often, the speed of the simulation is limited by a variety of expensive modeling computation required in computational geometry, etc. Since writing a realistic simulator is a difficult task, many compromises are made to simplify it. Unfortunately, each such compromise acts to decrease the value of the simulation. For instance, many simulations use a simple gaussian to characterize sensory error and noise. This generates a sensor behavior often entirely different from that

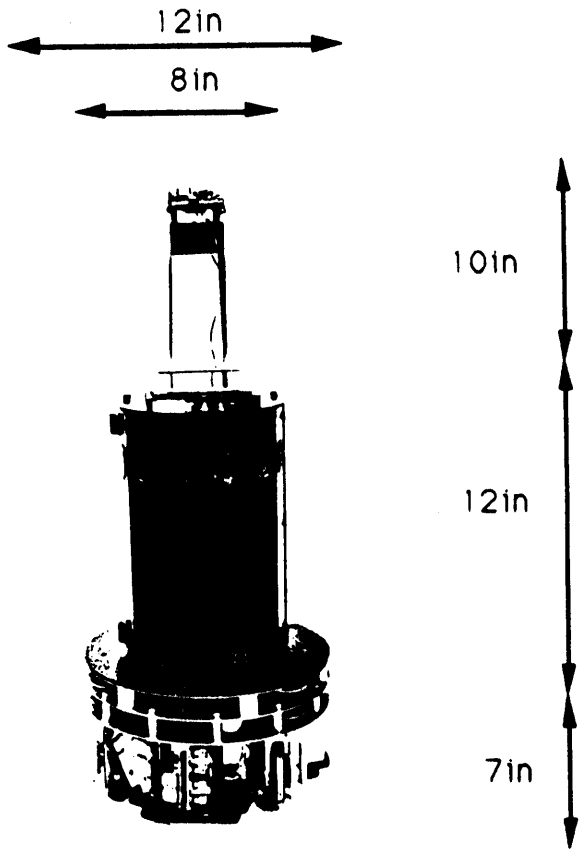


Figure 4.1: The robot: an omnidirectional three-wheeled base equipped with a ring of 12 sonars and a compass.

which would be observed in the real world.

In order to avoid both the difficulties and pitfalls of simulation, a physical robot, Toto, was constructed for testing and debugging all algorithms (Figure 4.1). All data gathering was done on the physical system, in real time, and in unaltered office environments.

4.2 Toto

Toto's only actuator is a Real World Interface three-wheeled circular robot base, 12 inches in diameter, 7 inches high. The wheels and the top platform of the base are connected so as to preserve a forward pointing vector. Since the robot can turn in place by an arbitrary angle, it can continuously follow any trajectory with discontinuous velocity. The built-in motor control processor in the base accepts rotational and translational position, velocity, and acceleration commands.

4.2.1 Sensors

Toto has three basic sensors: current sensors on the base motor, a ring of 12 Polaroid ultrasonic ranging sensors, and a flux-gate compass.

The current sensors on the base motor can detect stalling. This information is used to prevent Toto from pushing helplessly against various environmental barriers.

The sonar sensors are arranged in a ring and mounted on a cylinder, 8 inches in diameter, centered on the base. With the 30-degree cone of each transducer, the ring covers the entire 360-degree area around the robot. Additionally, the small diameter of the cylinder, placing the transducers within an inch of each other, eliminates blind zones in the immediate proximity of the transducers where the cone is not yet widened. The height of the sonar ring (17 inches from the ground) constrains the types of objects the robot can detect. Toto can easily detect all structures relevant to its task, such as walls and furniture. Shelves and any other high-mounted objects remain undetectable.

4.2.2 Sonar Hardware and Software Drivers

The Polaroid Ranging Sensor can be used to measure the distance to the nearest point within the range of 0.9 to 35 feet. The sensor consists of a transducer and a controller board. Upon receiving the signal (VSW) to activate the transducer, the board returns an acknowledgement of the signal (XLG), and sends a 300 Volt, 2.5 amp signal to the transducer **[Polaroid 87]**. The high-voltage pulse in the transducer generates a single-frequence ping. The transducer acts as both a transmitter and receiver; the necessary settling

time of the transducer membrane prevents instantaneous pulse detection. Consequently, the minimum detectable range for the sensor is 0.9 feet. The first echo it receives triggers a flag (FLG) from the controller board.

The twelve transducers used on the robot are driven by two Polaroid controller boards. This allows for simultaneous activation of two sonars. Only the diametrically-opposed transducers are activated to minimize the probability of echo interference. Sonar travels at the speed of sound (331.4 meters/second) which is the limiting factor of the sampling rate. Each transducer must wait an interval appropriate to the transducer's range for the receipt of the return echo. Due to the physical proximity of the transducers, the echos of neighboring sonars would easily interfere. This forces serialized activation once every 200 milliseconds, resulting in the sampling rate of 1.2 seconds for the entire ring.

The sampling rate of the sonar ring is the limiting factor of the robot's velocity. While the base can move at 2 meters/second, its safe velocity is limited to 20 centimeters/second due to the slow data refresh rate.

Sonar data acquisition is performed by a dedicated Hitachi 6301 microprocessor. The processor generates a trigger signal which is simultaneously sent to both sonar controller boards, resulting in near-concurrent transducer activation. The XLG and FLG signals are connected to different ports on the 6301, and signal timing and distance computation are performed in 6301 assembler software. The distances computed from the echo delays are sent to the main processor via a serial line.

The selection of the transducers to be activated is performed with a switching mechanism constructed with a set of ten mercury wetted reed relays. The mechanism, shown in appendix A, selects the relays according to a three-bit input pattern sent by the microprocessor. The hardware allows for choosing an arbitrary ordering of transducer activation, but the final system does not utilize this ability. For Toto's purposes, it is sufficient to continually obtain information about the distances in all directions surrounding the robot. Both the compass and the sonar ring provide continuous data streams which are sampled when needed. Due to its low data rate, the sonar data are sampled as quickly as they are available. Each new packet is sent to the central processor via a 9600 baud serial line.

The final sensor on the robot is a flux-gate compass. The compass is mounted on top of the robot to isolate it from magnetic interferences from the rest of Toto's hardware. It provides an analog signal which is discretized

with a special-purpose processor board to provide four bits of bearing. The compass is connected to the central processor via a serial line transmitting bearing information at 9600 baud.

4.2.3 The Power Supply

The power for the base and all the electronics on board the robot is provided by rechargeable silver-zinc batteries mounted under the base. The batteries provide +12 Volts, which are regulated to +5 Volts for the electronics, and +6 Volts for the sonar controllers. The power supply for the sonar circuitry is isolated from the rest of the electronics through the use of DC-to-DC converters. This is necessary due to the sonars' high current requirements and voltage spikes.

4.2.4 The Central Processor

Toto's computational hardware is located inside its cylindrical body. It consists of a half-height VME backplane, mounted horizontally on the base plate. The bus contains the sonar driver/relay selector board, the main processor board, and the extension memory/serial port board. All processor boards were developed at the MIT Mobile Robot Lab. The sonar driver and relay selector boards were custom developed for Toto. Their component diagrams and schematics are shown in appendix B. The processor boards can be used as a sensing subsystem for future robots using sonar.

All of Toto's processing is performed by a Hitachi CMOS 68000 microprocessor with 64K bytes each of RAM, ROM, and NVRAM [**Ciholas 88**]. Several 68000 assembler routines are used to interface the sonar controller board and the compass with the main processor board, using the extra serial ports provided by the serial port board. The serial ports on the 68000 bus are used for debugging. Figure 4.2 is a schematic showing the physical organization of the boards and communication lines constituting Toto's computational hardware.

A Toshiba liquid crystal display (LCD) is mounted on top of the robot's body and used for debugging purposes. Unfortunately, it produces magnetic fields which interfere with the compass resulting in the need to elevate the compass 10 inches above the top of the robot's body. The robot's effective height is increased by 28%, but its sonar sensors provide no information

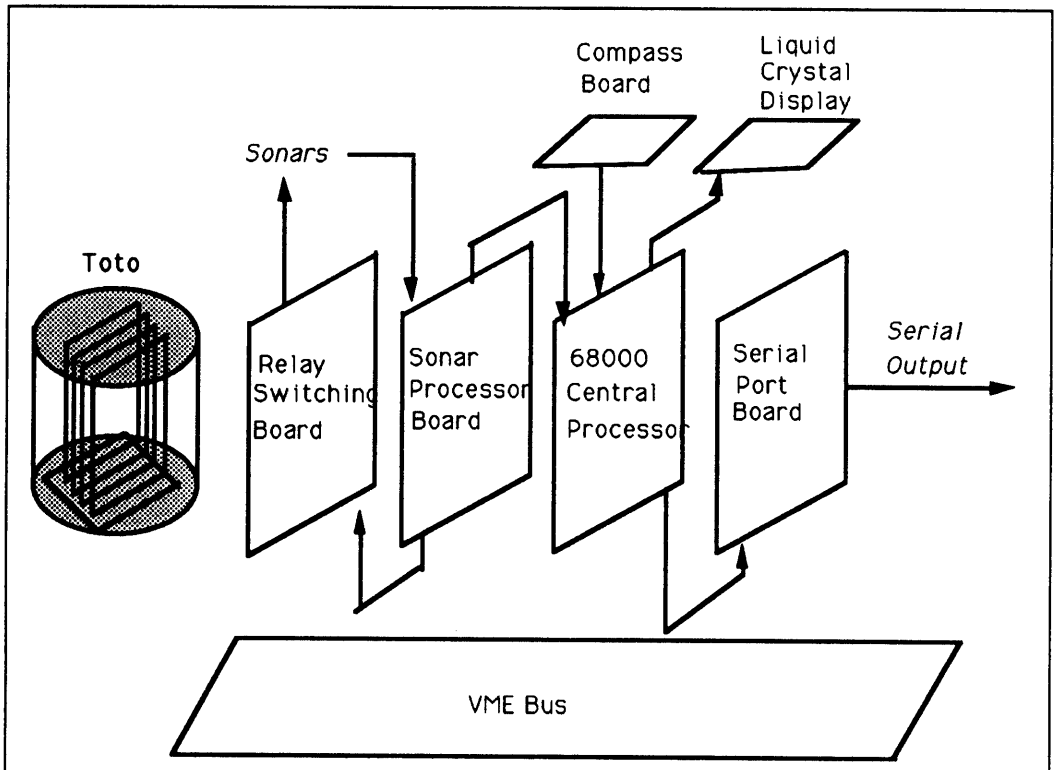


Figure 4.2: Toto's computational hardware: the relay switching board which selects the sonar transducers, the sonar processing board which performs signal timing, the main processor board executing the robot's behaviors, the auxiliary serial port board, the compass board, and the liquid crystal display used for debugging. The arrow indicate communication lines between the processor boards, the sensors, and the outputs.

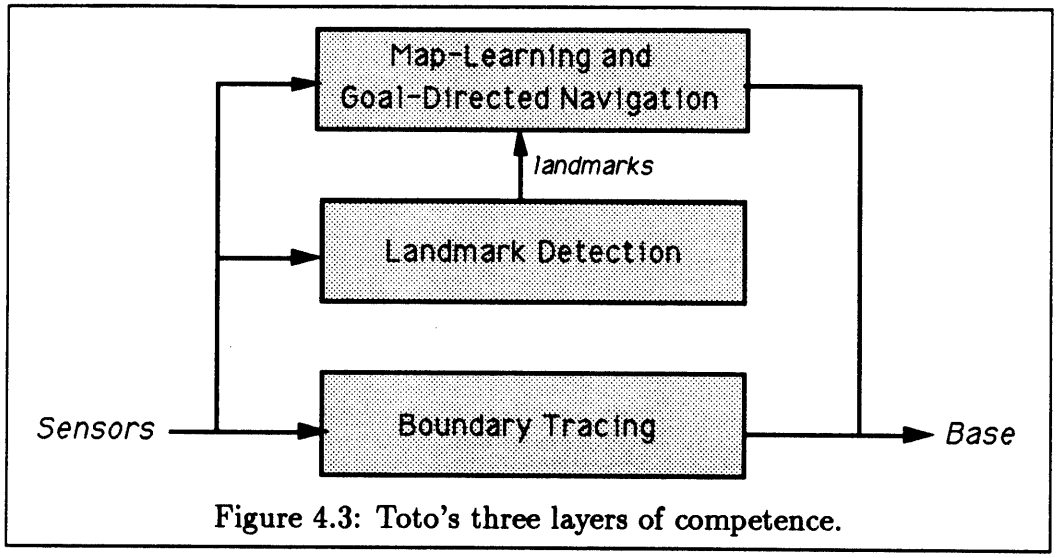


Figure 4.3: Toto's three layers of competence.

about the added area. Therefore, the robot may attempt to wander under high tabletops unaware of its telescoping high-mounted compass, which has a decapitating effect. This problem can easily be eliminated with an additional range sensor or a limit switch, but has not presented cause for concern in experiments up to date.

4.3 The Programming Environment

Toto's software was written in the Behavior Language, which compiles to the *subsumption architecture* language which, in turn, compiles to 68000 assembler code. The Behavior Language is a real-time rule-based parallel programming language **[Brooks 89b]**, an extension of the *subsumption architecture* **[Brooks 86]** **[Brooks and Connell 86]**. In contrast to the subsumption architecture language, which was programmed with interconnected augmented finite state machines (AFSMs), the Behavior Language groups AFSMs into behaviors. Each behavior is a coherent collection of related real-time rules producing a desired set of responses. A collection of interrelated behaviors defines a *layer of competence* of the robot. For example, Toto's interaction with the world is governed by three such layers of competence: collision-free boundary tracing, landmark detection, and map-learning and goal-directed navigation (figure 4.3).

Toto's software is a collection of behaviors which receive inputs from the sonars and the compass, as well as from other behaviors. Behaviors can output commands to actuators and other behaviors. Since the base is the only actuator on the robot, it is controlled by a dedicated low level behavior **[Brooks and Flynn 89]**. Any commands to the base are sent to this behavior. The rest of the communication on the robot is accomplished through message passing among behaviors. Explicit suppression and inhibition is minimized, as is communication between different layers of competence. Most communication occurs among the concurrently acting behaviors within a single layer of competence.

Minimizing inter-layer communication makes the system modular, and therefore more generally applicable. Each layer is independent from those above, and its dependence on the layers below is simplified as much as possible. Landmark detection, which is the second layer of Toto's system, relies on the first layer, collision-free object-tracing, only to the extent that it assumes its function. The second layer receives its inputs directly from the sensors. The third layer of the system builds maps based on landmark information received from the second layer. It expects landmarks as input, but is independent of the type of system which provides that information. In general, each layer assumes the functionality of those below, but does not depend on the way that functionality is realized.

4.4 Summary

In order to avoid the difficulties and pitfalls of simulation, all algorithms were implemented and tested on a physical robot. Toto is equipped with a ring of twelve ultrasonic ranging sensors and a compass, mounted on a 12-inch circular, omnidirectional base. All computation is performed onboard by a CMOS 68000 microprocessor. The robot is programmed in the Behavior Language.

Chapter 5

Navigation

5.1 Motivation

The motivation behind the presented approach was to implement an intuitive navigation method, in contrast to some more analytical approaches, such as potential fields. While potential fields are a simple way of processing radial sonar data, they do not map naturally to our understanding of reflexive behaviors in the biological systems we use as our models.

The goal is to implement navigation as a result of a collection of interacting behaviors. Each behavior consists of a set of rules associating some conditions in the world with appropriate actions. The rules are designed to be intuitive, and are of the form: "If approaching an obstacle on the right, turn to the left."

A set of important states of the world is selected and defined as a set of sensory patterns. Some states are defined as the absence of certain emergency patterns; they generate default actions. Each pattern triggers the appropriate reflex behavior. Since the world provides continual stimuli, some set of reflexes is activated at all times, resulting in a continuous stream of actions. The combinations of these actions results in the desired emergent behaviors.

5.2 Sensor Characterization

The navigation algorithm is strongly determined by the sensors available on the robot. Ultrasonic ranging sensors are an inexpensive means of obtaining

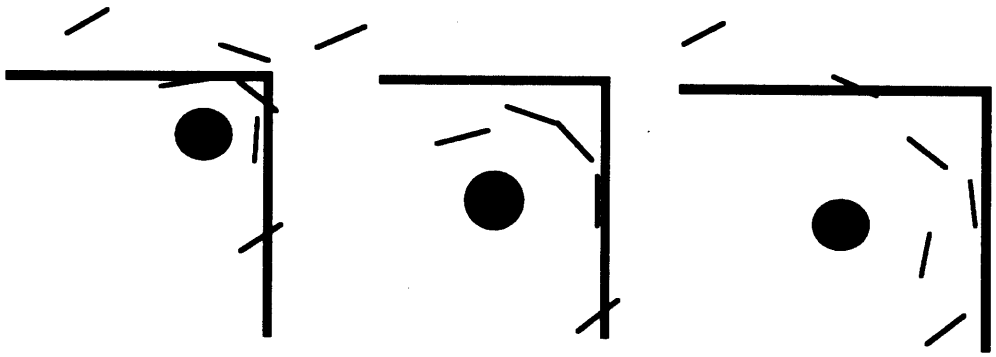
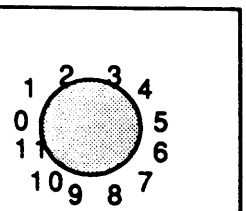


Figure 5.1: A typical set of radial sonar signatures plotted from real data. Since the incident angle determines the accuracy of the reading, near-perpendicular head-on transducers are more accurate than the lateral transducers.

directional distance information. Consequently, they have been used extensively in mobile robot applications. Much literature exists formalizing the limitations of ultrasonic ranging sensors and suggesting various analytical approaches to their application **[Kuc and Siegel 87]** **[Kuc and Di 86]** **[Letovsky 84]**.

After analyzing the task for which the sensor is to be used, a single crucial property of the sonar emerges as its sufficient characterization. The ultrasonic ranging sensor has high accuracy (near 95%) when the incident angle of the beam is less than 15 degrees from the normal **[Polaroid 87]**. At larger angles the sensor often suffers from specular reflection **[Flynn 88]**. The farther from perpendicular the incident angle, the higher the probability of specular reflection, resulting in a falsely long reading. Consequently, long readings have a higher probability of being inaccurate than short readings. This characteristic leads directly to the guiding heuristic of the functional sensor characterization: short readings are reliable, while long ones are not.

Figure 5.1 shows a set of three typical radial signatures. It illustrates the high accuracy in transducers near-perpendicular incidence angles, and specularities in the rest. Figure 5.2 tabulates the signatures varied over the entire ring. The data are consistent for each direction regardless of the transducer used. Consistent readings in a particular direction remain invariant for all transducers, as illustrated in columns 3, 5, 6, 8, 10, 11 and 12 of the table.



Trans	Distance											
0	80	300	95	85	115	200	25	400	165	80	120	120
1	80	130	95	85	115	205	25	390	165	80	120	120
2	80	120	95	80	115	205	40	400	165	80	120	120
3	80	440	95	85	115	205	45	400	165	80	120	120
4	175	405	95	80	115	205	55	400	175	80	120	120
5	185	290	95	85	115	205	25	400	175	80	115	120
6	80	130	95	85	115	205	30	400	175	80	120	120
7	80	140	95	85	115	205	30	400	125	80	120	120
8	80	290	95	85	115	205	30	400	165	80	120	120
9	80	280	95	85	115	205	25	400	185	80	120	120
10	80	265	95	80	115	205	25	400	170	80	120	120
11	80	120	95	80	115	205	25	400	175	80	120	120

Figure 5.2: A test configuration of the robot in the environment is shown. The robot is rotated clockwise in place by 30 degrees in order to show the consistency in performance between the 12 transducers. The table plots the results. The columns with consistent readings correspond to transducers with near-perpendicular angles of incidence. More variation is found in columns representing transducers with oblique angles. The data show reliable repeatability among different transducers

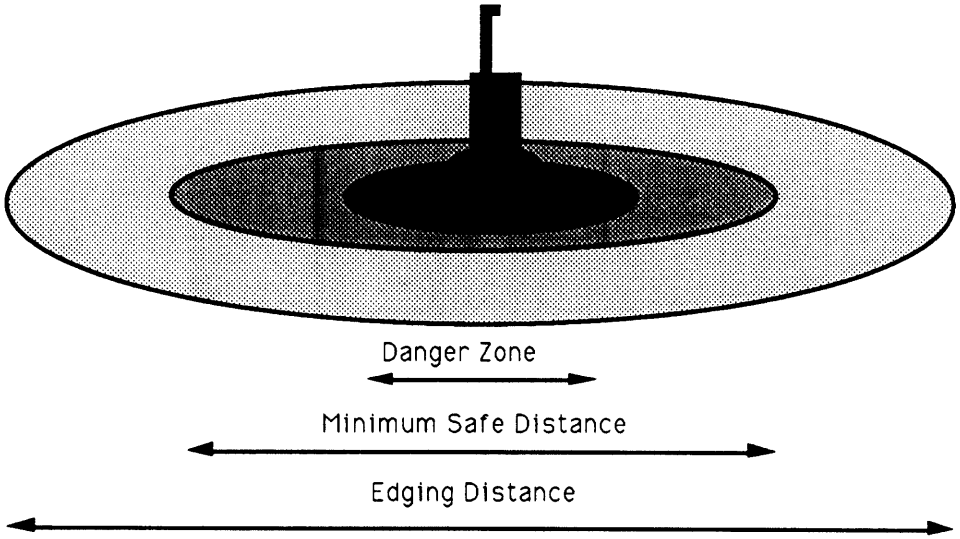


Figure 5.3: A schematic of the distance circles defining the three regions around the robot. These regions are used to implement low-level navigation rules which combine to produce boundary tracing behavior.

In contrast, the readings in column 2 vary as a result of specular effects due to the transducer's position relative to the objects.

5.3 The Navigation Rules

Figure 4.3 shows collision-free navigation as the robot's lowest competence level. The goal of the navigation behavior is to follow along the boundaries of the objects in the world while avoiding both dynamic and static obstacles. The avoiding behavior is simply a survival mechanism while boundary following is the basis of the robot's perception of the world.

The navigation rules rely on three distance regions or circles around the robot. In order of increasing radii, those are: the danger zone (1 foot), the minimum safe distance (2 feet), and the edging distance (2.3 feet, see figure 5.3. These boundaries utilize the short distance accuracy of the sensors to keep the robot neither too close nor too far from the objects in the world. The robot avoids any objects within the danger zone, attempts to stay near the minimum safe distance of the object it is following, and avoids getting out

of the edging distance region from the object whose boundary it is tracing.

The choice of these distances is empirical, based on the robot's velocity which, in turn, is determined by its sonar sampling rate. Given the sampling rate of 1.2 seconds per full sonar scan, the safe velocity of 20 centimeters/second allows Toto to prevent collision with all static and most dynamic obstacles within 0.3 meters. This defines the danger zone. Any dynamic obstacle which unexpectedly appears in the danger zone and moves toward the robot at a velocity nearly equal or higher than the robot's, will cause a collision. An obstacle in the minimum safe distance or farther can be avoided once detected. Finally, the edging distance is chosen so that the robot does not veer too far from the object but still allows it a buffer area within which to move. Since there is no attempt at position control, the algorithm does not aim to keep the robot at a constant radius away from objects, merely within a desirable range. That range is defined between the danger zone and the edging distance, optimally around the minimum safe distance.

As mentioned previously, these threshold radii were selected empirically, but could also be learned by the robot, through trial and error. While optimized for Toto's parameters, they can easily be adapted to fit a robot with a different geometry or velocity constraints.

An important feature of the wandering algorithm is that it involves no explicit arbitration among the constituent behaviors. Each of the rules is triggered by discrete and mutually-exclusive sensory characteristics, based on the three threshold radii around the robot. Consequently, arbitration is implicit.

The desired object-tracing behavior is the emergent result of the interaction of the following four simple navigational rules [**Mataric 89**]:

Stroll:

```
(defbehavior stroll
   (cond
((and (<= shortest-sonar danger-zone)
      (not stopped))
  (stop))

((and (>= front-sonars min-safe-distance)
      (not stopped))
```

```
  (move forward 3 meters))

((and stopped
      (or (> left-distance min-safe-distance)
  (> right-distance min-safe-distance)))
  (if (> left-distance right-distance)
 (turn left 30 degrees)
 (turn right 30 degrees)))

(t
   (move backward 0.4 meters)))
```

This behavior sends stop, go, and backup commands to the base, depending on Toto's distance from the danger zone. It allows the robot to move safely forward.

If there is no obstacle in the minimum safe distance range of the front two sonars, the robot continuously moves forward. It is repeatedly given a target distance to traverse, which serves like a carrot on a stick. Rather than getting discrete instructions to move forward to a certain location, the robot constantly receives "encouragement" to keep moving toward a perpetually escaping goal, which results in smooth, continuous motion.

If any of the transducers in the front detect an obstacle within the danger zone, the robot stops.

If stopped and within the danger zone of the obstacle, the robot backs up. This is another defensive behavior which allows the robot to get out of tight spots and away from unexpected obstacles. Consequently, if the obstacle is moving (e.g. a person walking close by) the robot will stop briefly. By the time it receives its next sensory reading the moving obstacle will have disappeared, and the robot will resume in its original direction. If the obstacle is still detected, the robot backs up. This strategy allows for minimization of course changes due to transient obstacles.

Stroll alone provides the robot with the basic safe straight-line motion. It allows it to move forward and stop and back up whenever necessary.

Avoid:

```
(defbehavior avoid
```

```
(if (>= minimum-of-the-two-sides min-safe-distance)
  (if (and (<= front-left-sonars min-safe-distance)
   (<= front-right-sonars min-safe-distance))
      (if (< left-side right-side)
   (turn right 30 degrees)
   (turn left 30 degrees)))

        (if (= minimum-of-the-two-sides front-left-sonars)
       (turn right 30 degrees)
       (turn left 30 degrees))))
```

If an obstacle is detected within the edging distance of the front sonars, the robot turns away from it. If it appears on the left, it turns to the right, and vice versa. If the obstacle is straight ahead, the robot consults its side sonars to determine the safe direction in which to turn. It turns in the direction which is not occluded by close obstacles. Appropriate heading changes are sent to the base behavior. In conjunction with *stroll* these rules result in an emergent collision-free wandering behavior. An example of this behavior is illustrated in figure 5.4. All figures show real data obtained from physical runs of the robot.

The robot moves freely around the world and is only forced to stop if an unexpected obstacle appears in its way. Any static object is detected and avoided by veering. Stopping is a defense-mechanism which is useful with dynamic obstacles, and rarely gets activated in static environments.

Align:

```
(defbehavior align
(if (and (<= minimum-of-all-directions edging-distance)
 (>= left-side edging-distance)
 (>= right-side edging-distance))
(if (= minimum-of-all-directions rear-right-two-sonars)
  (turn right 30 degrees)
(turn left 30 degrees))))
```

This behavior simply but effectively implements "stage fright": it keeps the robot from meandering away from the object boundary it is following.

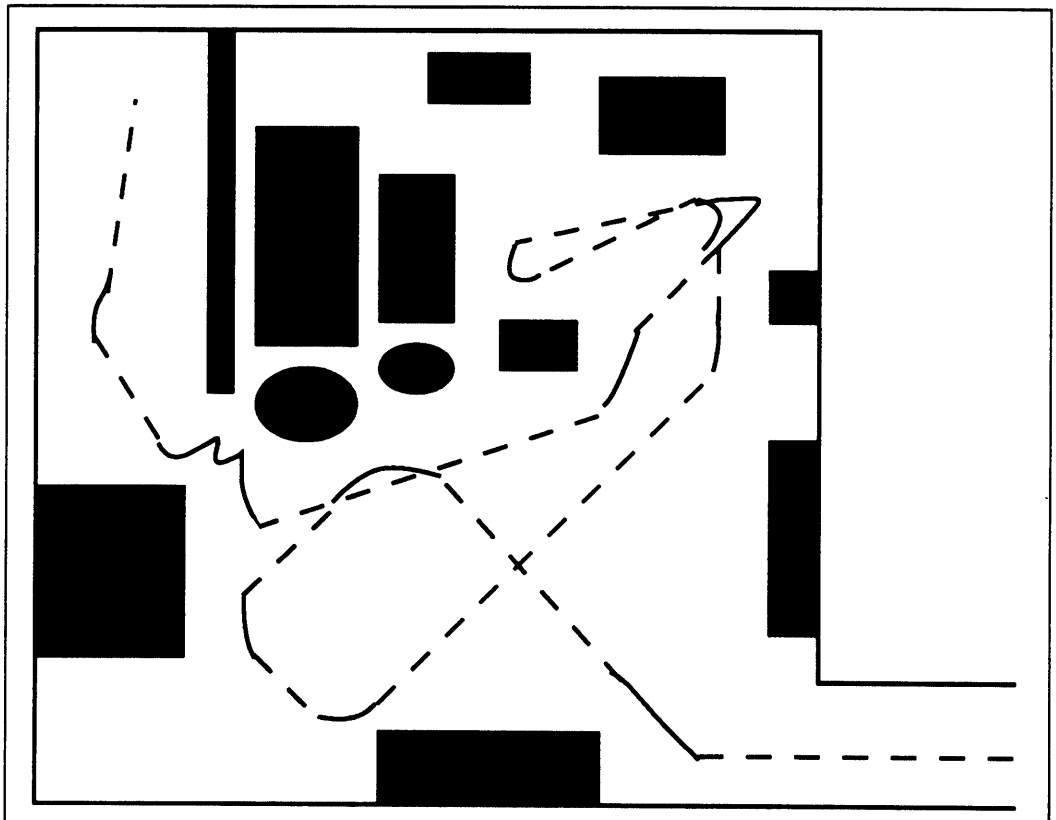


Figure 5.4: The performance of the combined stroll and avoid behaviors. Stroll produces straight-line path segments shown with dashed lines. Path segments generated by avoid are shown with continuous lines.

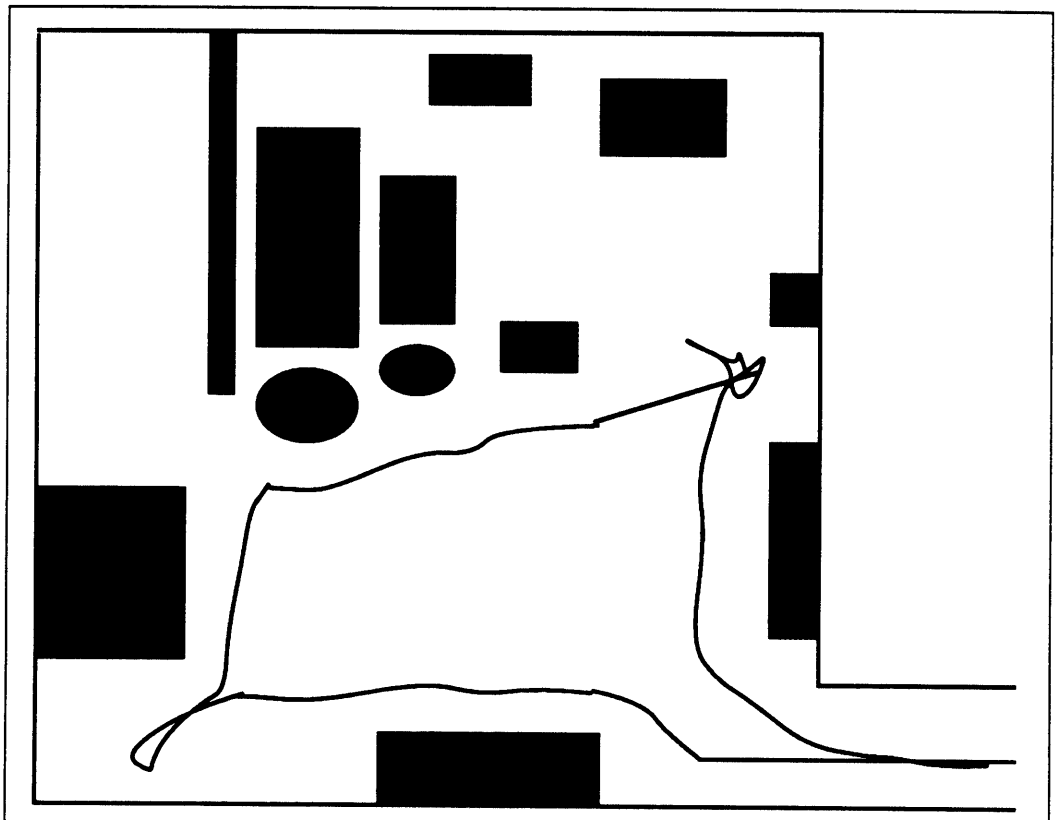


Figure 5.5: The performance of the boundary-tracing behavior as a combination of avoid, stroll and align. These three low-level behaviors allow the robot to follow any convex boundary.

If the robot turns away from the object and is getting out of the edging distance, it turns back towards it. The process of returning to the desired alignment is incremental. The rule simply states that if the distance behind the robot is shorter than that in the front, that it should turn by a small angle in the appropriate direction. This rule will be activated as long as the robot is not aligned with an object on either of its sides.

As the robot moves away from the wall, its side sonars detect the loss of a boundary on its right side, and make it turn slightly to the right. The process is repeated until the robot is aligned to the wall again.

The combination of *avoid*, *stroll*, and *align* allows the robot to follow convex, straight, and curving boundaries. The schematic of the behavior is shown in figure 5.5. The robot remains oblivious to doorways, sharp turns, and T-junctions.

Correct:

```
(defbehavior correct
(if (> minimum-of-the-front-quadrant edging-distance)
(if (and (>= side-right-first edging-distance)
 (<= side-right-second edging-distance))
(turn right 30 degrees))
  (if (and (>= side-left-first edging-distance)
    (<= side-left-second edging-distance))
    (turn left 30 degrees))))
```

This behavior allows the robot to turn around sharp corners. It keeps a single bit of history in order to compare a previous sonar reading with the current one. It uses the values of the two adjacent side sonars on the side of the robot next to the boundary that is being traced. (The robot decides which side to turn toward based on which is closer to an object. In a narrow space it may alternate between the two sides but continues to follow one of them. See next section for an example of such corridor following.)

If one of the two lateral sonars on the side of the robot next to the object loses sight of the boundary, the robot compensates by turning in the direction closer to that boundary. This, too, is an incremental process; the robot makes a series of small heading changes until the desired heading is

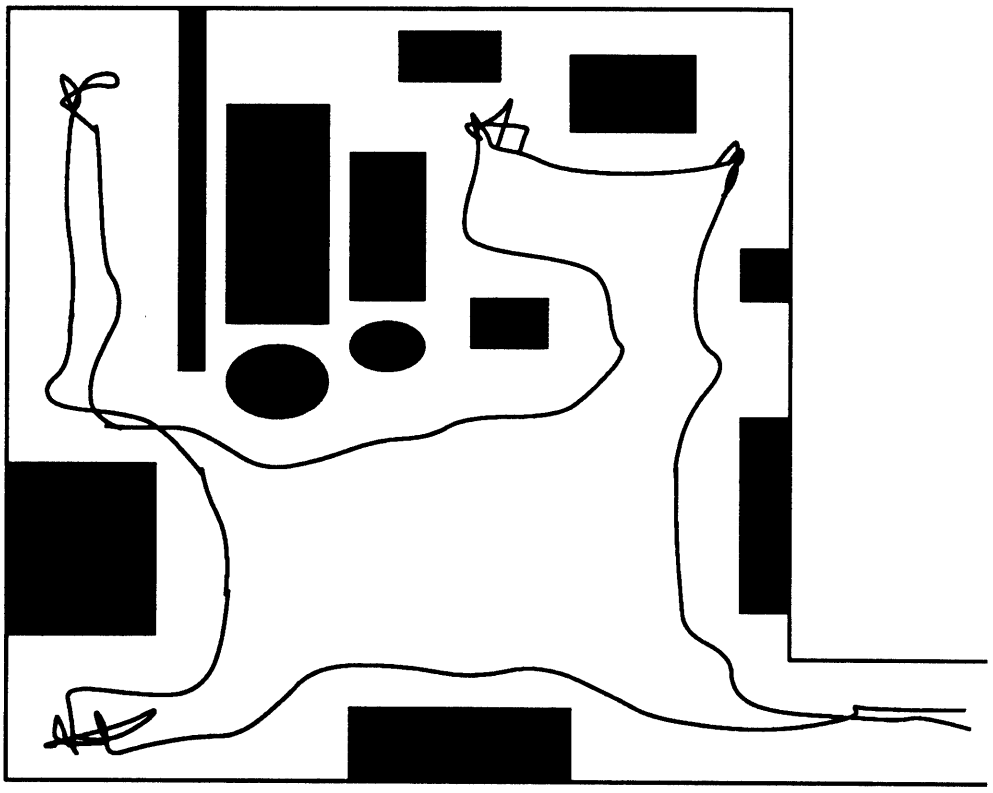


Figure 5.6: The performance of the complete boundary-tracing behavior. This behavior emerges as the result of the interaction of avoid, stroll, align, and correct, which combine to allow the robot to follow any concave or convex boundary.

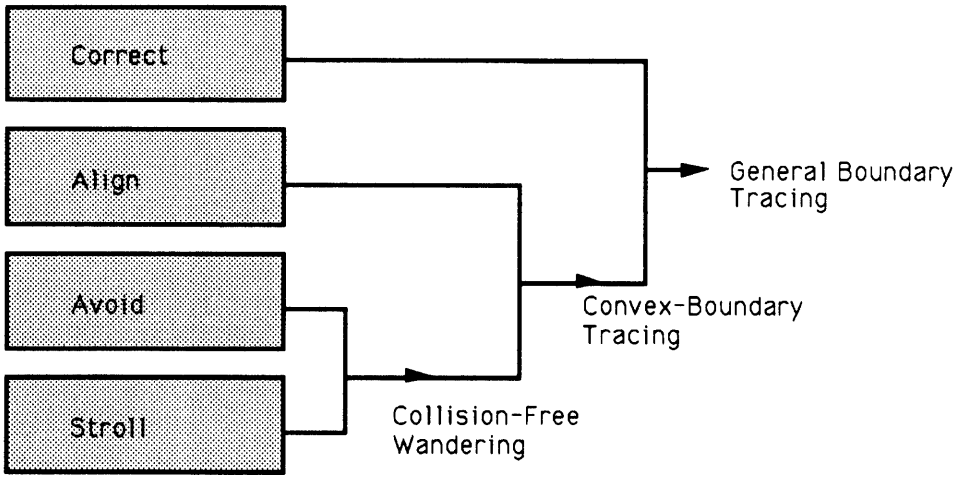


Figure 5.7: A schematic showing the incremental interaction of the low-level navigation behaviors resulting in emergent boundary tracing. The addition of each new behavior adds to the overall competence of the robot.

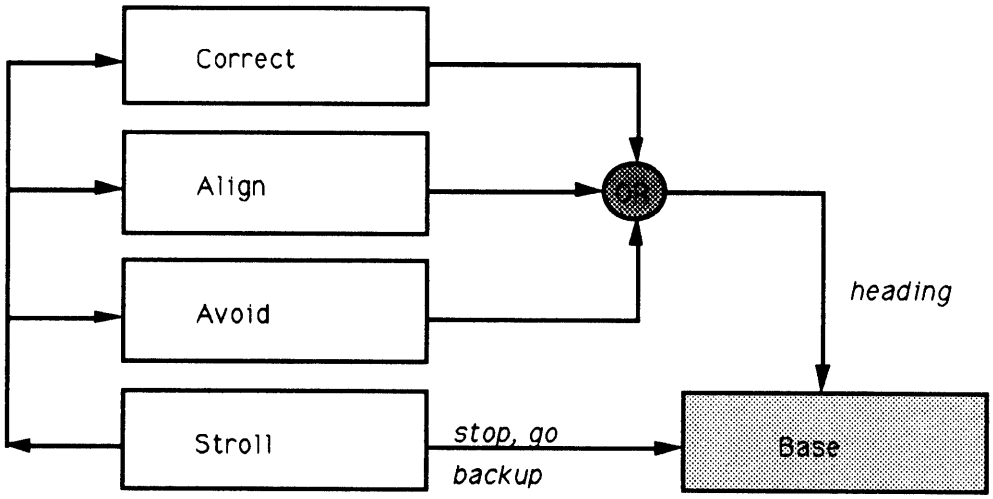


Figure 5.8: A schematic illustrating the implicit arbitration among the low-level navigation behaviors. Since the conditions triggering each of the four low-level behaviors are mutually exclusive, no explicit arbitration is needed.

reached. A universal 30 degree angle of rotation used for all heading changes sent to the base. The angle is equivalent to the width of the sonar cone. This choice guarantees a new, non-overlapping sonar cone direction for each transducer after a single turn. At the same time, the angle is small enough to avoid overcompensation past the desired range.

With this simple correcting rule the robot is able to follow sharp corners with arbitrary degree. Figure 5.6 illustrates an example of the robot turning around a sharp corner. As one of the pair of sonars on the left side loses sight of the wall, it makes the robot turn to the left. The rule is activated until both transducers detect the wall within the correct range.

The four behaviors interact to produce a robust boundary-following behavior. Figure 5.8 illustrates the interaction of the low-level behaviors resulting in the desired boundary-tracing. Figure 5.7 shows the implicit arbitration among those behaviors.

The use of gradual heading correction is an example of the qualitative approach of our method. The robot is controlled through the world rather than from an internal set of desired configurations. Rather than moving an exact distance or turning by an exact angle, it uses the world as its feedback to decide both when it needs to change its direction, and by how much. The approach consists of continuous execution of small, incremental improvements for which the conditions are met, rather than a sequential execution of a series of discrete, precise steps.

It is worth noting that, with the exception of the range boundary values, no metric information is used for navigation. Distances are compared and their relative sizes are used to make heading change decisions. The only other metric information is provided by the compass. Its four bits of bearing are used only as a reference to be compared against a broad range.

5.4 Emergent Properties

The four simple behaviors described in the previous section result in several useful emergent properties. The most important behavior which results is that of reliable tracing of the boundaries of the object in the world (Figure 5.7). This behavior is the basis of both landmark detection and goal-directed navigation algorithms. Figure 5.9 shows a cumulative plot of four independent runs of the robot in a large room with irregular boundaries con-

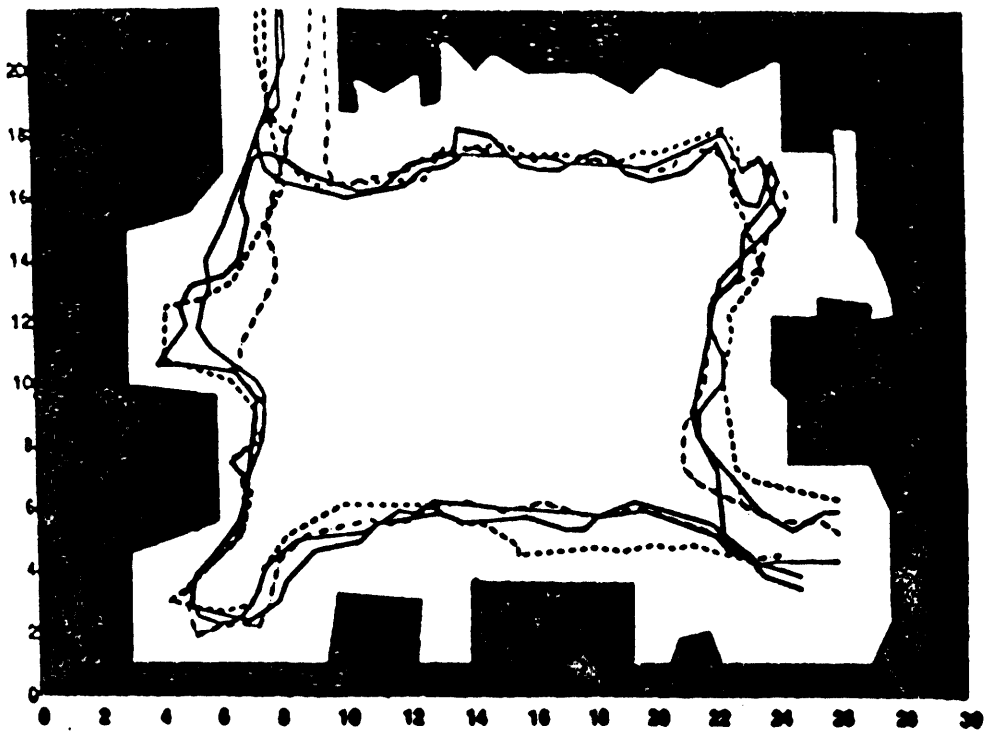


Figure 5.9: A plot of four independent real robot runs manifesting consistent object tracing. The room used for the experiment is the ninth floor playroom of the MIT AI Lab. The data were gathered by attaching a marker on the base of the robot and recording its path on the tiled floor.

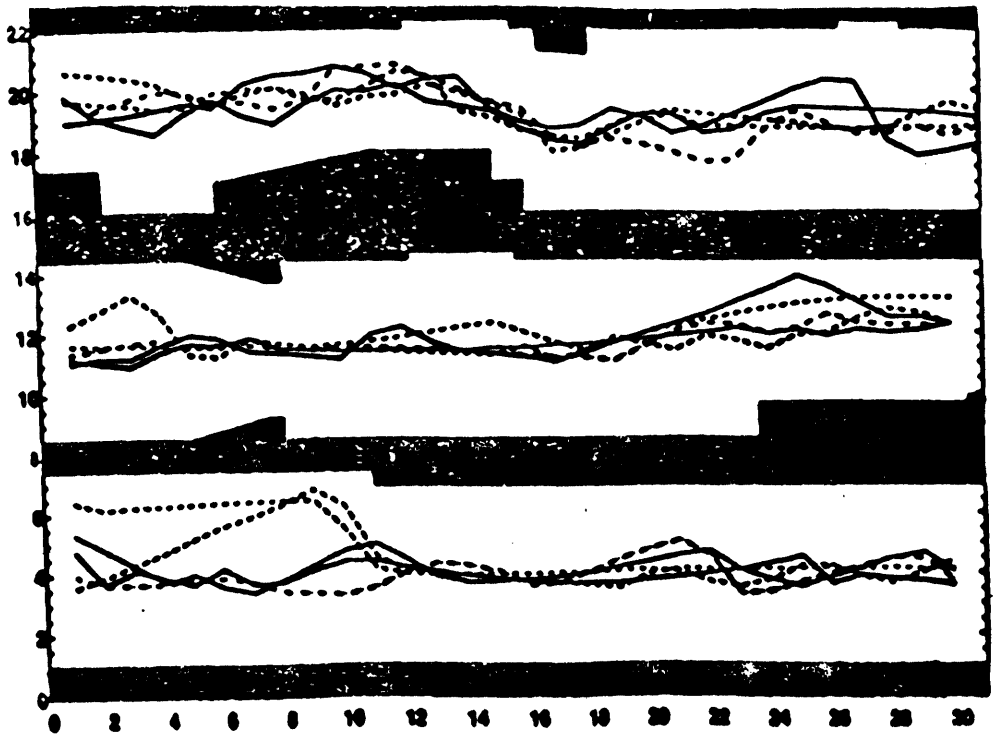


Figure 5.10: A plot of five independent runs showing convergence to the middle of a corridor on the ninth floor of the MIT AI Lab. The plot shows three consecutive sections of the same corridor, going left-to-right and bottom-up. The same technique for data gathering was used as previously described. The scale shown is in feet.

sisting of walls, chairs, tables, and retired robots. The data show reliable edge following in all trials.

Corridor following is also an emergent behavior. Figure 5.10 is a plot of five independent runs showing the robot's convergence to the middle of the corridor. The robot was started in different positions in each of the trials. The convergence to the middle of the corridor is an artifact of its width as related to the maintained distance thresholds. Any corridor with a width smaller than twice the edging distance will be followed in the middle. The shown corridor falls in that category. In a wider corridor the robot will follow one of the walls.

All data were gathered by attaching a marker to the robot base and

recording its motion on a tiled floor. The 1'X1' tiles provided an accurate grid for plotting the data.

5.5 Summary

The described navigation algorithm demonstrates successful object tracing as an emergent property of a set of interacting behaviors. Each behavior in the collection is a combination of simple rules relating specific sensory patterns with velocity and heading changes. This approach takes advantage of intuitive pairings between stimuli from the world and actions from the robot.

An important feature of this navigation algorithm is its independence of the type of range sensor used. Since the algorithm does not utilize sensor signatures but rather independent readings, it works with any type of sensor providing range in the desired direction. For example, the algorithms would work with a single range sensor as well (sonar or infrared), rotating to provide data from appropriate directions.

A useful feature of this approach to low-level navigation is the implicit arbitration among its constituent behaviors. This property keeps their interaction tractable which greatly helps in the debugging process, as well as in the analysis of the robot's performance. The simplicity of the inter-module interactions allows for purely incremental design which also keeps the system tractable since its behavior can be tested by cleanly separating the various competence levels. Finally, the dynamic approach resembles intuitive stimulus-response reflex couplings which most likely control navigation in simple biological systems.

Chapter 6

Landmark Detection

6.1 About Landmarks

The concept of a landmark is used extensively in navigational studies ranging from insects to humans and robots. Although intuitively clear, the concept is difficult to define. A landmark is any element (object or feature) which can serve as a point of reference **[Presson and Montello 88]**. According to Piaget, a landmark is a spatial primitive, and thus a basic building block of spatial representations **[Piaget and Inhelder 67]**. Three distinct landmarks are necessary and sufficient to localize any point in two-dimensional space **[Pick, Montello and Somerville 88]**. In addition to localizing within a single reference frame, landmarks are used as registration marks for aligning multiple frames. This role is useful in integrating local spatial information into a global representation.

Most landmark studies rely on visual cues. However, the notion of a landmark generalizes to any reference feature, as perceived by the available sensors. Auditory cues are ubiquitous as temporal landmarks (e.g. the school bell). Olfactory cues are used extensively by insects and animals, and play a prominent role in human memory **[Gould 82]**.

The use of landmarks by the blind is a good example of non-visual cue use for spatial orientation. The stimuli used include aural, tactile, and olfactory information. The blind tend to build lists of landmarks as paths between known locations in the world. A typical list is of the form: "Go straight until A, then turn left, keep going until reaching B, then turn right..." These paths are essentially topological. They rely on some method of recognizing

the landmarks in geographic space, and on the properties of the egocentric reference frame (front-back, left-right). Studies also show that people confronted with new, unknown environments (such as new cities), prefer the qualitative list of directions for reaching a goal, rather than a geometric map **[Kender and Leff 89]**.

Since the notion of a landmark is so vague, the process of landmark selection remains difficult. Psychological literature cites examples of landmarks selected on the basis of their distinctiveness value, or the saliency within the given context **[Anooshian 88]**.

In humans, the abundance of available sensory information makes the analysis of the selection and representation of landmarks difficult. In robotics, the problem of landmark selection is simplified by the limitations of the sensors. A landmark should be a feature or location which is robustly and reliably recognizable by the sensors. Consequently, a landmark is an extreme point in sensor space. This is the basis of the approach proposed by **[Kuipers 87]**.

The intuitive notion that larger objects serve as better landmarks for people has been confirmed in experiments **[Lockman 88]**. The rule especially applies to learning large space, which is one of the thrusts of this research.

A natural bias is to select landmarks which are meaningful to the robot designer. Unfortunately, those correspond to salient features in human sensory and semantic space, and usually not that of the robot. Often, much effort is spent in designing sensors to detect such features. This thesis presents an orthogonal approach; it utilizes the features in the environment which are easily and reliably detectable by the robot's sensors as a basis for defining landmarks. It is not surprising that those landmarks seem unusual to human observers. They correspond to what is usually thought of as connections between landmarks in the world, rather than actual landmarks. It is interesting is that they serve as effective landmarks as well. This stresses the difficulty and variability, as well as context dependence, of landmark selection.

6.2 Dynamic Versus Static Landmark Matching

A common approach to landmark detection is matching a received sensory pattern or signature to the stored model of a landmark. This approach

is commonly used with sonar data. For instance, **[Drumheller 87]** used real time sonar data for static localization, using a model-based matching algorithm (see chapter 2). **[Kuipers 87]** modeled sonar as a point-source sensor. He used a hill climbing algorithm for matching such idealized sensor data in simulation. While the simplicity of the approach is appealing and may work in simulation, using a physical sonar introduces a variety of difficulties which may make this model unrealistic.

Given the characteristics of the sensor (see chapter 5), it is improbable that identical sonar signatures will be generated in different trials on a physical robot. Additionally, static matching schemes often rely on positional control, whose accuracy is difficult to maintain due to wheel slipping, friction, and other factors resulting in cumulative errors. Consequently, static approaches to landmark recognition require a sophisticated matching process. The matcher must take into account sensor error and noise, as well as positional inaccuracy.

In contrast to static matching, the algorithm described here defines and detects landmarks dynamically. The constant motion of the robot is utilized, in conjunction with its boundary-following behavior. The robot monitors its proprioceptors, and communicates with itself through the world **[Connell 88]**. Rather than taking a snapshot of the world and executing a series of planned actions, it continuously senses and acts incrementally.

We can view this dynamic approach as utilizing a *procedural representation* of landmarks as compared to *declarative models* usually used. The advantage of the dynamic approach is its generality: it is independent of the specific sensory system on the robot. It will work with any sensor which provides proximity data regardless of its exact modality or physical structure.

Another advantage of dynamic landmark detection is its computational simplicity. It does not require an analytical model of the sensors, or a sophisticated matcher for recognizing the landmarks.

6.3 The Dynamic Landmark-Matching Algorithm

Toto's navigation algorithm (see chapter 4) produces a path around the boundaries of objects. The landmark detection algorithm uses this path dynamically to extract environmental features from the way the robot is

moving while it is moving. The approach utilizes the robot's motion to perform dynamic landmark detection.

One of the most important stages of robot design is matching sensors to the task. The primary concern in designing a landmark detecting algorithm is the selection of landmarks which can be robustly and repeatedly detected with the given sensors. This led to the choice of walls and corridors as frequent landmarks in the office building environment. They are large enough to be reliably detected dynamically, as well as static and unlikely to disappear during the robot's traversal of the environment. In contrast to detailed environmental mapping **[Chatila and Laumond 85]** **[Moravec 88]**, the purpose of this robot is to explore and learn the large-scale structure of its environment.

The patterns traced by the robot correspond to the basic set of environmental features or detectable landmarks: walls, corridors, and messy areas. These are detectable through continuous monitoring of the compass and the lateral sonar transducers. The following simple heuristics are sufficient:

- If the robot is moving in the same direction for a while, it is probably following a straight boundary.
- If the robot is following a boundary, one (or both) of its side sonars will be receiving consistent returns within the edging distance threshold.
- If the robot is moving in a nearly straight line, its compass will remain constant.

These heuristics translate directly into simple rules for feature detection. A behavior is dedicated to constant monitoring of the compass direction for consistency. Since the data rate of the sonar is low, gathering multiple readings would be overly time consuming. Instead, the algorithm utilizes all of the gathered data and filters out the bad data through dynamic averaging. Consistent compass readings result in increased compass confidence. The sonar returns of the two sides of the robot are monitored simultaneously. If either is consistently within the edging distance, its confidence is increased as well. Finally, if the confidence measures fall below a minimum threshold, the confidence counters are reset.

Whenever sufficient compass confidence has accrued, left or right side consistency is checked. If both have grown simultaneously, a corridor is

detected, otherwise it is a wall. It is necessary to couple the compass data with the sonar instances. A consistent compass bearing can mean that the robot is moving in a straight line through the middle of a room, as it does if is started away from any objects in search of a boundary to trace. Conversely, a consistent object-following behavior without a constant compass bearing is merely detecting irregular boundaries. Detection of irregular, messy areas as landmarks is useful as it provides a link between the "real" landmarks. Any traversed path can be represented as a continuous sequence of the given landmark types. Path continuity is an important quality which is utilized in goal-directed navigation. While they are not necessarily a useful destination point, the messy area landmarks ensure that the robot's list of consecutive landmarks is continuous in space. This, in turn, allows it to optimize paths based on physical rather than only topological distance (see chapter 8).

In this scheme, a landmark corresponds to a hypothesis which has gained a high level of confidence. Toto forms hypotheses based on simple rules which rely on the side sonar readings and the compass values.

Consistent sensor readings appropriately increase and decrease the landmark confidence levels. When a confidence level for a landmark grows enough to reach a preset threshold, a landmark is acknowledged. The threshold levels were empirically chosen to be 10 for a wall, 8 for a corridor, and 15 for a messy area (lack of landmark). The corridor threshold is carefully chosen to be lower than the wall threshold in order to minimize the number of instances in which the confidence into one of the walls builds up much faster than the confidence in the other. Finally, the threshold for a messy area is selected to be comparatively high to minimize possibly overlooked walls and corridors. The confidence levels for individual types of landmarks are maintained in parallel, with independently active monitors.

Assuming constant velocity (20 centimeters/sec) and the data update rate of 0.83Hz (a full sonar ring every 1.2 seconds), we can approximate the distance required for each of the landmarks to be detected. Specifically, the minimum distance required for a wall is 2.4 meters, for a corridor 1.9 meters, and for a "junk" landmark 3.6 meters. Given frequent spurious errors in the sensor data, the confidence counter is often decremented to compensate, which results in an actual longer distance threshold for each landmark.

6.4 Performance

The robustness of the dynamic landmark detection scheme lies in its qualitative nature. Feature detection does not rely on exact positioning of the robot, nor on the accuracy of its sensors. The algorithm uses a running average to eliminate spurious errors due to sonar specularities or inconsistent compass shifts. It also allows for averaging out quick dynamic obstacles, such as people walking by; they appear as transient noise. Spurious errors have a very small effect since they only temporarily decrease the related confidence measurement, but do not reset it completely. The landmark detector does not expect either the world or the sensors to be perfect. The robot can recognize a landmark regardless of where along its length it may be. Additionally, it can recognize a landmark in spite of noise or changes in the smaller features in the environment. For instance, adding small and medium-sized furniture or clutter along a wall will not prevent Toto from recognizing it. Landmarks are detected with repeatable accuracy independent of the robot's starting position or the exact path followed.

Figures 6.1 and 6.2 illustrate Toto's landmark detection performance in a room and a corridor, respectively. Landmark labels (LW for left wall, RW for right wall, C for corridor, and J for junk or messy area) with compass bearings (0 through 15) indicate the location where each of the landmarks was detected. In the room environment the algorithm manifests a desirable clustering of landmarks and a consistency of the detected compass directions although the robot is not started at identical locations nor does it follow the same path in each of the trials. In the corridor the robot reliably detects the type of the landmark and its compass bearing, but the positions of exact detection vary over different trials. The mapping algorithms (see chapter 7) is designed to take advantage of landmark reliability and does not rely on its exact position of detection.

The robustness and reliability of the robot's lowest level boundary tracing behaviors allow for this simple, dynamic landmark detecting scheme which circumvents the need for explicit landmark models and matching algorithms.

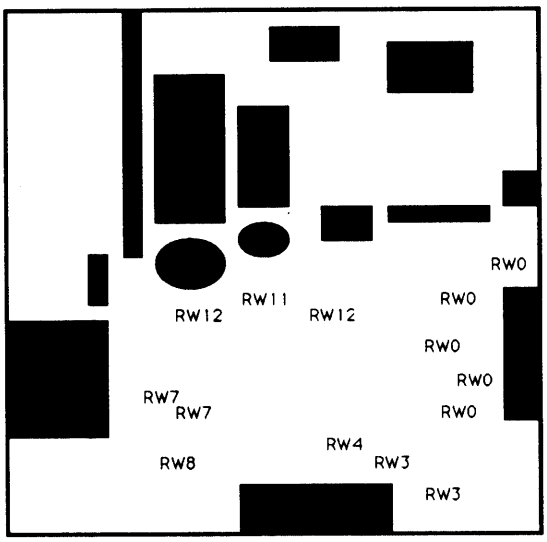


Figure 6.1: Toto's landmark detection performance over three trials in the same room. Each landmark consists of the type (e.g. RW = right wall) and the associated compass bearing. The indicated landmark locations correspond to the exact position of detection. The data show landmark clustering in spite of the lack of position control.

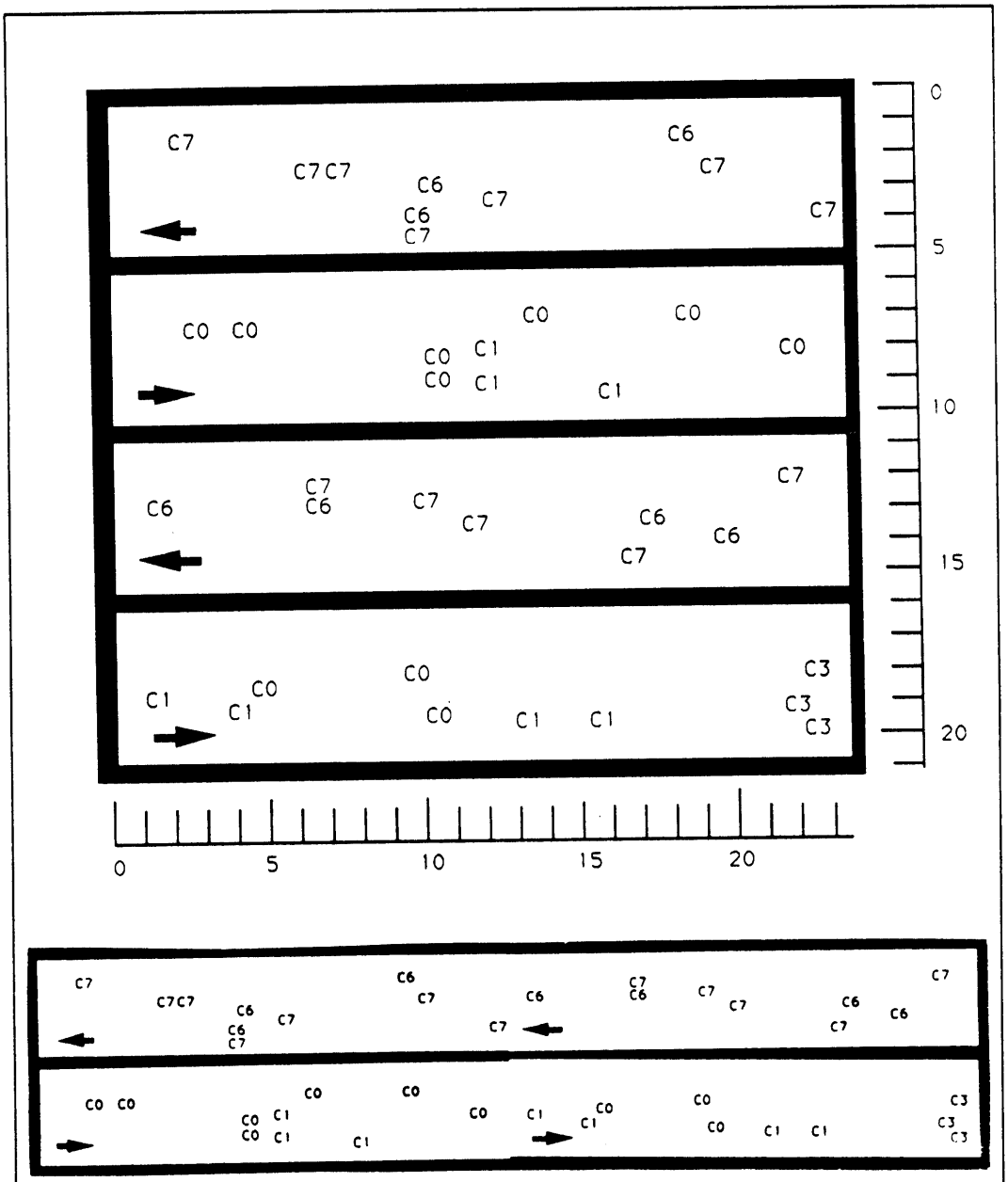


Figure 6.2: An example of Toto's landmark detection performance in four independent trials in a corridor. Even without position control, some clustering of landmarks is observed.

Chapter 7

Environment Learning

7.1 The Goals

In this project, the robot's task is to learn the large-space structure of the environment by recording its permanent features. The approach differs fundamentally from building a detailed map of the world which includes features of smaller size and higher probability of impermanence. Instead, the objective is to design both space-learning and goal-directed navigation algorithms which allow the robot to get within the sensing range of the goal. Its repertoire of behaviors can then be augmented with special purpose fine motion planning appropriate for the specific task and sensors used. This approach of large-scale navigation for approaching the goal with fine corrections when the goal can be sensed is also used by bees, homing pigeons, and even beach fleas **[Schöne 84]**.

In order to build a map of the environment, the robot must store the detected landmarks in some type of a representation which can be used for goal-directed navigation. Both the collision-free object-tracing and landmark detection algorithms described so far are qualitative. In the same vein, and for the same benefits, the map representation of the environment should be qualitative as well. Graphs provide a natural means for representing qualitative, topological relations, in contrast to metric-based, cartesian maps. Their structure implicitly contains adjacency properties between the nodes, information necessary for goal-directed navigation. As such, graphs have been used by **[Kuipers 79]**, **[Brooks 85]**, **[Chatila and Laumond 85]**, **[Elfes 86]**, etc. All of these applications use graphs as centralized data

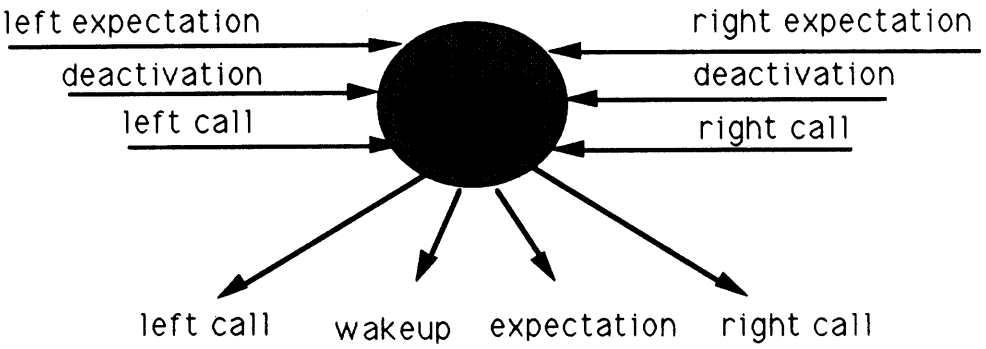


Figure 7.1: A schematic showing the communication links between the behaviors in the system. Each node receives expectation, deactivation and call inputs from both of its neighbors. Each node sends out bilateral expectation and call messages, and one-directional wakeup messages.

structures. In contrast, the approach presented in this thesis, utilizes fully distributed graphs.

Instead of a global data structure, the decentralized nature of the subsumption architecture itself is used to implement a distributed graph. Brooks suggested the use of behaviors (collections of AFSMs) as nodes in the graph. Equivalent to any other of the robot's behaviors (such as obstacle avoidance and object-tracing behaviors), each node is an independent process in the graph, which responds to certain inputs and generates appropriate outputs. Each node is a behavior which receives input from the landmark detector, the sonars, the compass, and the neighboring nodes in the graph. It outputs messages to its neighbors, as well as occasional directives to the base. Figure 7.1 illustrates the communication links between the behaviors in the system.

The behavior-based distributed world representation is a natural extension of the subsumption architecture. In contrast to global world models, the graph itself is not accessible as a whole, since each of its nodes is an independently acting behavior.

Due to the nature of the Behavior Language compiler, the topology of the graph must be statically determined at compile time. It is important that such a static topology be chosen properly so as to be as flexible as possible in order to accommodate the topology of the physical world (see chapter 9 for

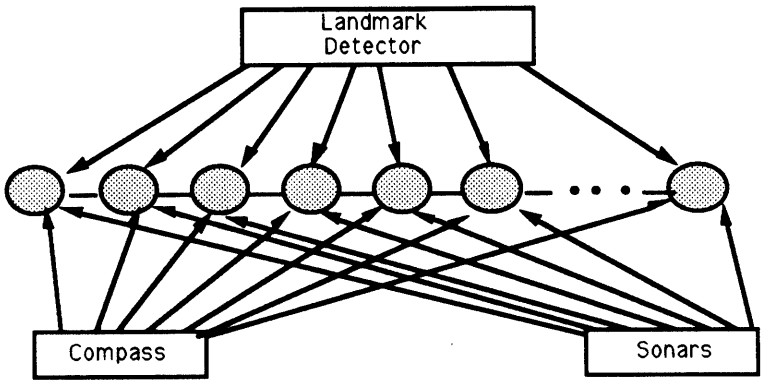


Figure 7.2: The structure of the graph connectivity. The landmark detector broadcasts the landmark to all nodes in the list. Each node has access to sonar and compass data.

a detailed discussion). The robot is initially given an underlying graph with "empty" nodes which are "filled" sequentially, as it explores its environment. The nodes are interconnected by message wires which allow communication between nearest neighbors in the graph. The graph connectivity is a space consideration. Arbitrary dynamically assignable connections between nodes can escalate to $O(n^2)$ connectivity and thus do not scale well. The presented approach employs only a few global broadcasting connections, in addition to nearest-neighbor connections between adjacent graph nodes (figure 7.2). The total number of connections is linear in the size of the graph.

The chosen graph topology should be capable of accommodating the space structure encountered in the physical world. Since the environment is not known *a priori*, the graph topology must be capable of embedding any possible physical organization. The first underlying graph topology which was explored was a linear list. It was chosen for its simplicity, but showed surprising functionality.

7.2 Spatial Learning Through Graph Construction

As the robot explores its environment, the landmarks it detects are broadcast to all the nodes in the graph. Initially, the graph is empty, and the first detected landmark is automatically stored in the first node. This node is special in that it knows it is the first. This is necessary since there is no global data structure which controls node allocation. Instead, each node "wakes up" its next unallocated neighbor in the list. The newly allocated node corresponds to the robot's current position in the graph. The qualitative descriptor of the landmark is saved, consisting of its type (left wall, right wall, corridor) and its associated averaged compass bearing. The newly added node receives a *wake up* call, and is *activated.*

Whenever a landmark is detected it is broadcast to all the nodes in the graph. When a node receives a landmark it compares it to itself. The matching is a simple process of comparing the landmark type and the compass bearing. Its simplicity is a result of the low-level navigation algorithm which guarantees that the robot will follow the outside edges of objects. This behavior generates only two possible directions along a single object boundary. The matching takes into account the duality of each landmark depending on the compass direction (e.g. a left wall going north is equivalent to a right wall going south). Since all graph nodes are matched in parallel, map localization effectively takes constant time, regardless of the size of the graph.

Given the sparse landmark set, more than one landmark may have the same label (type and bearing). The next section introduces a method for using context to disambiguate between similar landmarks in order to produce only a single match in every case.

After a landmark is broadcast if no graph node reports a match, the landmark is assumed to be new, and is to be added to the graph. This is accomplished by assigning the new landmark to the graph node adjacent to the currently active position. Thus the topological adjacency of the landmarks in the world is reflected in the graph. The active node sends a wake-up call to the neighbor who, upon receiving it becomes activated and spreads *deactivation* to its predecessor.

Figure 7.3 shows a sample environment with the indicated positions of landmark detection. The trace in figure 7.4 illustrates the process of path

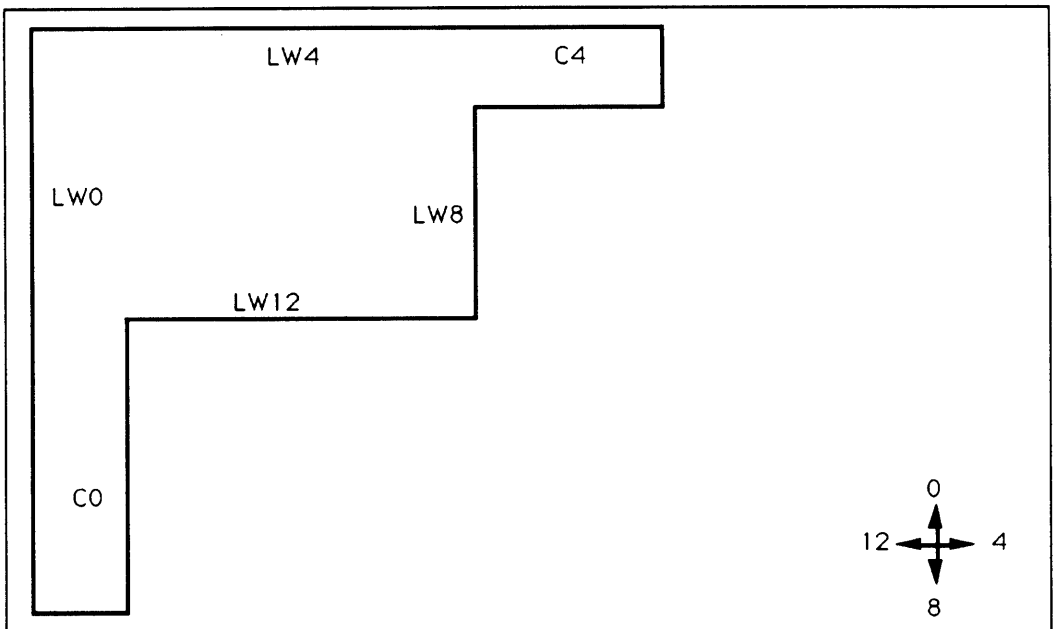


Figure 7.3: The large-space version of the ninth-floor area of the Lab. The landmark detection areas are marked with landmark types and compass bearings. The assumed exploration direction is arbitrarily chosen to be clockwise.

```
node#:  message:
--------------
0       woke-up
        corridor at bearing = 0
0       activated
1       woke-up
        leftwall at bearing = 0
0       deactivated
1       activated
2       woke-up
        leftwall at bearing = 4
1       deactivated
2       activated
3       woke-up
        corridor at bearing = 4
2       deactivated
3       activated
4       woke-up
        leftwall at bearing = 8
3       deactivated
4       activated
5       woke-up
        leftwall at bearing = 12
4       deactivated
5       activated
```

Figure 7.4: A trace of node allocation as the robot explores the shown environment. The left-hand column shows the node number, the right shows its activity.

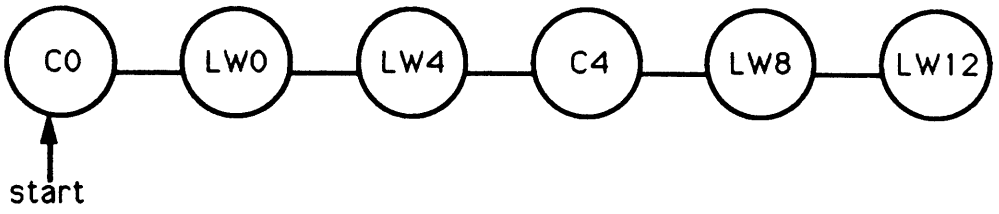


Figure 7.5: The distributed graph or map resulting from the robot's exploration of the shown environment.

learning through node allocation. The resulting graph is shown in figure 7.5.

A new node is allocated only when the landmark label changes. A physically long landmark will be detected repeatedly along its length, but will be prepresented by a single node in the graph. The advantages of this choice, along with possible alternatives, are discussed in the next chapter.

It is important to note that there is no guarantee that the active node is the last one in the graph having an unallocated neighbor on its right in the list. This situation requires a graph with higher than linear connectivity. A more flexible representation capable of handling any 2D physical topology is discussed in chapter 10.

Another more complicated scenario is one in which the path loops back on itself. Proceeding in either of the possible directions would lead the robot to another already discovered landmark. With a fixed linear-list topology, this forces a termination of the particular path. This is an inherent limitation of the topology. This issue is addressed in chapter 9, in which an augmented graph topology is described.

The process of learning unique landmarks is limited by the size of the graph. Once all of the nodes are allocated, the robot is forced to return to known territory.

An alternative approach to learning the environment is to supply the robot with a preconstructed graph representing the reachable world, or some portion thereof. The robot can localize within such a graph, verify it, and augment it through independent exploration. Many path planning systems rely on an *a priori* world map, while others employ a wandering, exploratory phase allowing the robot to construct a map based on its sensory information. While such a map is necessarily less accurate, it may provide a higher model matching probability since it is based on the input from the robot's own sensors. The difference between the two approaches may give rise to a somewhat different set of design concerns (e.g. accuracy in model construction versus the accuracy of localization), but they are fundamentally equivalent.

7.2.1 Using Expectation

Whenever a node in the graph is active, it spreads *expectation* to its neighbor in the direction of travel, thus alerting it to expect potentially upcoming activation. Whenever a landmark is matched to a node, and that node is expecting, the match is considered accurate. In a given environment (figure 7.6),

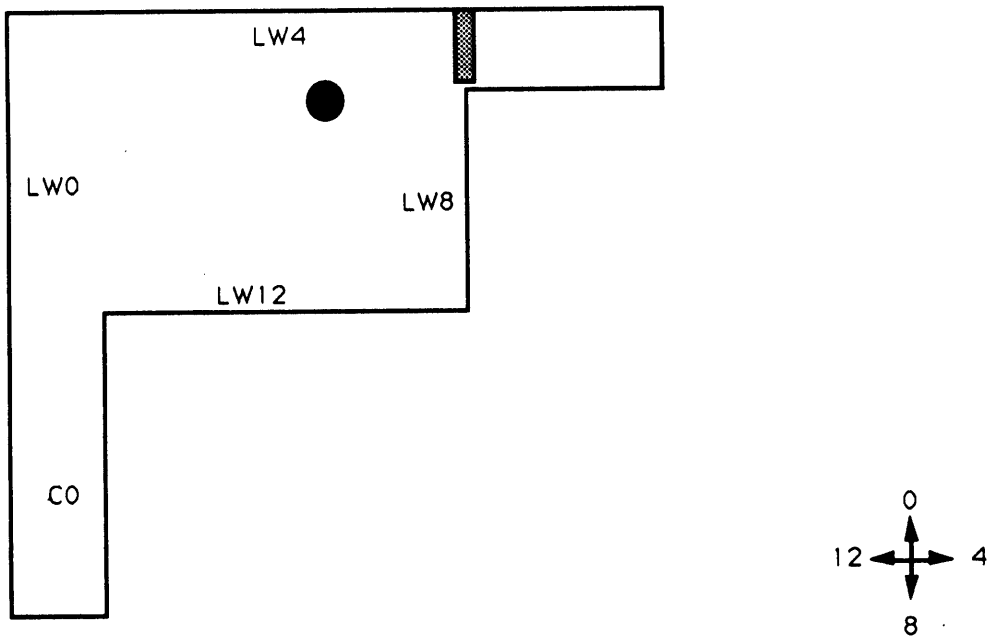


Figure 7.6: The ninth floor test environment with a blocked corridor. The areas of landmark detection are indicated, as is the position of the robot.

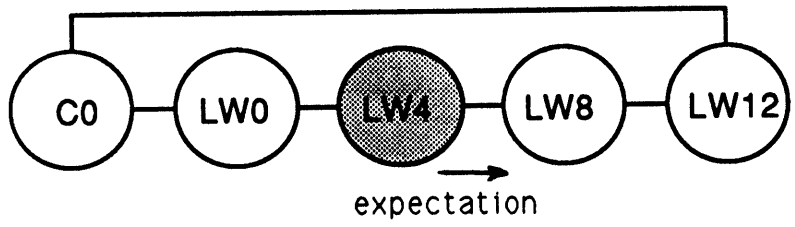


Figure 7.7: The graph of the explored environment showing the active (shaded) node corresponding to the robot's current location. The arrow indicated the direction of propagated expectation, based on the robot's current direction of travel along the landmark path.

without expectation the robot may match either of its adjacent landmarks (figure 7.7). Matching the correct one of those confirms the robot's position hypothesis as well as its confidence in the map's accuracy.

If a match occurs without expectation, it could either be false, or an indication that the path contains a loop. A simple linear list cannot handle loops in the graph. A more powerful graph representation is described in chapter 10. The algorithm deals implicitly with false positive and false negative matches by attempting to maximize the accuracy of each match through the use of context (expectation) as well as a rough position estimate.

7.2.2 Using Position Estimation

Given the small number of landmark types, it is necessary to employ an additional method of landmark disambiguation. In general, no totally position-independent method will be able to distinguish between two landmarks of the same qualitative type and compass bearing. Figure 7.9 shows an example of such a scenario. Brooks suggested obtaining a very coarse position estimate by integrating the compass bearing over time **[Mataric and Brooks 90]**. This estimate assumes constant velocity of the robot. Although extremely rough, it helps in disambiguating otherwise identical landmarks. Figure 7.8 shows the variation in the position estimate as the robot moves through the environment. The landmark matcher takes into account cumulative error, as well as the length of the particular landmark in setting the error margins. The method relies on the heuristic that two landmarks of the same type are unlikely to appear in close physical proximity; for example, two qualitatively identical left walls must be separated by a detectable space.

If a landmark matches, but is not expecting, its estimated position is compared to the robot's current rough position estimate. If the estimates match within some error margin, the path is assumed to have looped.

Finally, if the position estimate does not indicate a match, the match is assumed to be an error, and the robot proceeds with its exploration without updating its position.

Figure 7.10 shows an example of an ambiguous environment. Figure 7.11 is a trace of the code execution without the use of position referencing. The robot fails to recognize a previously known location due to the lack of expectation. Instead, it adds a new node to the graph, as shown in figure 7.12.

Figure 7.13 is a trace of the execution utilizing position information. Al-

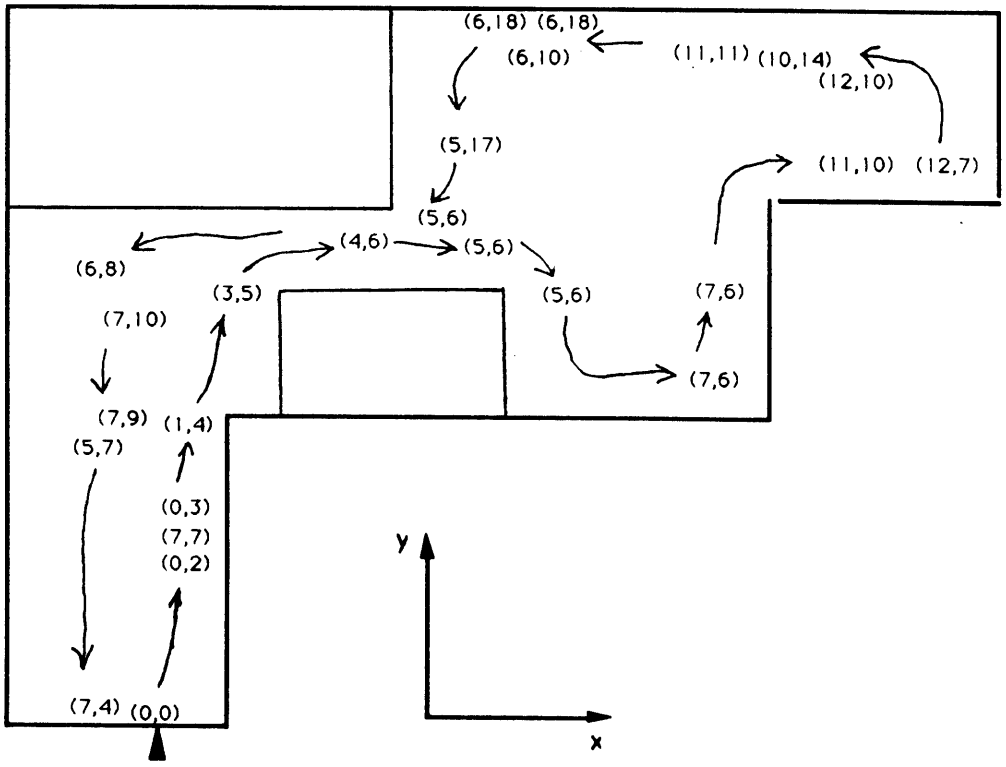


Figure 7.8: The position data shown here illustrate the roughness of the position estimate, and its cumulative error over time. The data were gathered by running the robot continuously through the environment and sampling the position estimate at the marked location. The (x,y) coordinate pairs are indicated in 0.5-meter units, but are only used relative to each other, rather than as absolute distances.

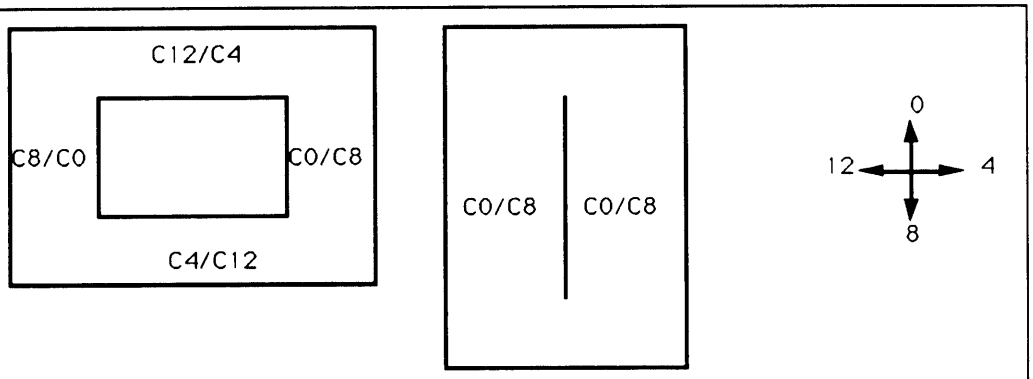


Figure 7.9: Given the small number of distinct landmark types, many locations appear indistinguishable, and necessitate the use of context or history, as well as position, in order to disambiguate. In the environments shown above, the robot must use the position estimate in order to localize correctly.

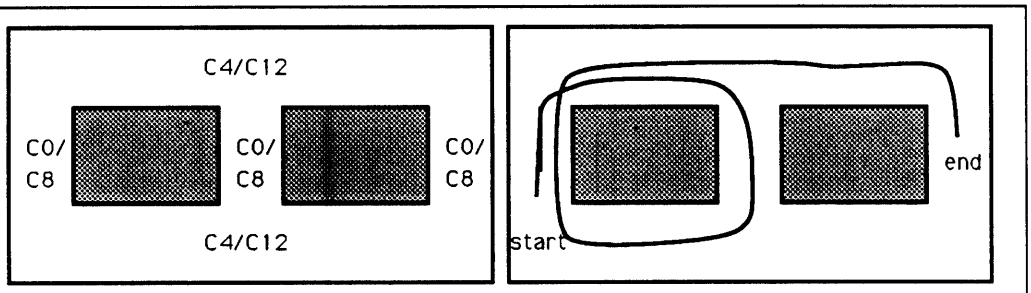


Figure 7.10: In the environment shown above, the robot must disambiguate between similar landmarks. The path it takes eventually leads it to the third corridor, marked as the end location, which appears identical to the second. Without the use of position information, the context alone results in an incorrect localization.

```
node#:  message:
---------------
...
        corridor at bearing = 12
1       activated
0       deactivated
        corridor at bearing = 8
2       activated
1       deactivated
```

Figure 7.11: The execution trace shows the robot localizing incorrectly based only on contextual information. Without position information, it cannot distinguish between the previously learned and newly discovered landmark.

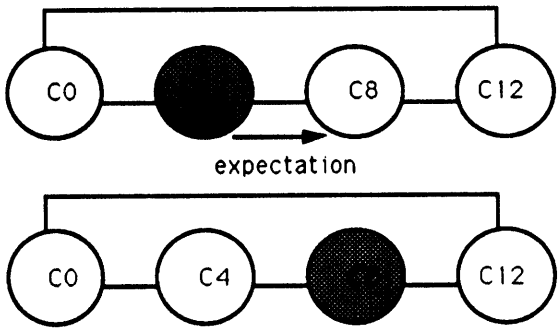


Figure 7.12: The graph showing illustrates how contextual information in the form of expectation leads the robot to localize incorrectly. The expecting node happens to be identical to the newly discovered landmark, so the robot cannot distinguish them.

```
node#:  message:
---------------
...
        corridor at bearing = 12
1       activated
0       deactivated
4       woke up
        corridor at bearing = 8
1       deactivated
4       activated
```

Figure 7.13: Execution trace illustrating correct localization after position information is taken into account. Although the two landmarks still appear identical, and the context clue is incorrect, the position estimate differentiates them. The robot recognizes the location as a new landmark, and adds it to the graph.

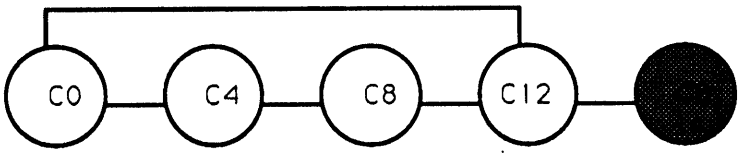


Figure 7.14: After using the position information to differentiate between two otherwise identical landmarks, the robot adds the new node to the graph.

though there is no expectation, the position match is close enough, and the robot localizes properly. Figure 7.14 shows the updated location of activation.

7.3 Summary

The described environment-learning algorithm uses a qualitative, topological representation of the world. It stores detected landmarks into a graph consisting of concurrently active behaviors as nodes. Topologically adjacent nodes in the graph communicate through message passing, which allows for spreading activation and deactivation through the graph. The landmark corresponding to the robot's current position is active and laterally inhibits the others by propagating deactivation to its predecessor.

Upon discovery, landmarks are broadcast to all nodes in the list in parallel. Concurrently active locations allow for graph localization in constant time. The notion of expectation and a rough position estimate are used for landmark disambiguation in order to facilitate localization. If no node matches, the location is assumed to be new. The currently active node sends a wake-up call to its neighbor which becomes activated and associated with the newly discovered landmark.

The orthogonality of the office environment structure, coupled with the chosen landmark set proved empirically robust. The incremental method for landmark detection using dynamic averaging limited large sensory errors which would have caused false positive landmark matches. The matching accuracy was further enhanced through the use of expectation and the rough position estimate. False negatives occurred if the robot failed to recognize a landmark as it was exploring. If the length of the landmark was sufficiently large, most matches eventually occurred. Otherwise, the robot failed to update its position estimate. While it never matched two landmarks incorrectly, it occasionally failed to detect a landmark. Such an occurrence during a discovery phase resulted in a sparser map which would later get augmented if the robot was allowed repeated runs through the same environment. Skipping a landmark on the way to the goal did not affect the goal-finding behavior. This resulted from the design of the path finding algorithm, described in the next section.

Chapter 8

Goal-Oriented Navigation

The existence of the world model allows the robot to return to an arbitrary landmark in the known world. On Toto, the goal landmarks is selected by pressing a combination of three buttons on the top of the robot. The buttons allow for selecting a particular landmark type (e.g. the nearest wall or corridor), the first discovered landmark (the first node in the graph), or an arbitrary landmark in the graph.

Since the graph structure is distributed, there is no notion of a global path to the goal. Global path planning in a decentralized graph must be accomplished with only local communication. The approach described here is based on the same process of message passing as that used for all communication within the graph [**Mataric 90**].

The algorithm is based on the concept of *spreading of activation* as used in semantic nets [**Quillian 69**]. In a semantic network, finding the relation between two concepts is accomplished by spreading activation from the two nodes in the network, and waiting for the two waves to intersect. This is equivalent to graph search.

In the algorithm used for locating a path in the spatial network, activation is spread in one direction only, starting from the goal. It propagates through the graph and eventually reaches the node corresponding to the robot's current position.

The node in the graph matching the goal landmark location repeatedly sends out a *call* which is propagated until it reaches the currently active node. The direction from which the incoming call arrives is the desired direction of motion. Pursuing it will lead the robot toward the next landmark on the

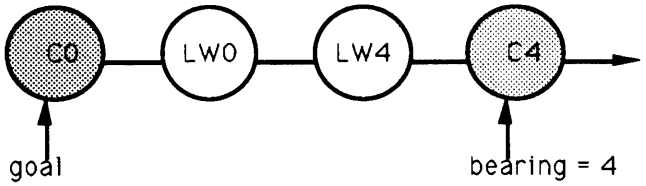


Figure 8.1: The lightly-shaded node indicates the robot's current position. The darkly-shaded node is the goal. The arrow indicates the robot's direction of travel along the landmark path. The shown graph is an example of a position which forces the robot to change direction in order to pursue the path to the goal node.

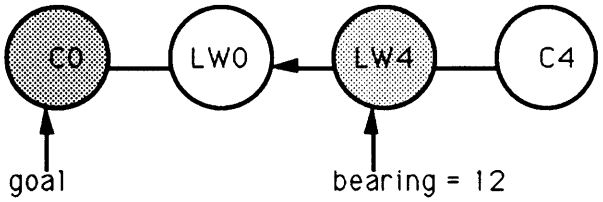


Figure 8.2: The lightly-shaded node indicates the robot's current position. The darkly-shaded node is the goal. The arrow indicates the robot's direction of travel along the landmark path. The shown graph is an example of a position from which the robot is already heading the proper direction in order to reach the goal node.

path to the goal.

Due to the design of the object-tracing behavior, the robot is always moving in one of two possible directions along the list of landmarks: left-to-right, or right-to-left. Each new landmark is added to the end of list, so left-to-right is the direction of initial exploration, by definition. This ordering implicitly preserves the direction of original exploration. If the robot is moving in the exploration direction, and it receives a call from the left, it must turn around, as illustrated in figure 8.1. A call from the right requires no turn.

Analogously, if the robot is moving right-to-left, and it receives a call from the right, it must turn around. No turn is required from a left call (figure 8.2).

Since the goal node emits the call repeatedly, the robot receives it at each of the landmarks it reaches. Consequently, it can chose the proper direction to pursue from any point in the graph. If it veers away from the path and arrives at an arbitrary node, it will resume its mission from that point in the graph, until it reaches the goal. When the currently recognized landmark matches the goal landmark, the goal node is reached and the call is terminated.

8.1 Finding the Shortest Topological Path

Once a node is selected as a *destination*, it begins to send out a *call* to both of its neighbors. The call is sent out continuously until the robot reaches the goal. Whenever a call is received by an active node in the graph, it is propagated on to its neighbors in the appropriate direction. If a call was received from the left it is passed on to the right, and vice versa. Eventually, the call must reach the currently active node in the graph.

Since there is no global data structure, there is also no global notion of a continuous path. Instead, the robot knows the locally correct direction to pursue. Since the destination emits its call continuously, all locations in the graph receive it and are provided with the correct direction toward the current goal. In this scheme there is no need for replanning if the robot strays off the desired path or becomes lost.

If the robot follows the landmarks in the direction of the incoming call, it is guaranteed to proceed on a shortest topological path to the goal. If the

calls are sent from two or more different nodes in the graph, the one closest to the current position will reach it first, given uniform activation dissipation.

Since calls are emanated continuously, some method must be employed in order to disambiguate between previous calls from distant locations reaching a node at the same time as more current calls from closer locations. This can be accomplished by time-stamping each call and always choosing the newest one.

8.2 Finding the Shortest Physical Path

Since spreading of activation is equivalent to graph search, the algorithm produces the shortest topological path in time linear in the size of the graph. However, the topological path is not necessarily the shortest known physical path. To obtain the shortest path in terms of physical distance, the notion of *time-as-distance* is used. As the robot traverses the world, it builds up confidences in certain landmarks. Confidences are thresholds which correspond to time periods (assuming constant velocity). This length can be used to estimate the size of each landmark.

Each landmark is detected continuously, and the number of consecutive times the same landmark label is matched corresponds to its rough physical size. For example, a corridor has an implicit length imposed by its detection threshold. Assume the threshold for detecting a corridor takes m seconds. Assuming continuous velocity v, the approximate length of a once-detected corridor is mv feet. If some corridor is detected c times consecutively, it is estimated to be about cmv feet long.

In the actual implementation, the length descriptor is represented in the confidence units since they are equivalent to distance. Analogously, these are also units of time, so the map thus contains an implicit representation of time as well.

The rough length estimate is stored in each node as an additional landmark descriptor. To compute the length of a path, the goal-call originating from a node grows from unit length into an estimate of the path's physical length. Whenever the call reaches a node in the graph, its length is added to its value. As the call is propagated along the wires, its size grows gradually. When it reaches the node corresponding to the robot's current position, its value represents the length of the path it traversed.

Depending on the fanout of the node, the robot can receive a number of calls from different directions. The call with the smallest length estimate corresponds to the direction on the physically shortest path. The direction of the incoming shortest path is the direction the robot pursues. Since the call-propagation process is continuous, the process of selecting the shortest path is repeated at each node the robot traverses. A greedy choice at each step guarantees the optimal solution. This can also be viewed as gradient descent on path length at each node.

In addition to providing the path length, the value of the call serves as its name as well, and distinguishes it from other incoming calls. This obviates the need for time-stamps. The length is not a unique path identifier, however, since two paths with equal length are indistinguishable. This issues does not arise in a linear list, but does in the more general cyclic graph representation described in chapter 10. However, it is not a problem since one is not more optimal than the other. The algorithm uses a greedy strategy of always choosing the shortest path, thus eliminating any circuitous routes resulting from undistinguished calls.

If we view the path planning process as graph search, then topological shortest path is a parallel search in a graph where all links have unit length. Physical shortest path uses a graph with weighted edges where the weights correspond to the length of each landmark on the path. In both cases we can obtain the shortest path in $O(n)$ where n is the number of landmarks in the graph.

Chapter 10 shows performance examples of the path optimization algorithm in a variety of trials.

8.3 Summary

Spreading of activation through the distributed graph provides a linear-time path planning algorithm. Equivalent to graph search, the process yields the shortest topological path to the goal, from any node in the graph. Accumulating landmark lengths produces a true shortest path. The robot receives motion directives at each landmark, and traverses the optimal path without a global view of the world or a notion of a global goal.

Chapter 9

Choosing the Right Network Topology

9.1 A Review of Related Network Topologies

The purpose of the world representation is to provide a means for storing and accessing learned facts about the world. Ideally, the map representation should be able to represent any possible real world configuration, i.e. accommodate all possible worlds.

The difficulty arises from the fact that the representation is fixed *a priori*, instead of being generated at run time to match the world topology. Limiting the connectivity of the graph to linear growth is valuable for silicon implementations [**Brooks 87**]. If the number of connections is squared, or, worse yet, exponential, its physical implementation is not realistic. The key question then is to select a network topology which will allow for embedding the physical world topology, without knowing the latter *a priori*.

Having implemented and tested a linear list graph representation, we can easily assess its limitations. While suitable for continuous paths, it is insufficient for sequences which loop back on themselves. In the general case, when viewed as a undirected graph, a linear list limits the outdegree of each node to 2.

A natural extension of such a one-dimensional list is a two-dimensional grid of nodes. Such a grid can be viewed as a Cartesian map, especially

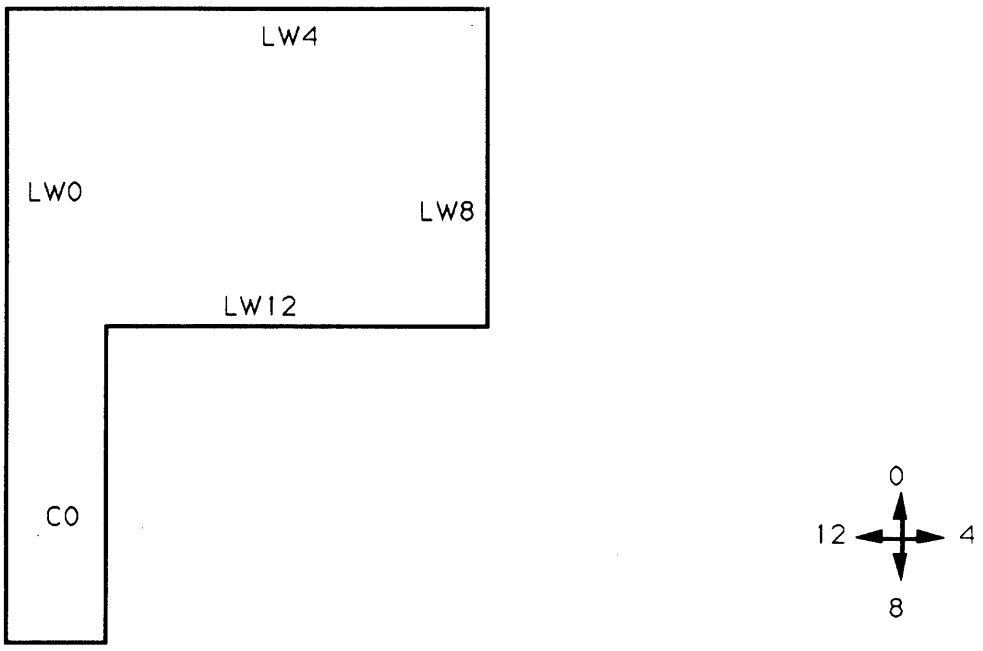


Figure 9.1: An example of an environment which cannot be properly represented on a two-dimensional grid without the use of metric information.

if distance and direction information is incorporated into the grid cells. In a 2-D grid, the outdegree of each node is 4, and the directions of the arcs can be naturally associated with four compass directions. Such a mapping appears attractive at first sight. It allows for simple node allocation through the use of compass bearing. For simple environments this results in graphs neatly matching the topology of the world. However, the rigid organization of the grid imposes several limitations.

- The outdegree of each node is limited to 4, and if additional connections are required, the same problem of fixed connectivity arises as in the one-dimensional case.

- If the grid is preserving purely qualitative information, i.e. the nodes are delimited by topological distance rather than geometric or temporal distance, then many possible physical arrangements will result in

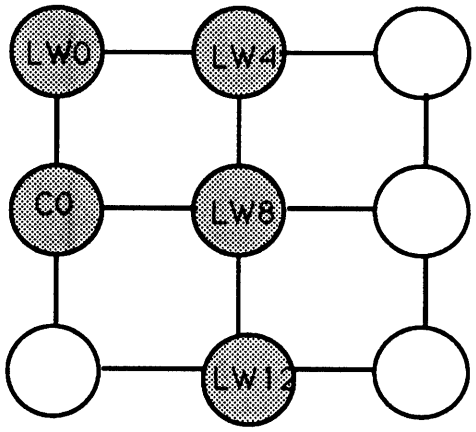


Figure 9.2: A straightforward but incorrect grid embedding of the example environment. The relative lengths of the landmarks result in an incorrect cycle in the graph. Two landmarks which are not topologically adjacent are allocated as neighbors due to the geometrical restrictions of the non-metric grid.

a contorted map. Consider, for instance, the environment shown in figure 9.1. The topological organization of landmarks is not representative of their physical organization. Consequently, our simple method of node allocation results in an unbalanced graph, as illustrated in figure 9.2.

- A simple solution to the topologically-limited information is to allocate a new node whenever a landmark is detected. In this way, the number of identical, adjacent nodes representing one landmark would also represent its length through time of discovery. Such a map corresponding to the environment shown in figure 9.1 is shown in figure 9.3. This scheme is undesirable because it requires a large network. Additionally, it is equivalent to a cartesian map with very rough metric information, but is less effective due to the limited outdegree of each node.

Assuming a sufficiently large outdegree, especially when coupled with arbitrary compass direction assignments, makes the two-dimensional grid a sufficient representation. The problem of topologically-limited information

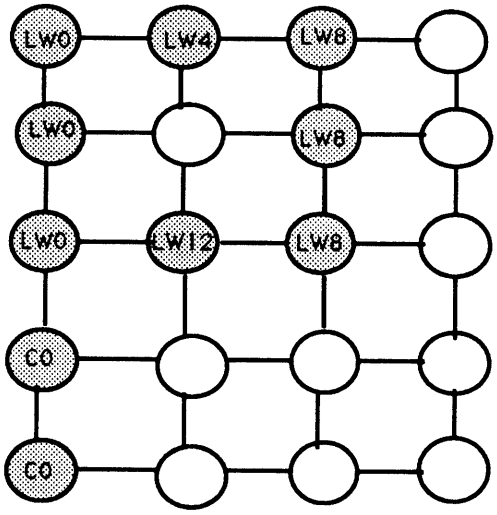


Figure 9.3: Utilizing metric information about the relative lengths of the detected landmarks allows for generating a more correct grid representation. Each node corresponds to a specific landmark length, thus resulting an a near-cartesian mapping of the explored space.

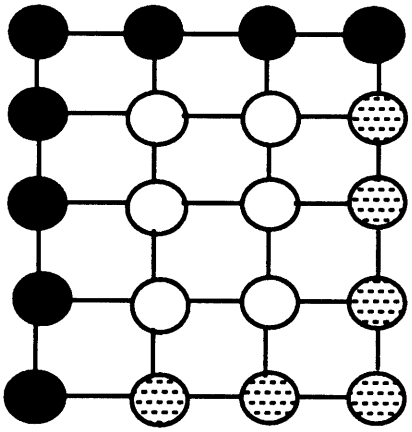


Figure 9.4: A fixed two-dimensional grid imposes an unnaturally rigid spatial mapping. Topologically adjacent landmarks may not be physically adjacent in the grid. Connecting them may involve going through many intermediate nodes. This results in a segmented graph with wasted nodes. In this figure, the intermediate nodes are shaded, and the white nodes constitute a now-isolated segment of the graph.

remains. Clearly, it is necessary to find a way to efficiently route connections between non-adjacent nodes in the grid. A simple activation spreading search, analogous to the technique used for goal-directed navigation, can be employed here as well. The resulting path will connect the two nodes, but will divide the graph into two regions. Figure 9.4 illustrates such a scenario. Subsequent connections between nodes in the different regions will be difficult if not impossible.

We can conclude that routing connections and "true" network connections should be isolated. A possible solution is a 2.5-dimensional grid. In such a grid the substrate, or the bottom layer, contains the landmark nodes and adjacent-neighbor links, while the top layer is used for routing. The two layers are connected with vertical links. Figure 9.5 shows an example of such a topology.

Recall that the purpose of the network is to propagate goal-directed navigation activation and expectation. The algorithm utilizes the number of the traversed nodes, as well as the distances they contain (represented in terms of traversal time), to compute the path. The search time in the described

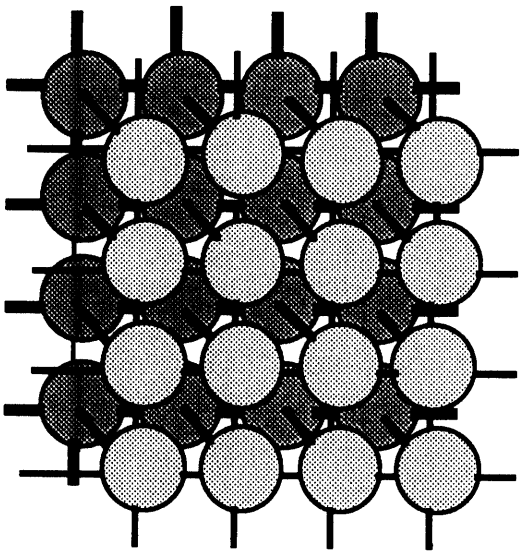


Figure 9.5: A example of a 2.5-dimensional grid: the bottom layer consists of landmark nodes, while the top layer is used for routing in order to avoid graph segmentation.

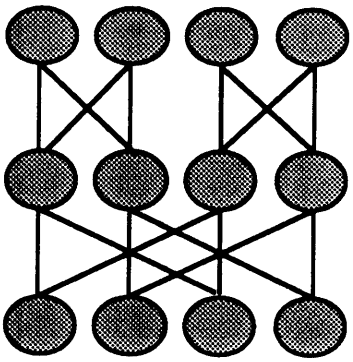


Figure 9.6: A butterfly network.

network would still be linear, but would require a delay period in order to allow all the possible paths to reach the destination node.

Unfortunately, this network topology does not provide a better solution to graph segmentation. The graph gets equally segmented, but the process is moved to the second, routing layer of the grid. It simply provides a larger space in which to lay down the routing links.

Other possible topologies include hypercubes and various higher dimensional alternatives. While the increased dimensionality decreases the graph segmentation problem, it also increases the routing algorithm complexity. Additionally, the number of nodes quickly becomes prohibitively large for higher dimensions.

9.1.1 Network Theory

We now consider a related problem in graph theory. In computer networks of interconnected processors, it is important to be able to route signals between arbitrary nodes so as to minimize the number of wire conflicts. The network topology, together with the routing algorithm, attempts to maximize the number of possible parallel connections. The butterfly is an example of such an architecture (figure 9.6). The routing algorithm is proven to provide a non-conflicting path from any node to any other in the list. Other examples of routing topologies include trees and tree meshes (figure 9.7). The latter two are not suitable for the routing purposes since they do not optimize the number of arbitrary parallel connections between nodes. Once a routing wire

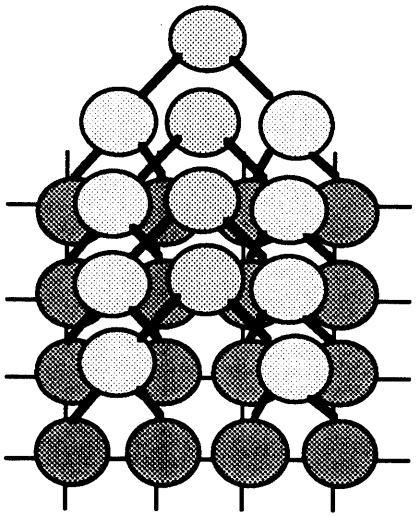


Figure 9.7: A tree-based network.

is chosen, it invalidates the entire parent tree.

While some of these data structures may appear appropriate, they attempt to optimize a different metric than that being explored for the purpose of map construction. The goal of the data structure is to avoid having to reuse links connecting the nodes, thus allowing for different signals to propagate between nodes in the graph at the same time. In contrast to map building purposes, they are not concerned with the number of reused nodes in the network.

The number of nodes in the butterfly grows as the log of the size of the basic list. The same is true for grid meshes. The butterfly is an example of a network topology which, combined with a clever routing algorithm, can produce a provably high inter-node connectivity. Unfortunately, it requires a large number of nodes and connections as overhead. The problem of having too many nodes can be simplified by using dummy routing nodes (as in 2.5 dimensional grids), but the approach is still not suitable for the task. The objective is to use a clever topology so that the routing algorithm can be very simple. This is important since the routing is executed continuously during the algorithm execution.

9.2 The Return to Linearity

Having considered various higher-dimensional topologies, as well as a few network theory solutions to routing, we return to the actual constraints of the problem. What the map-graph really consists of is a collection of paths, with occasional higher-outdegree junction nodes. The two main problems are: 1) the unknown outdegree of the junction nodes, 2) the inability to make arbitrary connections between nodes.

A simple solution uses an undirected, cyclic graph representation which provides a universal embedding for any physically feasible 2D topology. Given the structured nature of the office environment, dense junctions are expected to be sparse, and the environment is easily represented as a collection of linear path segments with occasional higher degree junctions. A series of paths can be represented with a single linear list in which path segments are appended to each other consecutively as they are discovered, with appropriate topological connections. Consequently, we can always represent a graph as a list with cycles and some inactive connections. Figure 9.8 shows an example of this embedding with an environment and its graph representation. The inactive links are represented with dashed lines.

What is needed next is a way to make connections between arbitrary nodes, as their topological adjacency is discovered in the world. The most difficult problem is that of dynamic node linking. A possible solution is setting up a fully connected graph at the time of compilation, and selectively activating the appropriate links. However, this solution is unrealistic for large networks, as the number or links grows as the square of the number of nodes. What is needed is a method of implementing a static topology, with fewer connections, which is capable of simulating dynamic links at run time. This can be implemented with a table-lookup method which serves as a switchboard between nodes. It learns which nodes to connect, and routes signals appropriately. The next chapter describes such an implementation.

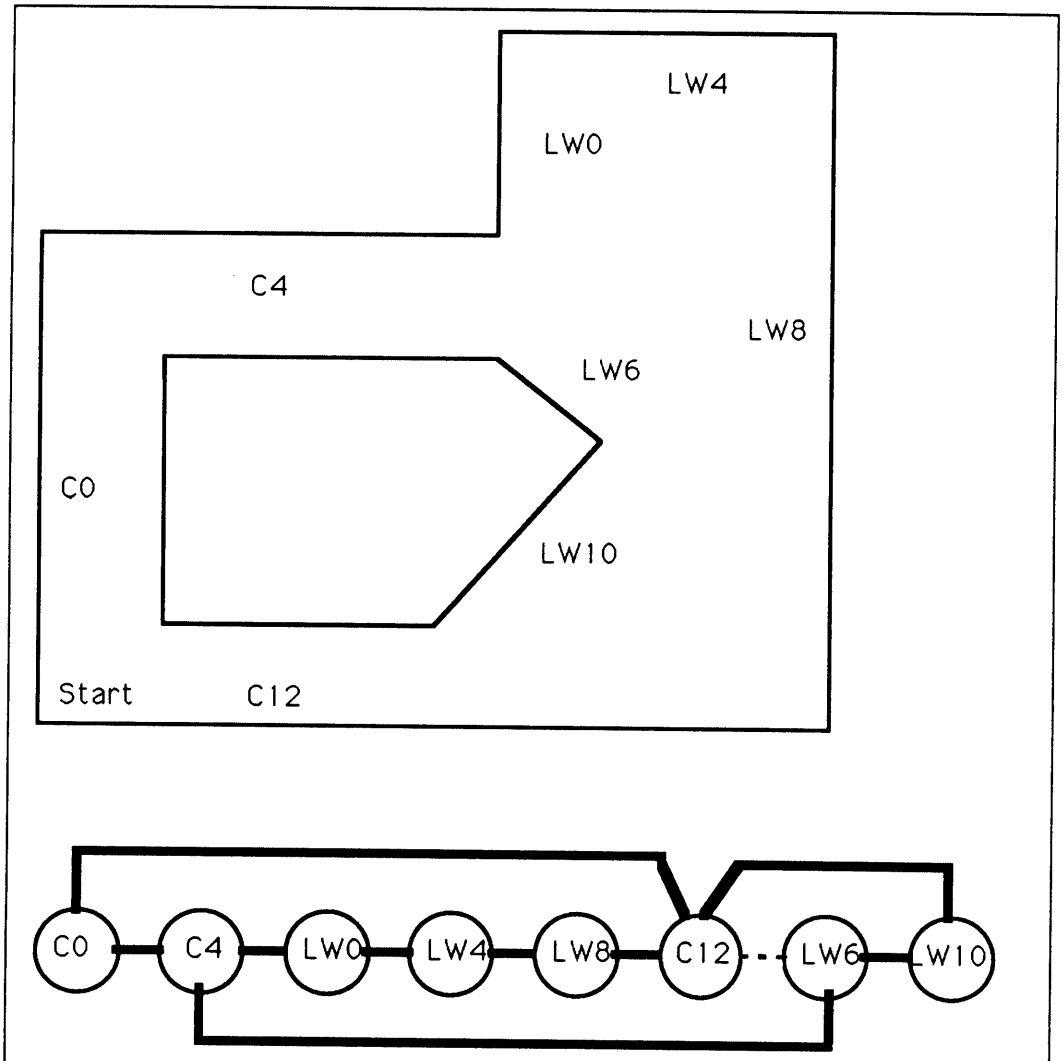


Figure 9.8: An example of an environment and the associated graph representation. The basis of the graph is still a list, but arbitrary internode connections are possible. This dynamic graph allows for embedding any two-dimensional physical topology. In order to preserve a continuous list structure, paths are appended together. Active topological links are indicated with full lines, inactive ones are dashed.

Chapter 10

The Dynamic Graph Topology

In the context of the task of landmark-based graph construction, the most desirable graph topology is a linear list with dynamic links (see chapter 9 for a detailed analysis). It is well suited for representing sets of interconnected linear lists, which is the defining topology of our environments (see chapter 7). This solution guarantees that any physically possible topology can be embedded into the given graph representation. Additionally, it eliminates the need for having two or more types of nodes in the graph, such as landmark nodes and routing nodes. The homogeneity of the graph can be preserved at no added cost since the graph nodes do not need to have any additional routing information. (Recall that adding complex routing information into each node is costly in terms of both space and computation.) Finally, this dynamic list-based network organization does not require much modification of the map-learning and goal-directed navigation algorithms described in the earlier sections.

10.1 The Graph Structure

The desired graph structure is an adaptation of the previously described linear list. A switching mechanism is added which allows connections between any two nodes in the graph. At compile time, each of the nodes in the list is bidirectionally linked to the switchboard (figure 10.1). These links allow for direct message passing between each node and the switchboard. Consequently, any two nodes can communicate with each other through the switch.

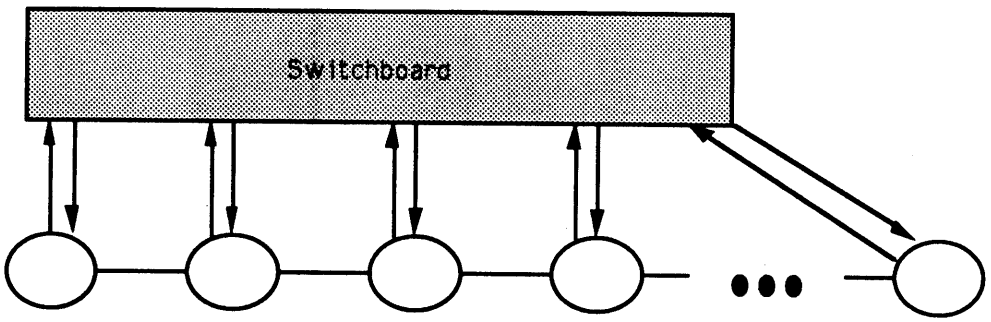


Figure 10.1: A schematic of switchboard interconnections. Each node in the graph has a bidirectional link with the switchboard. Consequently, and two nodes can communicate by sending and receiving messages through the switchboard. This obviates the need for linear links, but they are preserved for speed of processing, as well as for eliminating some side effects of cycles.

A crucial property of the chosen landmark set is that it represents the office environment (or any environment with orthogonal wall and corridor structures) as a set of linear paths with few intersections with higher degree nodes. Cross-connections between nodes in the graph are expected to be sparse. We exploit this property by preventing the switchboard from providing a full crossbar. Instead, we limit the connectivity to $O(n)$ by bounding the outdegree of each graph node.

The topology of the office environment coupled with the robot's boundary tracing behavior results in no more than four possible directions from any decision point. Consequently, the fanout of each node was fixed to four in all experimental trials by allowing each slot in the switchboard to accommodate up to 4 inputs. This bound was empirically proven to be sufficient.

Switchboard connections are established in the following way: as a connection is discovered between two non-adjacent nodes, an "entry" is made in its switch slot denoting a new connection. Subsequently, whenever a node sends out a message to its left and right neighbors, it also sends it to the switchboard. The switchboard then consults the table of slots to determine if there are any "jumper" links between the source node and any other nodes in the graph. If so, it passes the message on to all such nodes.

The process of establishing jumper links is facilitated by alerting the

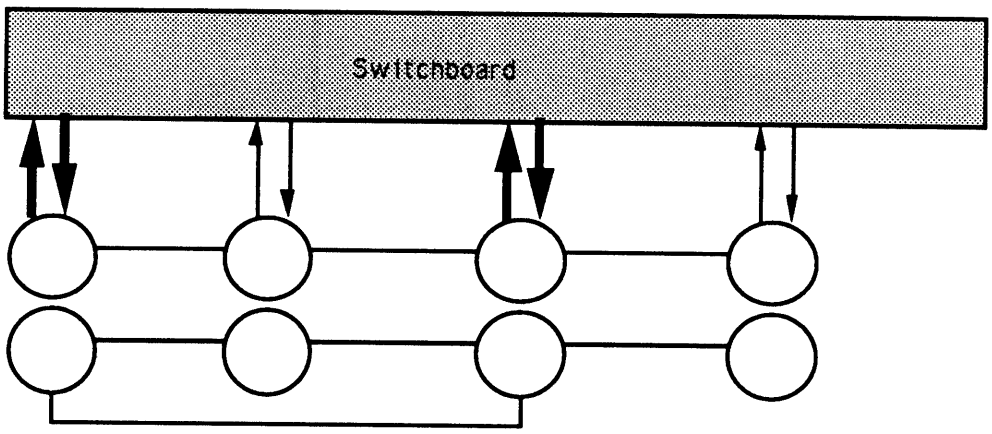


Figure 10.2: The process of generating a jumper: the two nodes to be connected send messages to the switchboard, which establishes a new link. The resulting graph has a virtual jumper link through the switchboard.

switchboard which nodes to connect. The switchboard keeps track of the currently active node. (This is easily implemented by sending a "named" message to the switchboard as a node becomes activated.) When a jumper is to be established, the new "neighbor" sends a message to the switchboard, which connects it to the currently active node by making slot entries for both the new node and the currently active node. Each such jumper link is bidirectional, allowing message propagation as if the two nodes were physically adjacent in the list. Figure 10.2 illustrates the process of setting up a jumper, and the resulting graph.

The presence of the switchboard obviates the need for nearest neighbor connections. However, preserving those connections serves as a method of more efficient message passing, considering that the map consists of interconnected sequences of paths, which are linear lists.

Additionally, taking advantage of the linear list organization of the graph eliminates some undesirable side effects of cycles, as described below. Instead of using the switchboard to make links between all nodes, it serves as a special router for jumper links only.

10.2 The Switchboard Algorithm

The switchboard introduces some interesting issues into the map-learning and goal-oriented navigation algorithms. Its effects on each of the types of network communication message are examined in turn:

Deactivation

The purpose of deactivation is to provide lateral inhibition throughout the network. The ordering of nodes affected by the spreading wave of deactivation is not critical, so it can be performed linearly, without utilizing the switchboard.

A convenient side effect of spreading information linearly is the guarantee that it will not return to the sender. This is particularly useful in the case of deactivation which could turn off the current node by propagating along a cycle through the switchboard.

In general, using linear propagation is useful due to its simplicity. The timing of linear message passing is not critical since the period required for detecting another landmark is orders of magnitude longer than linear message propagation for all but extremely large networks.

Goal-Calls

Goal-calls are the method of spreading direction information for goal-directed navigation, as well as for propagation of traversed path length. Calls utilize the topological properties of the graph by relying on message propagation in the order of topological adjacency, so as to preserve the correct metric of path length. To facilitate this process, they are routed through the switchboard.

Expectation

Expectation is used to alert the neighboring nodes of the possible upcoming activation. This method is used for map verification, as well as intelligent matching. Like the call, expectation utilizes the topological adjacency relations in the graph, and is therefore routed through the switchboard as well.

Wake-up Calls

In the simple linear representation, wake-up calls determined which node is to learn the currently discovered landmark. Since the switchboard allows

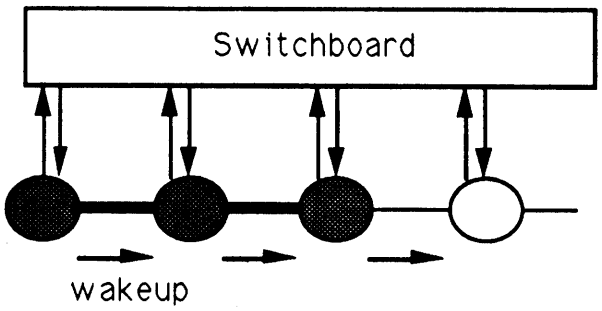


Figure 10.3: A sample scenario showing the start of a new branch in a path. The leftmost node has found another topological neighbor, and its fanout must be increased through the switchboard. It sends out a wakeup, which is propagated linearly to the first free node in the list.

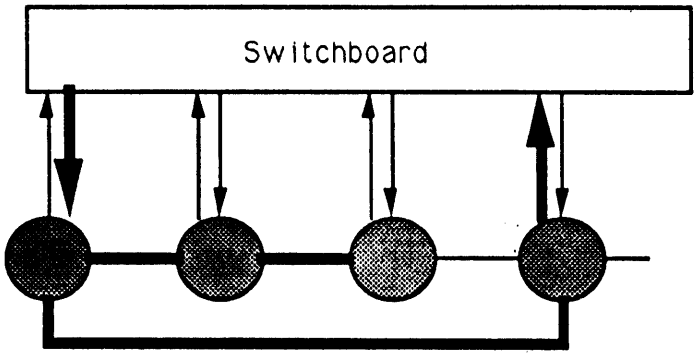


Figure 10.4: When the wakeup call reaches a node it uses its value to determine who to connect to. If the wakeup is a unit value, then it arrived from the nearest neighbor. Otherwise, it originated from farther away and indicates a jumper link through the switchboard. The new node sends a message to the switchboard, which makes the appropriate between the node and the wakeup originator.

for higher outdegree junctions at each node, wake-up calls will need to be sent farther than just to the nearest neighbor. More specifically, the wake-up call which marks the beginning of a new path will have to be propagated to the end of the list, until the first unallocated node is found (figure 10.3). That node will be awaken, but it will not have an active connection to its nearest neighbor on the left. Instead, it will have a jumper to its predecessor node which appears earlier in the graph (figure 10.4).

Consequently, propagation of wake-ups does not require the use of a switchboard. However, establishing a jumper between the new node and its predecessor does.

This demonstrates the two uses of the switchboard. The first is establishing contacts between two nodes, or growing a link. The second is using such links to propagate calls and expectation.

10.3 Direction Preservation in Path Planning

Part of the appeal of the simple linear list representation was its straightforward direction determination algorithm: a binary choice based on the direction of the incoming goal-call (see chapter 8). Depending on the robot's current direction, at most decision points it must only decide whether to proceed (it is already going in the correct direction) or to turn around by 180 degrees and go back. Once dynamic jumper links are added, the locations in the graph connected with such links now present a more challenging decision point for the robot. In these cases the robot uses the compass direction of the incoming call as the directive in choosing the correct direction to move in.

In the acyclic linear list representation classified all incoming calls into a node into "left calls" and "right calls." This division was useful since it kept the motion decision simple. The following algorithm makes use of simple graph theory to apply the two-directional motion decision to nodes with higher fanout as well. Therefore, even at junctions with higher fanout, the motion decision for the robot remains a simple choice.

The nodes in the graph are numbered consecutively in an increasing order as they are added to the list. If a node is numbered n then its immediate left neighbor is numbered $n-1$, and its immediate right neighbor is numbered

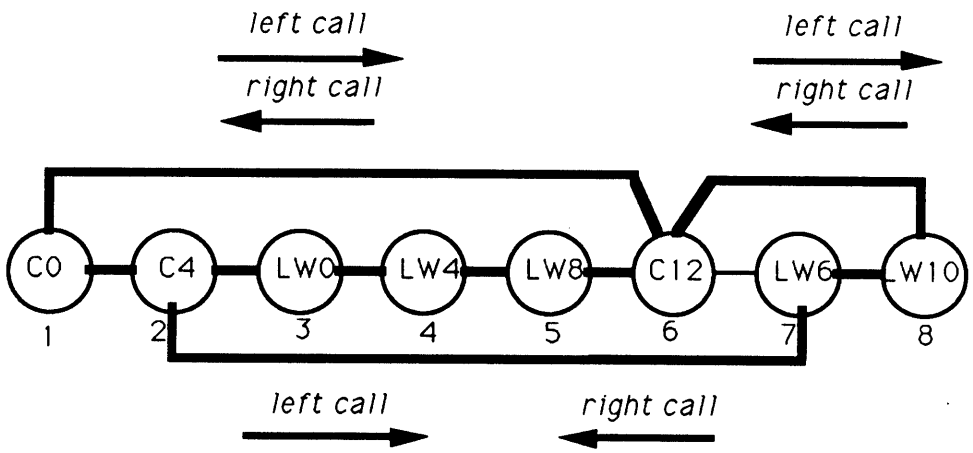


Figure 10.5: An illustration of the goal-call arbitration rule. Node allocation time, which serves as the node id number, is implicit in the node ordering in the list. A call arriving from a node with a "newer" node to an "older" node is a left call. Conversely, a call arriving from an "older" node to a "newer" node is a right call.

$n + 1$. Consequently, the nodes are ordered by discovery time. Any node on the left is "older" than all nodes on the right of it in the list. (Note that the sense in which a node is "old" is reversed from chronological age as reflected in its id number.)

A simple division of all calls can be made based on the ordinality of the call originator. If the recipient of a call has a smaller id than the immediate source of the call (i.e. an "older" node receives a call from a "newer" node) than the arriving call is necessarily a *right call.* Conversely, if the recipient of a call has a larger id than the immediate source of the call (i.e. a "newer" node receives a call from an "older" node) than the arriving call is necessarily a *left call.* Put simply:

If the ordinality of the recipient node is greater than that of the sender, it is a call from the left. Otherwise it is a call from the right.

As each node propagates a call, it sends in it its id number as well as the compass bearing. At most simple decision points the above described rule uses the id number to select the direction to move in. If the robot is at a node with a high fanout, and the call with the associated shortest path

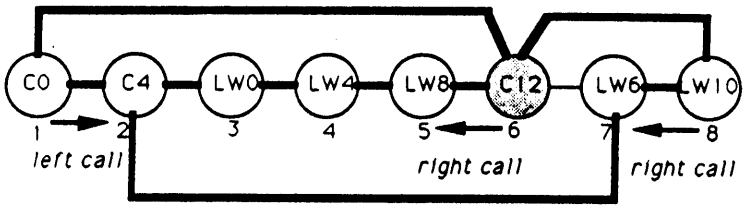


Figure 10.6: An example of goal-call propagation. The goal node, number 6, is shaded. It sends calls through all active linear and jumper links. The arrows indicate the direction of the call as it reaches a node.

requires a more complex turn, it uses the received compass bearing.

Figure 10.5 illustrates the goal-call arbitration rule as applied to the graph of the environment shown in figure 9.8. A specific example of goal-call propagation is shown in figure 10.6.

The introduction of cycles into the graph could result in a call to be propagated around the cycle indefinitely. However, this condition is remedied by the following:

1) Upon receiving a call each node in the graph stores it.

2) If the node is active, i.e. it corresponds to the robot's current position, it compares it to other received calls and takes the minimum looking for the physically shortest path. An active node does not propagate calls further since it is the destination. Therefore, all calls that reach the goal are terminated.

3) If the node is not active, it passes the call on in all directions (to the left and right if it is a simple path, and also to all jumper neighbors, if it is a higher fanout decision point). Since the weights on all edges are positive, going through a cycle necessarily increases the value of a call. Since the active node takes the minimal incoming call, it will never opt for a suboptimal choice. However, a call could continue to loop through a cycle and to increase monotonically. This condition is prevented by imposing an upper bound on the length of any call. Such a realistic bound can be chosen since the size of the graph is specified at compile time. Consequently, the cycle propagation problem is solved by terminating any calls longer than $O(n)$.

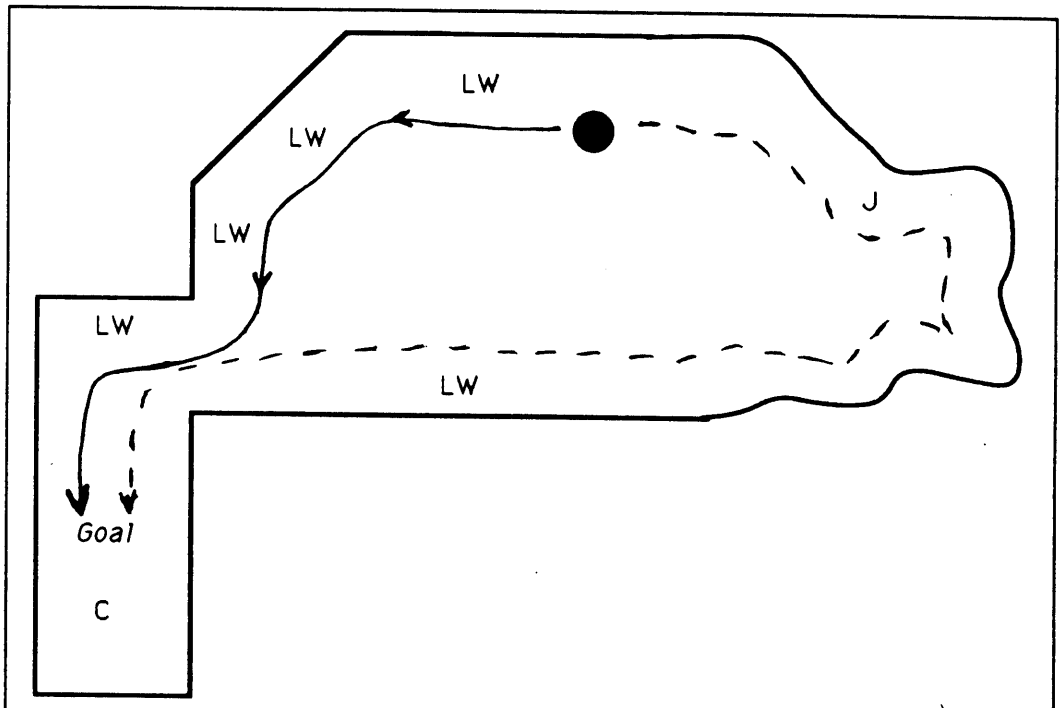


Figure 10.7: The ninth floor environment augmented with strategic clutter (indicated by landmark type J for "junk" or messy area) used for testing the robot's ability to optimize paths. The topologically shortest path (indicated with a dashed line) is longer than the physically shortest path (solid line). This situation tests the robot's ability to use the proper measure of optimality.

10.4 An Example of the Switchboard Performance

Integrating the switchboard mechanism into the system consisted of adding the *switching behavior*, as well as appropriate calls to it. No changes were made to any of the already existing software.

Figure 10.7 shows an environment used for testing path finding and optimization. After having explored the environment and learned its structure, the robot is told to return to the start location. As is illustrated in figure 10.8, the shortest topological path does not correspond to the shortest

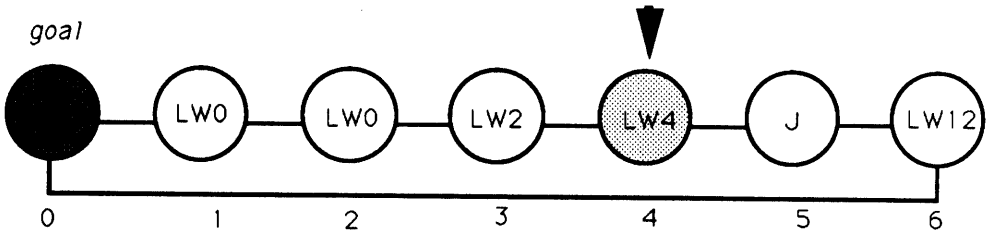


Figure 10.8: The map of the shown environment constructed by the robot. The arrow indicates the robot's current position.

physical one. Figure 10.9 shows the main parts of the trace of the robots path as it progresses toward the goal. The complete trace is shown in Appendix C. Using the estimated landmark length (see chapter 8) the robot correctly chooses the topologically longer but physically shorter path to the goal.

In testing the robustness of goal-oriented navigation, the robot was presented with various obstacles while on its mission to the selected goal. If the robot's path was blocked, its low-level navigation layer would assure that no collision occurred by turning the robot away from the obstacle. Simultaneously, the continuously emitted call from the goal forced the robot to turn in the direction of the desired path. The conflict of the two motivations results in taking the first free turn toward the desired direction. If the path to the goal is completely blocked, the robot abandons it after a certain time period in order to prevent endless oscillation.

10.5 Summary

The addition of the switchboard allows for implementing a general graph with dynamic links using a linear number of connections established at compilation time. The graph is capable of representing any physical topology the robot might encounter. Previously described localization and goal-directed navigation algorithms apply directly to this augmented graph structure.

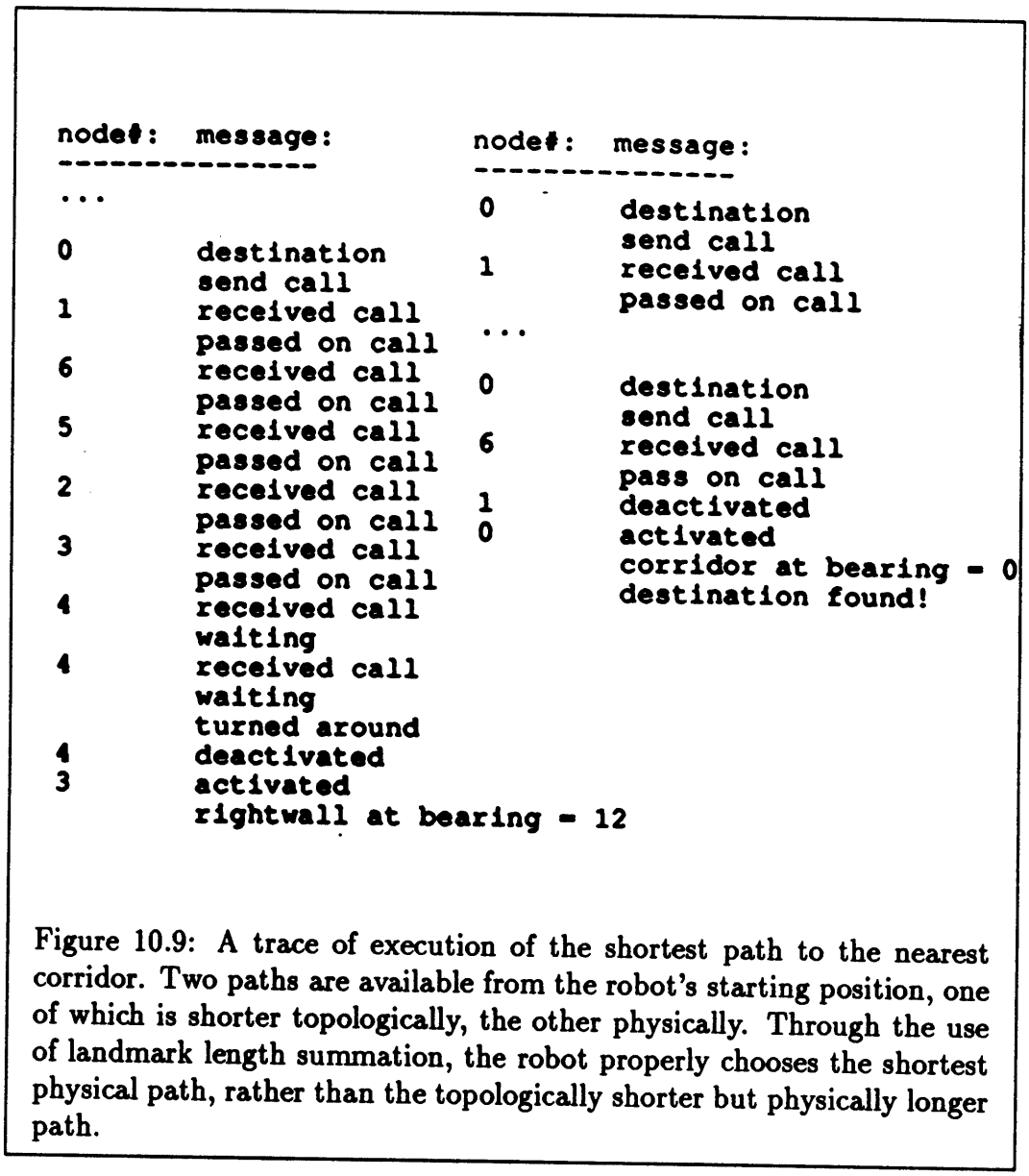


Figure 10.9: A trace of execution of the shortest path to the nearest corridor. Two paths are available from the robot's starting position, one of which is shorter topologically, the other physically. Through the use of landmark length summation, the robot properly chooses the shortest physical path, rather than the topologically shorter but physically longer path.

Chapter 11

Related Biology

The issues of navigation and spatial modeling in insects, animals, and people have long been studied by biologists and psychologists. A large body of experimental results and associated theories exists in those fields. A review of the related literature reveals a number of findings which are potentially applicable to the related navigation and representation problem in robotics. This chapter presents a brief review of such biological data.

11.1 The Topological Versus Metric Dilemma in Biology

A cognitive map is a generic term used by psychologists for the representation of spatial information. The psychological literature is divided on the issue of topological versus metric spatial representations. Studies presenting a variety of objects and testing response time show that adults are certainly capable of reconstructing Euclidian distance and constructing metric maps. However, these processes seem to take time and are more costly than topologically based tasks. It is certain that sufficient information is gathered and recorded to make metric inference possible. The computation does not seem to be performed routinely, though, but rather based on need. One hypothesis supposed that metric properties are results of inferential procedures [**McNamara 89**]. Other data show that roughly metric representations tend to be distorted based on local feature density and importance. Using a qualitative representation allows for a simplified matching algorithm for

localization within the world. Additionally, a convenient choice of a representation, such as a graph, will yield some of the topological information at no cost (e.g. adjacency and containment).

The nature of the representation determines the type and number of landmarks required for localizing. In a qualitative representation, an object can be remembered as being proximate to a landmark, which defines it within a rough circle around that landmark. On the other end of the spectrum, the position of the object can be computed precisely from the known locations of three landmarks. The question, as before, is "How much metric information is recorded?" A body of psychological test data points toward topological spatial representations in infants. These studies are especially interesting for testing the limits of qualitative representations. If it is indeed found that most spatial information in infants is qualitative, the belief in the necessity of analytical information will be weakened.

[Piaget and Inhelder 67] proposed, and later research supported, the hypothesis that early spatial knowledge (preschool years) is topological in nature. While spatial knowledge of that period is believed to be fundamentally non-metric, it relies on the presence of landmarks. More specifically, the landmarks must be close, so that the child can form relative distance relations among them **[Newcombe 88]**. This introduces a paradox: while children do not seem to record metric information, they must be using it to form relations between landmarks.

A possible solution lies in an abundance of landmarks, which allows for increased accuracy of localization even if fuzzy, rough metric relations are used. Given a large number of landmarks, a point can be found based on its relative rather and absolute distance. What results is a relational network. Psychological studies show that human spatial memory may be organized in this way **[Sadalla 88]**.

The qualitative versus quantitative dilemma is ubiquitous in both biology and robotics. The advantages and disadvantages of using one approach over the other are common to both fields. A compromise utilizing the benefits of both representations appears to be the most optimal solution. A topological model containing relevant quantitative information is the favored hypothesis of biological studies, as well as the representation of choice in many path planning mobile robot systems. The approach described in this thesis used such a representation.

11.2 Bats, Bees, Birds, and Rats

11.2.1 Bats

Bats are known for their extremely accurate ability to navigate and avoid obstacles using echolocation with multiple-frequency sonar data. Considering that bat habitats are very crowded, their ability to extract useful information from a multitude of available echoes indicates very clever, if not complex, sensory apparati.

Studies have shown that bats construct cognitive maps of the known environment. The existence of these maps was tested by allowing bats to learn a particular environment containing suspended wires. If the wires are unexpectedly removed, the bats continue to navigate around, as if they continue to be present **[Gallistel 89]**.

This behavior indicates the maintenance of some sort of an environment representation. Additionally, it shows the animal's dependence on the representation over the stimuli in the environment. Analogous experiments were performed in monkeys. When a treat is hidden in one of many available boxes, the monkey retrieves it readily if he witnessed the hiding process. Next, if the boxes are shuffled around, the monkey still tries the location of the original box, rather than the box itself **[Gallistel 80]**.

11.2.2 Bees

The behavior of bees has been intriguing biologists, behaviorists, and ethologists for centuries. Bee hives represent by far the most complex animal social organization, second only to human societies **[Gould 82]**. As such, they have been objects of many controlled studies.

The largest portion of a hive consists of nectar gatherers. These bees employ what appear to be sophisticated navigational techniques which allow them to depart far from the hive, search for and gather food, and then return home on the most direct available path. This behavior has coined the term "beeline."

Experiments have shown that bees use a relatively simple mechanism for locating goals, based on proximal landmarks, coupled with global references, such as the compass bearing and the polarization of sunlight. It appears that bees are capable of utilizing both route-specific information, such as

sequences of landmarks, as well as cognitive maps [**Gould 87**]. They tend to prefer absolute references, such as the position of the sun, but in the absence of those will successfully utilize relative landmarks [**Winston 87**]. Bees are able find a direct path home from an arbitrary location within a known terrain. This is exactly to be expected if they utilize internal maps of the environment. They are capable of utilizing polarized light, color, patterns, olfactory and magnetic information for navigation.

11.2.3 Birds

Birds provide interesting examples of redundant systems of simple navigational mechanisms. Studies of migratory birds show that they are equipped with innate maps of star patterns and seasonal directional preferences [**Kreithen 83**]. Homing pigeons use an entire repertoire of navigational techniques. In order of preference, they have at their disposal strategies using the polarization patterns of the sun, the magnetic field of the earth, wind direction, olfactory cues, and possibly ultrasound and other as yet unexplored sensory modalities.

11.2.4 Rats

Most interesting data have been gathered from rat experiments. Internal representations have been tested on rats in maze-running experiments. After allowing rats to familiarize themselves with a maze, the length of some corridors was altered. The rats ignored their sensory input and ran into the walls of the shortened corridors, and stopped before the end of the lengthened ones, at locations corresponding to their previous length [**Gallistel 80**].

In experiments with rotated radial mazes, rats enter already sampled arms without realizing the redundancy of their actions. This evidence points toward a metric representation of space, utilizing absolute angles and distances, as opposed to a topological representation relying on adjacency relations. In order to establish global references, the rats used external landmarks available in the environment.

A more challenging experiment allowed the rats to learn a maze and then tested the rats in a forced detour experiment. The rats were able to find alternate as well as short cut routes to the goal. Most interestingly, they

succeeded in doing so in the dark, when they could not rely on any visible landmarks.

One of the most famous rat navigation experiments showed that even when given perceptible sensory features, animals seem to prefer to find the goal using the cognitive maps they have developed [**Morris, Garrud, Rawlins, and O'Keefe 82**]. The experiment involved placing a rat into a circular vat of opaque liquid and allowing it to locate a submerged platform. In subsequent trials, even if the platform was moved to a new but visible location, the rat preferred to locate it based on previously established landmarks.

Previously thought of as primitive, insect and animal spatial representations have proven to be very well adapted to their navigational tasks. Elaborate sensory systems are designed for functional redundancy. Special-purpose computational hardware appears to exist for various routine tasks, such as position and angular displacement from a reference point. Both topological and metric information is represented and accessible.

11.3 Models of Spatial Memory

11.3.1 Hierarchical Models

The impressive efficiency observed in the navigation performance of biological systems has lead to various speculations about the way spatial memory is organized. Many proposed models of spatial memory share features in common with models of semantic memory [**Sadalla 88**]. The main similarity is their hierarchical nature. In a model of semantic memory proposed by [**Quillian 67**] concepts are organized along ascending levels of a hierarchical graph. Each level in the representation corresponds to appropriate semantic attributes. Analogous models of spatial memory were proposed, in which locations (based on their position relative to known landmarks) are clustered based on regions. Relations between regions are established on higher levels of the spatial hierarchy. Again analogous to semantic networks, the models of hierarchical spatial networks were tested by measuring a person's response time in establishing relationships between members of equal or different levels. Experiments have included tests on spatial layouts of objects, as well as locations of objects on maps [**McNamara, Hardy and Hirtle 89**]. In spite of the absence of any physical and perceptual boundaries in the spatial stimuli, region boundaries were assigned by the viewer. The reported latency

in object and location recall correlated positively with the assumed spatial hierarchy.

11.3.2 Deutsch's Topological Model

Based on rat navigation experiments, Deutsch proposed a purely topological model of spatial representation **[Gallistel 80]**. The method represents locations in the world as nodes in the graph connected by appropriate adjacency links. The animal is assumed to be localized by always having several such locations in view. Each perceived location can serve as a reference point. The rat is assumed to align with and head toward the most desirable location. The desirability of location is determined by *motivation* which is propagated throughout the graph. The motivation is the highest at the goal, and the rat performs gradient descent search to reach it.

The weakness of the approach lies in its purely topological nature. Biological studies have shown that the rat's representation of familiar spaces are more Euclidian or metric, than topological. More specifically, rats seem to utilize the metric qualities (such as distances and angles) of their internal representations over topological ones, when the two are in conflict.

11.3.3 Zipser's Model

The hippocampus is the area of the brain strongly associated with memory and spatial learning. It has been shown to contain so-called "place cells" or neurons which fire in relation to the location of the animal in the known environment **[Zipser 86]**. Conversely, the area of the environment that corresponds to the firing cell is called the "place field" of that neuron. If a place field is altered through reorganizing landmarks or removing them, the neuron no longer responds to it. Hippocampal lesions dramatically impair learning and memory.

Zipser presents a computational model of hippocampal place fields with the goal of biological plausibility. His model relates the configuration of distal landmarks in the environment to the location, size and shape of place fields **[Zipser 86]**. A model of a location is analogous to a radial proximity sensor signature. The purpose of the model is to match locations in the world with an internal representation of landmarks. A two-layer neural network is used for this task. Zipser acknowledges that purely metric signatures do

not correspond to the way landmarks are truly selected, at least as shown in studies with rats. In his model, landmarks can consist of both distance relations between perceived objects, and relations between the observer and the objects.

Zipser suggests incorporating direction information into place fields to aid goal-oriented navigation. Such oriented view fields have been found in rat hippocampuses.

The problem that arises with this type of spatial model is that of resolution. It is clearly impossible to have a unique field view for every place in the environment. Consequently, some process of abstraction takes place, which is equally difficult to analyze as the landmark selection process. Having selected particular locations as salient, the navigating individual or robot is almost never at the exact location of the corresponding place field. This further complicates the matching scheme. Without an absolute direction reference, the process becomes quite complex.

11.3.4 Global Versus Distributed Neural Models

The representational dichotomy carries into the neural model community as well. A direct translation of landmark locations to specific neurons corresponds to many models of spatial maps. Such representation directly preserves the spatial topological layout in the structure of the neuronal network.

Neurophysiological data gathered from rats seems to indicate a distributed representation on a still lower level. Even the individual landmarks are distributed. Rather than having specific neurons or network patches correspond to particular locations, the entire hippocampal network participates in representing various aspects of the mapped region **[Eichenbaum and Cohen 88]**. Place fields appear to be distributed evenly around the mapped environment. Often, a single neuron was found to have multiple place fields.

The neurophysiological findings indicate that while locations of external stimuli are mapped topographically onto the primary sensory areas (i.e. the retina), egocentric maps are not projected onto the hippocampus.

The distributed neuronal evidence is in accordance with parallel distributed models of neural processing **[McClelland and Rumelhart 86]**. **[Eichenbaum and Cohen 88]** point out their advantage: such a distributed representation of space lends the system additional generality. Such a system can store a variety of relationships, rather than being limited to spatial

modeling.

11.4 Summary

Most animals, including humans, spend much of their waking time in transit **[Waterman 89]** from one place to another. It comes as no surprise that biological systems for spatial modeling and goal-oriented navigation have evolved to perform with impressive robustness.

A review of biological data shows that insects, animals, and people use cognitive maps as internal representations of spatial information. The maps have been shown to contain both topological and metric information. Most studies agree on a similar model of landmark-based spatial layouts augmented with distance and rotational angle information. The cognitive maps correspond to the environmental maps constructed and used by mobile robots. Like many biological and robotic systems, the robot described in this work utilizes a combination of qualitative and quantitative information in order to optimize various forms of task-specific computation.

Within the goal of neural feasibility, spatial representation models have been mapped to networks of neurons. The models vary in the level of distributedness, but they all share a common, neurally-inspired decentralized nature. It is precisely this distributed nature that is being explored in the navigation approach described in this thesis.

Chapter 12

Conclusion and Future Work

12.1 Directions for Future Work

A variety of possible improvements as well as direction for future study are described. Some relate to the details of the algorithm, while others address more global issues of the approach.

12.1.1 Fine Tuning the Representation

The map learning algorithm can be further improved by the addition of a "usefulness" measure for each node in the graph. The usefulness would be increased with each traversal through the node, and would decay with time. Assuming the robot is used for a task which requires repeated traversals of many different locations in the environment (such as a delivery task or plant watering), the usefulness measure will isolate the most commonly traversed areas of the graph. Nodes which decay past a certain threshold can be assumed to be either on circuitous paths, or incorrectly allocated in the first place.

A garbage collection algorithm can then be added to the network, which would continuously scan the graph for "useless" nodes. These would be removed from the graph through a process of deallocation and graph compaction.

12.1.2 Testing Generality

Generality is one of the strengths of the map-learning and path planning approach presented in this thesis. The algorithm expects landmarks as inputs, regardless of their type. It is designed for robust functionality in spite of an impoverished as well as an inaccurate sensory system. A possible direction of future research would be to apply the algorithm on a system with a greater sensor variety. In a system with a wealth of sensory data the landmarks can be characterized with more attributes. Increased landmark descriptors would generate more landmark types which would, in turn, simplify or eliminate many localization ambiguities.

12.1.3 Adding Reasoning

The method described here is based on distributed topological information. However, with the use of the available geometric information, in the form of rough position estimates, new topological data can be obtained through making geometric inferences. This is precisely the idea behind short cuts. If each landmark in the graph has some geometric range of absolute positions associated with it, a rough comparison can be performed between unconnected nodes in the graph. Discovering physical proximity yields new neighborhood information. A full maze-searching algorithm, which is what the boundary-tracing approach is based on, is guaranteed to find all topologically connected locations. Alternative, the information can be inferred, and then confirmed through physical trials.

12.1.4 Optimizations

The object code controlling the robot is notably small. A network of 10 nodes took up 51K bytes. The division between code and data is blurred since the graph representation, which comprises most of the code, is actually data. It is important to note that there is not separate environment learning and path finding engine. All of the reasoning is contained in the graph itself.

It would be interesting to test the system in a variety of typical office environments in order to obtain an estimate of the average generated graph size. Next it would be beneficial to compare the size of the produced code to that of traditional systems with similar applications.

12.1.5 Achieving True Parallelism

A natural extension to this work in transferring the distributed system into truly distributed hardware, in order to properly measure its parallel performance. An extreme case of distributedness would be to implement a system of interacting physical agents which, as a collective, represent the global world model. Each agent can store a piece of the network, or even a single landmark. This would allow for simple, computationally light robots, to learn large, complex spaces, and navigate within them. Communication is the main problem in such a system. Given a method for message-passing among agents, it would be possible to test the map construction and goal-directed navigation on a fully distributed system.

12.2 The Contribution

This thesis has addressed a number of issues relevant to mobile robot navigation and path planning. Emphasis was placed on the following:

- distributed versus global representation,
- qualitative versus quantitative computation
- qualitative versus quantitative representation,
- procedural versus declarative representation,
- design of emergent behaviors,
- dynamic versus static landmark matching,
- minimizing and simplifying communication.

The primary goal of this research was to explore a parallel distributed approach to spatial learning and navigation. The approach emphasized qualitative computation and representation as an alternative to cartesian or metric data manipulation. Rather than focusing on a portion of the goal-oriented navigation problem, such as path planning, a complete system was implemented which combines interrelated solutions to collision-free navigation,

landmark detection, map building, and path planning. The system is constructed as a hierarchy of three layers of competence.

The lowest layer combines simple, intuitive rules to obtain robust emergent collision-free boundary-tracing behavior. As an alternative to less intuitive approaches, this method employed a combination of cooperating reflex-like rules. The rules were designed based on a simple but sufficient functional sensor characterization. They were added to the system incrementally thus preserving tractability. Each of the rules was triggered by mutually exclusive environmental conditions which eliminated the need for explicit arbitration among behaviors. Such a simple interaction scheme was useful for incremental testing of the robot's performance.

The middle layer in the competency hierarchy utilizes the boundary-tracing feature of the low-level navigation level to extract features in the environment. The robot monitors its own motion to find landmarks in the environment. Landmark types were selected as large, permanent, robustly-detectable environmental features such as walls and corridors. Since the landmark set is very sparse, landmark disambiguation is accomplished through a limited use of context (expectation) and a very rough position estimate.

In contrast to static landmark detection approaches which use model matching, this method employed an implicit, procedural representation of landmarks. The method is based on continuous updating of confidence levels associated with features detected as the robot is moving. The landmark detection layer utilizes the nature of the boundary-tracting layer bellow, but does not explicitly communicate with it. Both of the procedural methods employed for low-level navigation and landmark detection are sensor independent and are easily portable to other proximity sensor systems.

The top layer in the hierarchy uses the detected landmarks to construct a distributed map. The approach utilizes an undirected cyclic graph as a universal embedding for any physical topology the robot might encounter. Since the topology of the graph is fixed at compilation, a method for embedding the graph into a linear list was presented. The structure of the environment, coupled with the nature of the navigation algorithm, were used to simplify the necessary connectivity of the graph, which was effectively bounded to be linear in the number of nodes.

The graph is distributed in that each node in it is a concurrently acting behavior representing a particular landmark in the world. The reasoning engine is embedded into the spatial representation. The parallel representation

allows for constant time localization and linear time shortest path finding. Communication and goal-orientation is performed through message broadcasting and spreading of activation through the graph. This method propagates globally optimal direction information to every node in the graph. Consequently, the robot uses only local information for a globally optimal path selection. Additionally, replanning is only required when the goal changes. The method scales well, due to the parallel localization and path finding algorithms, as well as the overall small implementation.

Working with a physical robot provided a variety of unexpected challenges which firmly grounded the emphasis of the research. It required the project to proceed by constantly satisfying both the top-down constraints of the top-level goal, and the bottom-up limitations of the physical implementation.

The research described in this thesis explored an alternative approach to goal-oriented navigation and showed a possible direction of research orthogonal to what is considered to be the classical method. The described approach may have biological implications and be better suited for certain mobile robot applications.

Bibliography

[Angle 89] "Genghis, a Six Legged Autonomous Walking Robot", Colin M. Angle, *MIT S.B. Thesis in Electrical Engineering and Computer Science*, March 1989.

[Anooshian 88] "Places Versus Procedures in Spatial Cognition: Alternative Approaches to Defining and Remembering Landmarks", Linda J. Anooshian, *British Journal of Developmental Psychology*, 1988: 6, 389-390, The British Psychological Society.

[Arkin 89] "Towards the Unification of Navigational Planning and Reactive Control", Ronald C. Arkin, *AAAI Spring Symposium on Robot Navigation Working Notes*, March 1989, 1-5.

[Arkin 87] "Motor Schema Based Navigation for a Mobile Robot: An Approach to Programming by Behavior", R. C. Arkin, *Proceedings of the 1987 IEEE International Conference on Robotics and Automation*, 264-271.

[Braunegg 90] "MARVEL: A System for Recognizing World Locations with Stereo Vision", David J. Braunegg, *MIT Artificial Intelligence Lab Technical Report 1229*, April 90.

[Brooks 90] "The Behavior Language; User's Guide", Rodney A. Brooks, *MIT Artificial Intelligence Lab Memo*, to appear, 1990.

[Brooks 89] "A Robot that Walks; Emergent Behavior from a Carefully Evolved Network", Rodney A. Brooks, *Neural Computation*, 1:2.

[Brooks 87] "A Hardware Retargetable Distributed Layered Architecture for Mobile Robot Control", Rodney A. Brooks, *Proceedings 1987 IEEE International Conference on Robotics and Automation*, Raleigh, NC, April 1987, 106-110.

[Brooks 86] "A Robust Layered Control System for a Mobile Robot", Rodney A. Brooks, *IEEE Journal of Robotics and Automation*, RA-2, April 1986, 14-23.

[Brooks 85] "Visual Map Making for a Mobile Robot", Rodney A. Brooks, *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, March 1985, 824-829.

[Brooks 83] "Solving the Find-Path Problem by Good Representation of Free Space", Rodney A. Brooks, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-13, No. 3, April 1983.

[Brooks and Connell 86] "Asynchronous Distributed Control System for a Mobile Robot", R. A. Brooks and J. H. Connell, *SPIE's Cambridge Symposium on Optical and Opto-Electronic Engineering Proceedings*, Vol. 727, October 1986.

[Brooks and Flynn 89] "Robot Being", Rodney A. Brooks and Anita M. Flynn, *NATO Workshop on Robotics and Biological Systems*, Tuscany, Italy, July 1989.

[Canny and Donald 87] "Simplified Voronoi diagrams", John F. Canny and Bruce Donald, *Proceedings of the 28th IEEE Symp. FOCS*, October 1987.

[Chatila and Laumond 85] "Position Referencing and Consistent World Modeling for Mobile Robots", R. Chatila and J. Laumond, *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, March 1985.

[Ciholas 88] "A Dual-Processor Vision System for Real-Time Stereo and Motion", Michel E. Ciholas, *MIT S.M. Thesis in Electrical Engineering and Computer Science*, June 1988.

[Connell 89] "A Colony Architecture for an Artificial Creature", J. H. Connell, *MIT Artificial Intelligence Lab Technical Report 1151*, October 1989.

[Connell 88] "Navigation by Path Remembering", J. H. Connell, *SPIE 1988 Mobile Robots III Proceedings*, November 1988.

[Connell 87] "Creature Design with the Subsumption Architecture", J. H. Connell, *IJCAI-87 Proceedings*, Vol. 727, October 1987, 77-84.

[Crowley 85] "Dynamic World Modeling for an Intelligent Mobile Robot Using a Rotating Ultra-Sonic Ranging Device", J. L. Crowley, *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, St. Louis, MO, March 1985.

[Drumheller 87] "Mobile Robot Localization Using Sonar", M. Drumheller, *IEEE Transactions on PAMI*, Vol. PAMI-9, No. 2, March 1987.

[Durrant-Whyte and Leonard 89] "Navigation by Correlating Geometric Sensor Data", Hugh F. Durrant-Whyte and John J. Leonard, *IEEE/RSJ International Workshop on Intelligent Robots and Systems*, Tsukuba, Japan, September 4, 1989, 440-447.

[Eichenbaum and Cohen 88] "Representation in the Hippocampus: What Do Hippocampal Neurons Code?", Howard Eichenbaum and Neal J. Cohen, *Trends in Neuroscience*, Vol 11, No. 6, 1988.

[Eichenbaum, Wiener, Shapiro, and Cohen 89] "The Organization of Spatial Coding in the Hippocampus: A Study of Neural Ensemble Activity", H. Eichenbaum, S. I. Wiener, M. L. Shapiro, and N. H. Cohen, *The Journal of Neuroscience*, 9(8):2764-2775.

[Elfes 86] "A Sonar-Based Mapping and Navigation System", Alberto Elfes, *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, February 1986.

[Everett, Gilbreath, and Bianchini 88] "Environmental Modeling for a Mobile Sentry Robot", H. R. Everett, G. A. Gilbreath, and G. L. Bianchini, *NOSC Technical Document 1230*, January 1988.

[Faverjon 84] "Obstacle Avoidance Using an Octree in the Configuration Space of a Manipulator", Bernard Faverjon, *Proceedings of IEEE Conference on Robotics and Automation*, 504-512, March 1984.

[Flynn 88] "Combining Sonar and Infrared Sensors for Mobile Robot Navigation", Anita M. Flynn, *International Journal of Robotics Research*, December 1988.

[Flynn 85] "Redundant Sensors for Mobile Robot Navigation", A. M. Flynn, *Massachusetts Institute of Technology AI Lab Technical Report 859*, September 1985.

[Flynn, Brooks, Wells, and Barrett 89] "Squirt: The Prototypical Mobile Robot for Autonomous Graduate Students", Anita M. Flynn, Rodney A. Brooks, William M. Wells, and David S. Barrett, *MIT AI Memo 1120*, July 1989.

[Gallistel 89] "Animal Cognition: The Representation of Space, Time and Number", C. R. Gallistel, *Annual Review of Psychology 1989*, Vol 40, Palo Alto, California, 155-189.

[Gallistel 80] "The Organization of Action: A New Synthesis", C. R. Gallistel, *Lawrence Erlbaum Associates*, Hillsdale, New Jersey.

[Giralt, Chatila and Vaisset 83] "An Integrated Navigation and Motion Control System for Autonomous Multisensory Mobile Robots", G. Giralt, R. Chatila and M. Vaisset, *First International Symposium on Robotics Research*, M. Brady and R. Paul (eds.), MIT Press, Cambridge, MA, 1983.

[Gould 87] "Flower-shape, Landmark, and Locale Memory in Honeybees", James L. Gould, *Neurobiology and Behavior of Honeybees*, R. Menzel and A. Mercer, eds., Springer Verlag, 1987.

[Gould 82] "Ethology: The Mechanisms and Evolution of Behavior", James L. Gould, *W. W. Norton and Company, New York*, 1982.

[Grimson and Lozano-Pérez 87] "Localizing Overlapping Parts by Searching the Interpretation Tree", W. E. L. Grimson and T. Lozano-Pérez, *IEEE Transactions on PAMI*, Vol. PAMI-9, No. 4, July 1987.

[Horswill and Brooks 88] "Situated Vision in a Dynamic World: Chasing Objects", AAAI-88, *St. Paul, MN*, August 1988, 796-800.

[Kender and Leff 89] "Why Direction-Giving is Hard: The Complexity of Linear Navigation by Landmarks", John R. Kender and Avraham Leff, *AAAI Spring Symposium on Robot Navigation Working Notes*, March 1989, 38-42.

[Khatib 86] "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots", Oussama Khatib, *The Internation Journal on Robotics Research*, Vol. 5, No. 1, Spring 1986.

[Kreithen 83] "Orientational Strategies in Birds", Melvin L. Kreithen, *Behavioral Energetics: The Cost of Survival in Vertebrates*, Aspey and Lustick, eds., Ohio State University Press, 1983.

[Kuc and Di 86] "Intelligent Sensor Approach to Differentiating Sonar Reflections From Corners and Planes", R. Kuc and Y. Di, *International Congress on Intelligent Autonomous Systems Proceedings*, Amsterdam, The Netherlands, 1986.

[Kuc and Siegel 87] "Physically Based Simulation Model for Acoustic Sensor Robot Navigation", R. Kuc and Y. Di, *IEEE Transactions on PAMI*, 1987.

[Kuipers 87] "A Qualitative Approach To Robot Exploration and Map Learning", Benjamin J. Kuipers, *AAAI Workshop on Spatial Reasoning and Multi-Sensor Fusion*, October 1987.

[Kuipers 79] "Commonsense Knowledge of Space: Learning from Experience", Benjamin J. Kuipers, *IJCAI-79 Proceedings*, Tokyo, Japan, August 1979, 499-501.

[Kuipers and Byun 88] "A Robust, Qualitative Approach to a Spatial Learning Mobile Robot", Benjamin J. Kuipers and Yung-Tai Byun, *SPIE Advances in Intelligent Robotics Systems Proceedings*, November 1988.

[Letovsky 84] "Interpreting Range Data For a Mobile Robot", S. Letovsky, *CSCSI-SCEIO Proceedings*, London, Ontario, 1984.

[Lockman 88] "Toward an Ecological Conception of Landmarks: A Developmental Perspective", Jeffrey J. Lockman, *British Journal of Developmental Psychology*, 1988: 6, The British Psychological Society, 381-383.

[Lozano-Pérez 87] "A Simple Motion-Planning Algorithm for General Robot Manipulation", Tomás Lozano-Pérez, *IEEE Journal of Robotics and Automation*, Vol. RA-3, Mo. 3, June 1987.

[Lozano-Pérez and Wesley 79] "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles", Tomás Lozano-Pérez and Michael A. Wesley, *Communications of the ACM October 1979*, Volume 22 No. 10.

[Lozano-Pérez 81] "Automatic Planning of Manipulator Transfer Movements", Tomás Lozano-Pérez, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-11, No. 10, October 1981.

[Mataric 90] "Environment Learning Using a Distributed Representation", Maja J Mataric, *Proceedings of 1990 IEEE International Conference on Robotics and Automation*, May 1990.

[Mataric 89] "Qualitative Sonar Based Environment Learning for Mobile Robots", Maja J Mataric, *SPIE Mobile Robots IV Proceedings*, November 1989.

[Mataric and Brooks 90] "Learning a Distributed Map Representation Based on navigation Behaviors", Maja J Mataric and Rodney A. Brooks, *Proceedings of 1990 USA-Japan Symposium on Flexible Automation*, June 1990.

[McClelland and Rumelhart 86] "Parallel Distributed Processing", James L. McClelland and David E. Rumelhart, eds., *MIT Press*, Cambridge, Massachusetts.

[McNamara 89] "Mental Representations of Spatial and Nonspatial Relations", Timothy P. McNamara, *AAAI Spring Symposium on Robot Navigation Working Notes*, March 1989, 51-52.

[McNamara, Hardy and Hirtle 89] "Subjective Hierarchies in Spatial Memory", Timothy P. McNamara, James K. Hardy and Stephen C. Hirtle, *Journal of Experimental Psychology: Learning, Memory, and Cognition 1989*, Vol. 15 No. 2, 211-227.

[Moravec 88] "Sensor Fusion in Certainty Grids for Mobile Robots", Hans P. Moravec, *AI Magazine, 9:2, Summer 1988*, 61-74.

[Moravec 83] "The Stanford Cart and the CMU Rover", IEEE Proceedings, *Hans P. Moravec*, Vol. 71, No. 7, July 1983, 872-884.

[Moravec 81] "Rover Visual Obstacle Avoidance", IJCAI-7 Proceedings, *Hans P. Moravec*, 1981.

[Moravec and Cho 89] "A Bayesian Method for Certainty Grids", Hans P. Moravec and Dong Woo Cho, *AAAI Spring Symposium on Robot Navigation Working Notes*, March 1989, 57-60.

[Moravec and Elfes 85] "High Resolution Maps From Wide Angle Sonar", Proceedings of the 1985 IEEE International Conference on Robotics and Automation, *Hans P. Moravec and Alberto Elfes*, St. Louis, MO, March 1985.

[Müller and Wehner 88] "Path Integration in Desert Ants, Cataglyphis Fortis", Martin Müller and Rüdiger Wehner, *Proceedings Natural Academy of Sciences*, 1988.

[Newcombe 88] "The Paradox of Proximity in Early Spatial Representation", Nora Newcombe, *British Journal of Developmental Psychology*, 1988:

6, The British Psychological Society, 376-378.

[Payton 88] "Internalized Plans: a representation for action resources", Workshop on Representation and Learning in an Autonomous Agent, *November*, 1988.

[Piaget and Inhelder 67] "The Child's Conception of Space", J. Piaget and B. Inhelder, *New York*, Norton.

[Pick, Montello and Somerville 88] "Landmarks and the Coordination and Integration of Spatial Information", Herbert L. Pick, Jr., Daniel R. Montello, and Susan C. Somerville, *British Journal of Developmental Psychology*, 1988: 6, The British Psychological Society, 372-375.

[Polaroid 87] "Polaroid Ultrasonic Ranging System Handbook", Polaroid Corporation, *Application Notes and Technical Papers*, 1987.

[Presson and Montello 88] "Points of Reference in Spatial Cognition: Stalking the Elusive Landmark", Clark C. Presson and Daniel R. Montello, *British Journal of Developmental Psychology*, 1988: 6, The British Psychological Society, 378-381.

[Sadalla 88] "Landmarks in Memory", Edward K. Sadalla, *British Journal of Developmental Psychology*, 1988: 6, The British Psychological Society, 386-388.

[Sarachik 89] "Visual Navigation: Constructing and Utilizing Simple Maps of an Indoor Environment", Karen B. Sarachik, *MIT Artificial Intelligence Lab Technical Report 1113*, March 1989.

[Schöne 84] "Spatial Orientation; The Spatial Control of Behavior in Animals and Man", Hermann Schöne, *Princeton University Press*, Princeton, New Jersey, 1984.

[Stewart 88] "Multisensor Modeling Underwater with Uncertain Information", W. Kenneth Stewart, Jr., *MIT Artificial Intelligence Lab Technical Report 1143*, July 1988.

[Viola 90] "Neurally Inspired Plasticity in Oculomotor Processes", Paul Viola, *Proceedings of the 1990 IEEE Conference on Neural Information Processing Systems-Natural and Synthetic*, Denver, CO, April 1990.

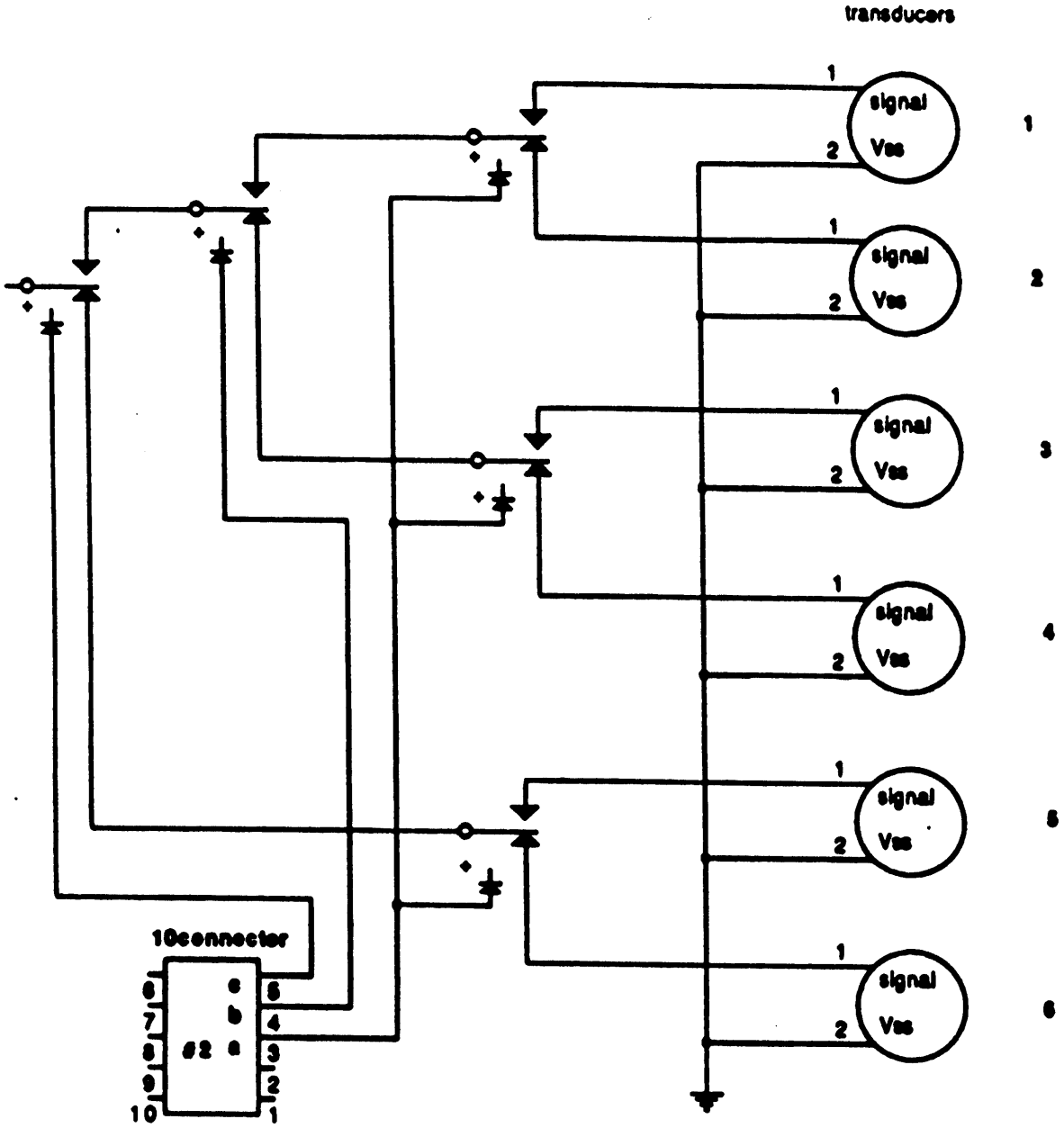
[Quillian 69] "Semantic Memroy", M. Ross Quillian, *Semantic Information Processing, Marvin Minsky, ed.*, MIT Press, Cambridge, MA, 1969, 227-270.

[Waterman 89] "Animal Navigation", Talbot H. Waterman, *Scientific American Library*, New York, 1989.

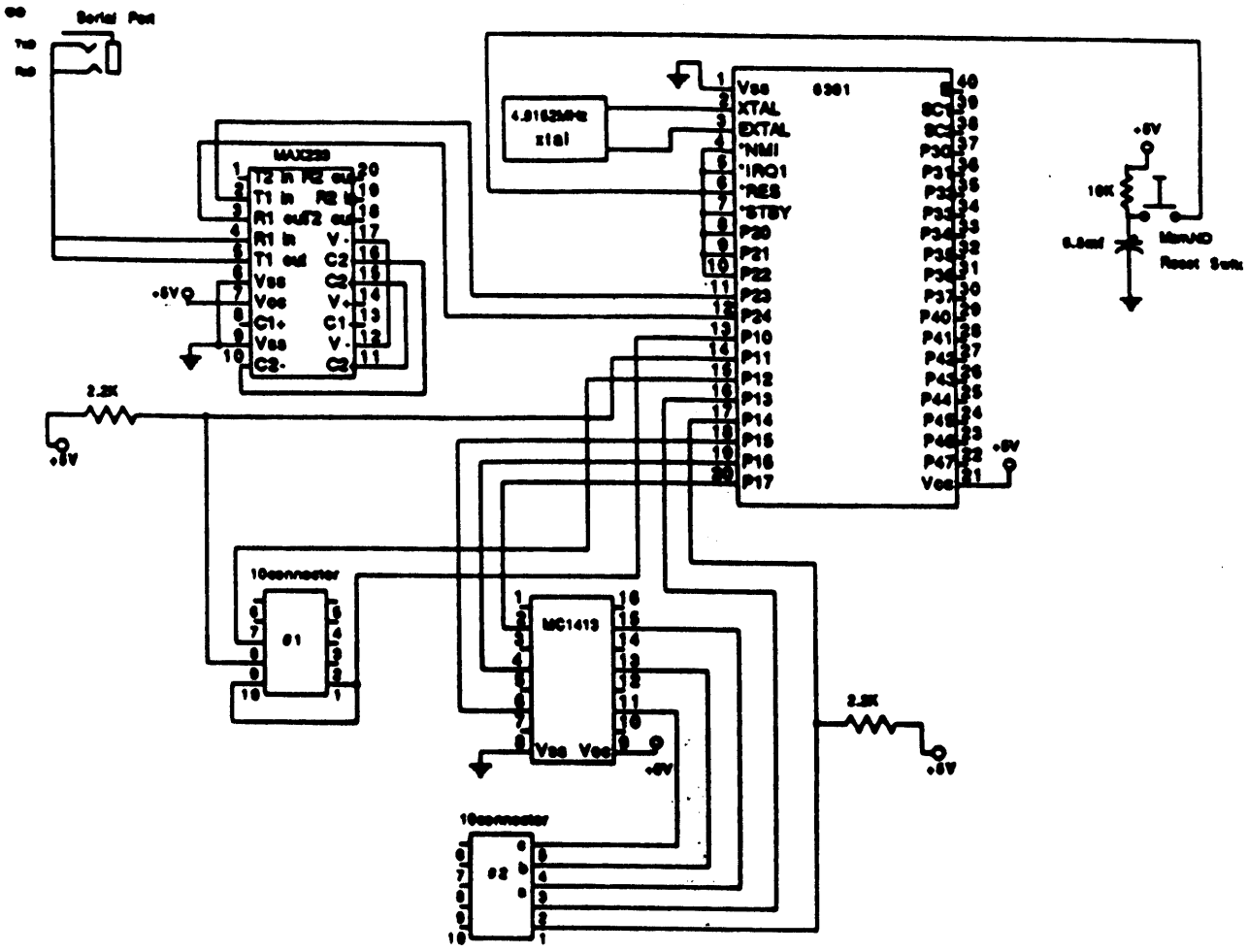
[Winston 87] "The Biology of the Honey Bee", Mark Winston, *Harvard University Press*, 1987.

[Zipser 86] "Biologically Plausible Models of Place Recognition and Goal Location", D. Zipser, *Parallel Distributed Processing, Vol 2: Psychological and Biological Models, Chapter 43*, McClelland and Rumelhart, eds., MIT Press, Cambridge, MA, 433-470.

Appendix A: The Relay Switching Mechanism



Appendix B: The Schematic of the Sonar Processor Board



Appendix C: A Trace of Shortest-Path Goal-Directed Navigation

```
node#:  message:
---------------
...

0       destination
        send call
1       received call
        passed on call
6       received call
        passed on call
5       received call
        passed on call
2       received call
        passed on call
3       received call
        passed on call
4       received call
        waiting
4       received call
        waiting
        turned around
4       deactivated
3       activated
        rightwall at bearing = 12
0       destination
        send call
1       received call
        passed on call
6       received call
        passed on call
5       received call
        passed on call
2       received call
        passed on call
3       received call
        waiting
4       received call
        passed on call
3       received call
        waiting
3       deactivated
2       activated
        rightwall at bearing = 10
```

```
0       destination
        send call
1       received call
        passed on call
6       received call
        passed on call
5       received call
        passed on call
2       received call
        waiting
4       received call
        passed on call
3       received call
        passed on call
2       received call
        waiting
0       destination
        send call
2       deactivated
1       activated
        leftwall at bearing = 0
6       received call
        send call
1       received call
        waiting
5       received call
        passed on call
4       received call
        passed on call
3       received call
        passed on call
2       received call
        passed on call
1       received call
        waiting
0       destination
        send call
6       received call
        pass on call
1       deactivated
0       activated
        corridor at bearing = 0
        destination found!
```